第3章 物联网软件开发管理

本章重点介绍物联网开发团队、项目进度控制、项目成本估算与控制、软件质量管理，要求学生了解物联网软件开发的过程和项目进度控制、成本控制、质量管理的相关内容。

3.1 物联网开发团队

本节重点介绍个人软件过程、团队软件过程、软件项目组以及微软软件开发团队。

3.1.1 个人软件过程

1. 概述

软件工程师都知道，要开发高质量的软件，必须改进软件生产的过程。软件生产能力成熟度模型 SW-CMM(简称 CMM)是当前最好的软件过程，并且 CMM 已经成为软件过程工业标准。但是，CMM 仅提供了一个有力的软件过程改进框架，却只告诉人们“应该做什么”，而没有告诉人们“应该怎样做”，并未提供有关实现关键过程域所需要的具体知识和技能。为了弥补这个欠缺，Humphrey 主持开发了个人软件过程(Personal Software Process，PSP)。

PSP 是一种可用于控制、管理和改进个人工作方式的自我持续改进过程，是一个包括软件开发表格、指南和规程的结构化框架。PSP 与具体的技术(程序设计语言、工具或者设计方法)相对独立，其原则能够应用到软件工程任务之中。PSP 说明了个人软件过程的原则；帮助软件工程师做出准确的计划；确定软件工程师为改善产品质量要采取的步骤；建立度量个人软件过程改善的基准；确定过程的改变对软件工程师能力的影响。

在 CMM 1.1 版本的 18 个关键过程域中有 12 个与 PSP 有关，据统计，软件项目开发成本的 70%取决于软件开发人员个人的技能、经验和工作习惯。因此，软件开发人员若能接受 PSP 培训，对软件能力成熟度的升级是一个有力的保证。面向软件开发单位，CMM 侧重于软件企业中有关软件过程的宏观管理；面向软件开发人员，PSP 则侧重于企业中有关软件过程的微观优化。二者互相支持，互相补充，缺一不可。

按照 PSP 规程，改进软件过程的步骤首先需要明确质量目标，也就是软件将要在功能和性能上满足要求和用户潜在的需求。接着就是度量产品质量，有了目标还不行，目标只是一个原则性的东西，还不便于实际操作和判断，因此，必须对目标进行分解和度量，使软件质

量能够“测量”。然后就是理解当前过程，查找问题，并对过程进行调整。最后应用调整后的过程度量实践结果，将结果与目标做比较，找出差距，分析原因，对软件过程进行持续改进。

2. PSP 等级

就像 CMM 为软件企业的能力提供一个阶梯式的进化框架一样，PSP 为个人的能力也提供了一个阶梯式的进化框架，以循序渐进的方法介绍过程的概念，每一级别都包含了更低一级别中的所有元素，并增加了新的元素，如图 3-1 所示。这个进化框架是学习 PSP 过程基本概念的好方法，它赋予软件人员度量和分析工具，使其清楚地认识到自己的表现和潜力，从而可以提高自己的技能和水平。

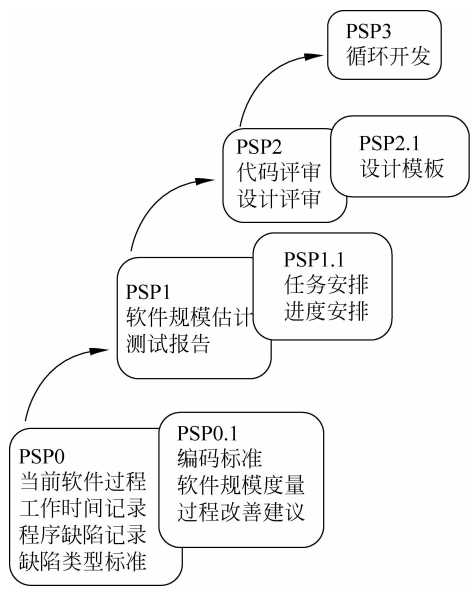

图 3-1　PSP 过程进化框架

1）个人度量过程 PSP0 和 PSP0.1

PSP0 的目的是建立个人过程基线，通过这一步，让软件工程师学会使用 PSP 的各种表格采集过程的有关数据，此时执行的是该软件开发单位的当前过程，通常包括计划、开发（包括设计、编码、编译和测试）以及后置处理 3 个阶段，并要做一些必要的试题，如测定软件开发时间，按照选定的缺陷类型标准度量引入的缺陷个数和排除的缺陷个数等，用以作为测量在 PSP 的过程中进步的基准。

PSP0.1 增加了编码标准、程序规模度量和过程改善建议等 3 个关键过程域，其中过程改善建议表格用于随时记录过程中存在的问题、解决问题的措施以及改进过程的方法，以提高软件开发人员的质量意识和过程意识。

应该强调指出，在 PSP0 阶段必须理解和学会使用表格进行规划和度量的技术和经验，以准确地满足期望的需求，其中最重要的是要保持数据的一致性、有用性和简洁性。

2）个人规划过程 PSP1 和 PSP1.1

PSP1 的重点是个人计划，引入了基于估计的计划方法，用自己的历史数据来预测新程序的大小和需要的开发时间，并使用线性回归方法计算估计参数，确定置信区间以评价预测的可信程度。PSP1.1 增加了对任务和进度的规划。

在 PSP1 阶段，软件工程师应该学会编制项目开发计划，这不仅对承担大型软件的开发十分重要，即使是开发小型软件也必不可少。因为，只有对自己的能力有客观的评价，才能做出更加准确的计划，才能实事求是地接受和完成客户（顾客）委托的任务。

3）个人质量管理过程 PSP2 和 PSP2.1

PSP2 的重点是个人质量管理，根据程序的缺陷建立检测表，按照检测表进行设计复查和代码复查（有时也称“代码走查”），以便及早发现缺陷，使修复缺陷的代价最小。随着个人经验和技术的积累，还应学会怎样改进检测表以适应自己的要求。PSP2.1 则论述设计过程和设计模板，介绍设计方法，并提供了设计模板，但 PSP 并不强调选用什么设计方法，而强调设计完备性准则和设计验证技术。

实施 PSP 的一个重要目标就是学会在开发软件的早期实际地、客观地处理由于人们的疏忽所造成的程序缺陷问题。人们都期盼获得高质量的软件，但是只有高素质的软件开发人员并遵循合适的软件过程，才能开发出高质量的软件，因此，PSP2 引入并着重强调设计复查和代码复查技术，一个合格的软件开发人员必须掌握这两项基本技术。

4）个人循环过程 PSP3

PSP3 的目标是把个人开发小程序所能达到的生产效率和生产质量延伸到大型程序。其方法是采用螺旋式上升过程，即迭代增量式开发方法，首先把大型程序分解成小的模块，然后对每个模块按照 PSP2.1 所描述的过程进行开发，最后把这些模块逐步集成为完整的软件产品。

应用 PSP3 开发大型软件系统必须采用增量式开发方法，并要求每一个增量都具有很高的质量。在这样的前提下，在新一轮开发循环中，可以采用回归测试的方法，集中力量考察新增加的这个（这些）增量是否符合要求。因此，要求在 PSP2 中进行严格的设计复查和代码复查，并在 PSP2.1 中努力遵循设计结束准则。

从对个人软件过程框架的概要描述中可以清楚地看到，如何做好项目规划和如何保证产品质量是任何软件开发过程中最基本的问题。

PSP 可以帮助软件工程师在个人的基础上运用过程的原则，借助于 PSP 提供的一些度量和分析工具，了解自己的技能水平，控制和管理自己的工作方式，使自己日常工作的评估、计划和预测更加准确、更加有效，进而改进个人的工作表现，提高个人的工作质量和产量，积极而有效地参与高级管理人员和过程人员推动的组织范围的软件工程过程改进。

3. 时间管理

1）时间管理的逻辑原理

为了制定切实可行的计划，必须对所用的时间进行跟踪。要搞清楚时间都用在什么地方，必须对时间进行跟踪，保留一份准确的记录。为了检查时间估计和计划的准确性，必须把它们写成文档并在今后与实际情况进行比较。做计划是一种技能，学习制定好的计划，第一步就是要先做计划，然后把该计划写下来，以便今后与实际数据相比较。为了管理好时间，首先制定时间分配计划，然后按照计划去做。制作计划容易，但真正实施计划是困难的。

2）了解时间的使用情况

了解时间的使用情况的方法：记录每项主要活动所花费的时间；用标准的方法记录时间；以分钟为测量单位；处理中断时间。时间数据保存的方法是用工程记事本来记录，同时做周活动总结表，并记录时间的提示。

4. 制订计划

这里介绍两种计划：阶段计划和产品计划。阶段计划是关于这段时间内对时间的安排，产品计划是关于制作产品活动期间的时间安排。

1）如何制定阶段计划

为了制定阶段计划，必须清楚时间的使用情况。根据上周事物活动总结表制订下一周的计划。一种较为精确的方法就是首先考虑下周将要做的工作内容，然后根据以前的最长和最短时间来估计出一个合适的时间。

2）如何制定产品计划

（1）收集历史项目数据。产品计划只包括对任务或作业所需时间的估计，通过收集以前不同任务所用时间的数据，就能够估计将来类似的任务大概所需要的时间。

（2）估算程序规模。产品计划的第一步是要估计产品的规模。对于程序来说，可以使用代码行测量方法估计新程序的规模。查看新程序的需求，估计各类代码有多少行，然后与以前统计的数字进行比较，可以得出开发新程序需要多少时间完成。

3）管理好时间

关于如何管好时间有以下建议值得参考：复查时间使用的分类情况；做出时间安排；找出更多可用的时间；制定基本行为规则；设定时间分配的优先级；制定执行时间安排表；收集时间数据；时间管理的目标。

可以按照如下步骤管理时间：

（1）分析自己使用时间的历史记录。

（2）制定时间安排表，决定如何使用时间。

（3）对照制定的安排表跟踪使用时间的方式。

（4）决定应该改变什么以使自己的行动达到所做安排的要求。

5. 缺陷管理

1）缺陷查找技术

研究证明，开发过程每前进一步，发现和修复缺陷的平均代价要增长 10 倍，因此要尽早发现和修复缺陷。缺陷查找技术有多种，比如：编译器能够表示出大部分语法缺陷；测试可以发现程序的错误；复查源程序清单是发现程序缺陷最有效的方法。发现程序缺陷的步骤如下：

（1）表示缺陷征兆。

（2）从征兆中推断出缺陷的位置。

（3）确定程序中的错误。

（4）决定如何修复缺陷。

（5）修复缺陷。

（6）验证这个修复是否已经解决了这个问题。

2）代码复查

代码复查就是从头到尾阅读源代码，并从中发现错误。代码复查的第一步是了解自己引入的缺陷的种类，因为在下一个程序中引入的缺陷种类一般会与前面的基本类似，只要采用同样的软件开发方法，情况会一直如此。第二步是建立标准编码实践集，以预防缺陷。第三步是建立个人代码复查检查表，并遵照其规程进行。

3）缺陷预测

开发一个新的程序时可能会觉得很难估计引入了多少缺陷，首先是经验问题。随着个人技能的不断提高，缺陷预测的准确性将会提高。试验证明，如果在代码复查方面花了足够的时间，对缺陷预测的准确性会稳定下来。一旦稳定，缺陷将容易预测。

3.1.2 团队软件过程

1. 概述

团队软件过程(Team Software Process,TSP)为开发软件产品的开发团队提供指导。TSP 的早期实践侧重于帮助开发团队改善其质量和生产率,以使其更好地满足成本及进度的目标。TSP 加上 PSP 可以帮助高绩效的工程师在一个团队中工作,来开发有质量保证的软件产品,生产安全的软件产品,改进组织中的过程管理。

TSP 被设计为满足 2～20 人规模的开发团队,大型的多团队过程的 TSP 被设计为大约最多为 150 人左右的规模。通过 TSP,一个组织能够建立起自我管理的团队来计划追踪他们的工作、建立目标,并拥有自己的过程和计划。这些团队可以是纯粹的软件开发团队,也可以是集成产品的团队,规模可以从 3 到 20 个工程师不等。TSP 使具备 PSP 的工程人员组成的团队能够学习并取得成功。如果组织中运用了 TSP,它会帮助该组织建立一套成熟规范的工程实践,确保安全可靠的软件。

2. TSP

软件过程控制是软件企业成功的关键,但过去一直缺乏一套可操作的规范来具体指导和规范项目组的开发。PSP 和 TSP 为企业提供了规范软件过程的一整套方案,从而解决了长期困扰软件开发的一系列问题,有助于企业更好地应对挑战。PSP 主要指导软件工程师个人如何更好地进行软件设计与编码,关注个人软件工程师能力的提高,从而保证个人承担的软件模块的质量,对于大型项目中的项目组如何协同工作、共同保证项目组的整体产品质量则没有给出任何指导性的原则。个人能力的提高同时需要一个有效地工作在一个团体(小组)的环境,并知晓如何一致地创造高质量的产品。为了提高团队的质量及生产能力,更加精确地达到费用、时间要求,结合 PSP 的原则提出了 TSP 以提高小组的性能,从而提供工程质量。TSP 能够指导项目组中的成员如何有效地规划和管理所面临的项目开发任务并且告诉管理人员如何指导软件开发队伍始终以最佳状态来完成工作。

1) TSP 的 4 条基本原理

(1) 应该遵循一个确定的、可重复的过程并迅速获得反馈,这样才能使学习和改革最有成效。

(2) 一个群组是否有效是由明确的目标、有效的工作环境、有能力的教练和积极的领导这 4 个方面因素的综合作用所确定的,因此应在这 4 个方面同时努力,而不能偏废其中任何一个方面。

(3) 应注意及时总结经验教训,当学员在项目中面临各种各样的实际问题并寻求有效的解决问题方案时,就会更深刻地体会到 TSP 的威力。

(4) 应注意借鉴前人和他人的经验,在可知利用的工程、科学和教学法经验的基础上来规定过程改进的指令。

2) TSP 设计的 7 条原则

在软件开发(或维护)过程中,首先需要按照 TSP 框架定义一个过程。在设计 TSP 过程时,需要按照以下 7 条原则:

(1) 循序渐进的原则。首先在 PSP 的基础上提出一个简单的过程框架,然后逐步完善。

(2) 迭代开发的原则。选用增量式迭代开发方法,通过几个循环开发一个产品。

(3) 质量优先的原则。对按 TSP 开发的软件产品,建立质量和性能的度量标准。

(4) 目标明确的原则。对实施 TSP 的群组及其成员的工作效果提供准确的度量。

(5) 定期评审的原则。在 TSP 的实施过程中,对角色和群组进行定期的评价。

(6) 过程规范的原则。对每一个项目的 TSP 规定明确的过程规范。

(7) 指令明确的原则。对实施 TSP 中可能遇到的问题提供解决问题的指南。

3) TSP 管理的 6 条原则

在实施 TSP 的过程中,应该自始至终贯彻集体管理与自我管理相结合的原则。具体地说,应该实施以下 6 项原则:

(1) 计划工作的原则。在每一阶段开始时要制订工作计划,规定明确的目标。

(2) 实事求是的原则。目标不应过高也不应过低,而应实事求是,在检查计划时如果发现未能完成或者已经超越规定的目标,应分析原因,并根据实际情况对原有计划做必要的修改。

(3) 动态监控的原则。一方面应定期追踪项目进展状态并向有关人员汇报,另一方面应经常评审自己是否按 PSP 原理进行工作。

(4) 自我管理的原则。开发小组成员若发现过程不合适,应主动、及时地进行改进,以保证始终用高质量的过程来生产高质量的软件,任何消极埋怨或坐视等待的态度都是不对的。

(5) 集体管理的原则。项目开发小组的全体成员都要积极参加和关心小组的工作规划、进展追踪和决策制订等项工作。

(6) 独立负责的原则。按 TSP 原理进行管理,每个成员都要担任一个角色。

在 TSP 的实践过程中,TSP 的创始人 Humphrey 建议在一个软件开发小组内把管理的角色分成客户界面、设计方案、实现技术、工作规划、软件过程、产品质量、工程支持以及产品测试八类。如果小组成员的数目较少,则可将其中的某些角色合并;如果小组成员的数目较多,则可将其中的某些角色拆分。总之,每个成员都要独立担当一个角色。

4) 开发小组素质的基本度量元

软件开发小组按 TSP 进行生产、维护软件或提供服务,其质量可用两组元素来表达。一组元素用以度量开发小组的素质,称之为开发小组素质度量元;另一组元素用以度量软件过程的质量,称之为软件过程质量度量元。

(1) 开发小组素质的基本度量元有以下 5 项:

① 所编文档的页数。

② 所编代码的行数。

③ 花费在各个开发阶段或花费在各个开发任务上的时间(以分钟为度量单位)。

④ 在各个开发阶段中注入和改正的缺陷数目。

⑤ 在各个阶段对最终产品增加的价值。

应该指出,这 5 个度量元是针对软件产品的开发来陈述的,对软件产品的维护或提供其他服务可以参照这些条款给出类似的陈述。

(2) 软件过程质量的基本度量元有以下 5 项：

① 设计工作量应大于编码工作量。

② 设计评审工作量应占一半以上的设计工作量。

③ 代码评审工作量应占一半以上的代码编制的工作量。

④ 每千行源程序在编译阶段发现的差错不应超过 10 个。

⑤ 每千行源程序在测试阶段发现的差错不应超过 5 个。

无论是开发小组的素质，还是软件过程的质量，都可用一个等五边形来表示，其中每一个基本度量元是该等五边形的一个顶。基本度量元的实际度量结果落在其顶点与等五边形中心的连线上，其取值可以根据事先给出的定义来确定。在应用 TSP 时，通过对必要数据的收集，项目组在进入集成和系统测试之前能够初步确定模块的质量。如果发现某些模块的质量较差，就应对该模块进行精心的复测，有时甚至有必要对质量特别差的模块重新进行开发，以保证生产出高质量的产品，且能节省大量的测试和维护时间。

3. TSP 的具体操作

TSP 由一系列阶段和活动组成，各阶段均由计划会议发起。在首次计划中，TSP 组将制订项目整体规划和下阶段详细计划，TSP 组员在详细计划的指导下跟踪计划中各种活动的执行情况。首次计划后，原定的下阶段计划会在周期性的计划制订中不断得到更新。通常无法制订超过 3～4 个月的详细计划。所以，TSP 根据项目情况，每 3～4 个月为一阶段，并在各阶段进行重建。无论何时，只要计划不再适应工作就进行更新。当工作中发生重大变故或成员关系调整时，计划也将得到更新。在计划的制订和修正中，小组将定义项目的生命周期和开发策略，这有助于更好地把握整个项目开发的阶段、活动及产品情况。每项活动都用一系列明确的步骤、精确的测量方法及开始、结束标志加以定义。在设计时将制订完成活动所需的计划、估计产品的规模、各项活动的耗时、可能的缺陷率及去除率，并通过活动的完成情况重新修正进度数据。开发策略用于确保 TSP 的规则得到自始至终的维护。

3.1.3 软件项目组

1. 概述

项目组织机构是项目型组织，是指那些一切工作都围绕项目进行、通过项目创造价值并达成自身战略目标的组织。

项目组是指为了完成某个特定的任务而把一群不同背景、不同技能和来自不同部门的人组织在一起的组织形式。其特点是根据任务的需要，将各种人才集合在一起进行联合攻关，任务完成后，小组基本就可以解散了，项目组人员不固定。其优点是适应性强，机动灵活，容易接受新观念、新方法。其缺点是缺乏稳定性；在规模上有很大的局限性。

项目组织是保证工程项目正常实施的组织保证体系，就项目这种一次性任务而言，项目组织建设包括从组织设计、组织运行、组织更新到组织终结这样一个生命周期。项目管理要在有限的时间、空间和预算范围内将大量物资、设备和人力组织在一起，按计划实施项目目标，必须建立合理的项目组织。

项目组织的特征：组织目标单一，工作内容庞杂；项目组织是一个临时性机构；项目组

织应精干高效；项目经理是项目组织的关键。

项目组织的设置原则：有效幅度管理；权责对等；才职相称；命令统一；效果与效率；适时重组。

2. 项目组织机构的类型

1）垂直团队组织

垂直团队组织由多面手组成。用例分配给了个人或小组，然后由他们从头至尾地实现用例。

垂直团队组织的优点：以单个用例为基础实现平滑的端到端开发；开发人员能够掌握更广泛的技能。

垂直团队组织的缺点：多面手通常是一些要价很高并且很难找到的顾问；多面手通常不具备快速解决具体问题所需的特定技术专长；主题专家可能不得不和若干开发人员小组一起工作，从而增加了他们的负担；所有多面手水平各不相同。

垂直团队组织的成功因素：每个成员都按照一套共同的标准与准则工作；开发人员之间需要进行良好的沟通，以避免公共功能由不同的组来实现；公共和达成共识的体系结构需要尽早地在项目中确立。垂直团队组织结构示意图如图 3-2 所示。

2）水平团队组织

水平团队组织由专家组成。此类团队同时处理多个用例，每个成员都从事用例中有关其自身的方面。

水平团队组织的优点：能高质量地完成项目各个方面（需求、设计等）的工作；一些外部小组，如用户或操作人员，只需要与了解他们确切要求的一小部分专家进行交互。

水平团队组织的缺点：专家们通常无法意识到其他专业的重要性，导致项目的各方面之间缺乏联系；“后端”人员所需的信息可能无法由“前端”人员来收集；由于专家们的优先权、看法和需求互不相同，所以项目管理更为困难。

水平团队组织的成功因素：团队成员之间需要有良好的沟通，这样他们才能彼此了解各自的职责；需要制定专家们必须遵循的工作流程和质量标准，从而提高移交给其他专家的效率。水平团队组织结构示意图如图 3-3 所示。

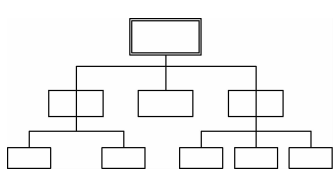

图 3-2　垂直团队组织结构示意图

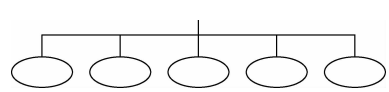

图 3-3　水平团队组织结构示意图

3）混合团队组织

混合团队组织由专家和多面手共同组成。多面手继续操作一个用例的整个开发过程，支持并处理多个使用例中各部分的专家们一起的工作。

混合团队组织的优点：拥有前两种方案的优点；外部小组只需要与一小部分专家进行交互；专家们可集中精力从事他们所擅长的工作；各个用例的实现都保持一致。

混合团队组织的缺点：拥有前两种方案的缺点；多面手仍然很难找到；专家们仍然不

能认识到其他专家的工作并且无法很好地协作，尽管这应该由多面手来调节；项目管理仍然很困难。

混合团队组织的成功因素：项目团队成员需要良好的沟通；需要确定公共体系结构；必须适当地定义公共流程、标准和准则。

如何组织项目团队？是采用垂直方案、水平方案还是混合方案？以垂直方案组织的团队由多面手组成，每个成员都充当多重角色；以水平方案组织的团队由专家组成，每个成员充当一、两个角色；以混合方案组织的团队既包括多面手，又包括专家。一个重要的考虑因素是可供选择的人员的性质。如果大多数人员是多面手，则往往需要采用垂直方案；如果大多数人员是专家，则采用水平方案；既有多面手又有专家，就采用混合方案。

3. 如何组织软件开发团队

如何构建软件开发团队取决于可供选择的人员、项目的需求以及组织的需求。有效的软件项目团队由担当各种角色的人员所组成。每位成员扮演一个或多个角色；可能一个人专门负责项目管理，而另一些人则积极地参与系统的设计与实现。常见的一些项目角色包括分析师、策划师、数据库管理员、设计师、操作/支持工程师、程序员、项目经理、项目赞助者、质量保证工程师、需求分析师、主题专家(用户)以及测试人员。

3.1.4 微软软件开发团队

微软的开发团队被公认是最成功的开发团队之一。微软的团队模型描述了微软的一个产品开发团队的组成及其内部人员的分工和职责等情况。

1. 微软的产品团队的原则

该原则包括：小型并具有多功能的团队；角色互相依赖并分担责任；具有深厚的技术和商业敏锐性；关注于完成产品；有明确的工作目标；用户积极参与；共享产品远景；人人参与设计；努力从过去的产品中学习；共享产品总体管理和决策；所有团队成员都在同一个地方工作；大团队的工作方式类似于小团队。

在微软的产品团队中，权威仅仅来自于知识，而不是来自于职位。

2. 各团队的角色及主要目标

微软的产品团队由一些地位平等的小团队组成，这些小团队在整个产品团队中扮演着互相依赖、互相合作的不同角色，每一个角色都有自己特定的任务。他们共享对产品的管理，也共享对产品的责任。每一个角色都始终存在并作用于整个产品的开发过程。

(1) 产品管理团队(Product Management Team)。产品管理部门负责人，即为产品经理(General Manager)。产品总经理下面有一个或几个产品单元经理(Product Unit Manager)。同时产品管理团队中还包含产品计划、市场分析、产品推销和公共关系的负责人。

(2) 项目管理团队(Program Management Team)。项目管理的任务是控制决策的各种因素，以保证在适合的时候推出合适的产品，同时负责创建功能规定文档，并将它作为如何实施产品或服务的一种决策工具。最后，项目管理团队将面对使产品或服务与组织标准和

操作目标相一致的日常协调工作。

(3) 软件开发团队。开发人员的任务比较单一，主要就是负责代码的设计和程序的实现。

(4) 软件测试团队。软件测试团队的任务很清楚，就是站在使用者和攻击者的角度上，通过不断地使用刚开发出来的软件产品，尽量多地找出产品中存在的问题。

3.2　项目进度控制

本节重点介绍项目进度概述、进度控制的4个过程以及如何实施进度控制。

3.2.1　项目进度概述

1. 概念

1) 项目进度计划

项目进度计划(plan)是指对一个工程项目按一定的方式进行分解，并对分解后的工作单元(activity)规定相互之间的顺序关系以及工期(duration)。

2) 进度

进度(schedule)是指作业在时间上的排列，强调的是作业进展(progress)以及对作业的协调和控制(coordination & control)，在规定工期内完成规定任务的情况。

3) 工期

工期是由从开始到竣工的一系列施工活动所需的时间构成的。工期目标包括总进度计划实现的总工期目标、各分进度计划(采购、设计、施工等)或子项进度计划实现的工期目标、各阶段进度计划实现的里程碑目标。通过计划进度目标与实际进度完成目标值的比较，找出偏差及其原因，采取措施调整、纠正，从而实现对项目进度的控制。

4) 进度控制

进度控制是指在限定的工期内，以事先拟定的合理且经济的工程进度计划为依据，对整个建设过程进行监督、检查、指导和纠正的行为过程。进度控制是反复循环的过程，体现运用进度控制系统控制工程建设进展的动态过程。进度控制在某一界限范围内(最低费用相对应的最优工期)对加快施工进度能达到使费用降低的目的。而超越这一界限，施工进度的加快反而将会导致投入费用的增大。因此，对建设项目进行三大目标(质量、投资、进度)控制的实施过程中应互相兼顾，单纯地追求某一目标的实现均会适得其反。因而对建设项目进度计划目标实施的全面控制，是投资目标和质量目标实施的根本保证，也是履行工程承包合同的重要工作内容。

2. 进度控制全过程

在工程项目进度计划的实施中，控制循环过程包括：

(1) 执行计划的事前进度控制，体现对计划、规划和执行进行预测的作用。

(2) 执行计划的过程进度控制，体现对进度计划执行的控制作用，以及在执行中及时采取措施纠正偏差的能力。

(3) 执行计划的事后进度控制,体现对进度控制每一循环过程总结整理的作用和调整计划的能力。

建设项目实施全过程的三项控制各有各的实用环境、控制工作内容和时间。能实现对施工进度事先进行全面控制最好,但是,工程进度计划的编制者很难事先对项目的实施过程可能出现的问题进行全面估计。因此,进度控制工作大量的是在过程控制和事后控制中完成。

3. 进度控制的措施

进度控制是一项全面的、复杂的、综合性的工作,原因是工程实施的各个环节都影响工程进度计划。因此要从各方面采取措施,促进进度控制工作。采用系统工程管理方法,编制网络计划只是第一道工序,最关键的是如何按时间主线进行控制,保证计划的实现。为此,采取进度控制的措施包括:

(1) 加强组织管理。网络计划在时间安排上是紧凑的,要求参加施工的不同管理部门及管理人员协调配合、努力工作。因此,应从全局出发合理组织,统一安排劳力、材料、设备等,在组织上使网络计划成为人人必须遵守的技术文件,为网络计划的实施创造条件。

(2) 为保证总体目标实现,对工期应着重强调工程项目各分级网络计划控制。严格界定责任,依照管理责任层层制定总体目标、阶段目标、节点目标的综合控制措施,全方位地寻找技术与组织、目标与资源、时间与效果的最佳结合点。

(3) 网络计划的实施效果应与经济责任制挂钩。把网络计划内容、节点时间的要求具体落实,实行逐级负责制,使对实际网络计划目标的执行有责任感和积极性。同时规定网络计划实施效果的考核评定指标,使各分部、分项工程完成日期、形象进度要求、质量、安全、文明施工均达到规定要求。

(4) 网络计划的编制修改和调整应充分利用计算机,以利于网络计划在执行过程中的动态管理。

3.2.2 进度控制过程

1. 进度控制过程的 4 个阶段

进度控制的 4 个步骤(PDCA)是计划(Plan)、执行(Do)、检查(Check)、行动(Action)。进度控制过程是一个周期性的循环过程。进度控制过程的 4 个阶段如图 3-4 所示,分别是:

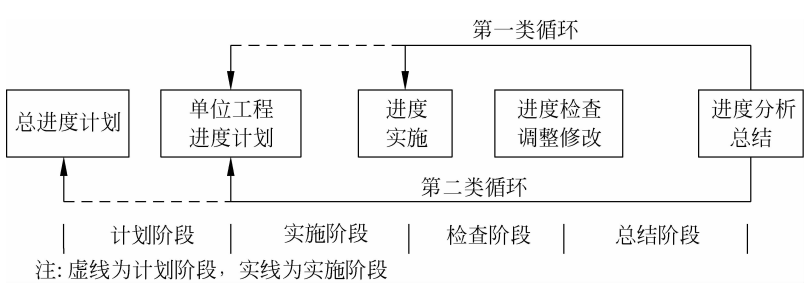

图 3-4 进度控制过程的 4 个阶段

(1) 计划阶段(编制进度计划)。

(2) 实施阶段(实施进度计划)。

(3) 检查阶段(检查与调整进度计划)。

(4) 总结阶段(分析与总结进度计划)。

2. 进度计划的编制

进度计划是表示各项工程的实施顺序、开始和结束时间以及相互衔接关系的计划。进度计划是现场实施管理的核心指导文件,是进度控制的依据和工具。进度计划是按工程对象编制的,重点是安排工程实施的连续性。

1) 进度计划编制的目的

具体目的包括:

(1) 保证按时获利以补偿已经发生的费用支出。

(2) 协调资源。

(3) 使资源需要时可以利用。

(4) 预测在不同时间上所需资金和资源的级别以便赋予项目不同的优先级。

(5) 保证项目正常完成。

2) 进度计划编制的要求

具体要求包括:

(1) 保证项目在合同规定的时间内完成,实现项目目标要求。

(2) 实施进度安排必须满足连续性和均衡性要求。

(3) 实施顺序的安排应进行优化,以便提高经济效益。

(4) 应选择适当的计划图形,满足使用进度计划的要求。

(5) 讲究编制程序,提高进度计划的编制质量。

3) 进度计划编制的原则

具体原则包括:

(1) 应对所有大事及其期限要求进行说明。

(2) 确切的工作程序能够通过工作网络得以说明。

(3) 进度应该与工作分解结构(WBS)有直接关系。采用 WBS 中的系统数字来说明工作进度,应该表明项目开始和结束时间;全部进度必须体现时间的紧迫性。

可能的话应详细说明每件大事需要配置的资源;项目越复杂、专业分工就越细,就更需要综合管理,需要一个主体的、协调的工作进度计划。

4) 进度计划的内容

具体内容包括:

(1) 项目综合进度计划。

(2) 设备(材料)采购工作进度计划。

(3) 项目实施(开发)进度计划。

(4) 项目验收和投入使用进度计划。

3. 进度计划的实施

1）做好准备工作

具体工作包括：

（1）将进度计划具体化为实施作业计划和实施任务书。

（2）分析计划执行中可能遇到的阻力、计划执行的重点和难点，提出保证计划成功实施的措施。

（3）将计划交给执行者。交底可以开会进行，也可结合下达实施任务书进行。管理者和作业者均应提出计划实现的技术和组织措施。

2）做好实施记录

具体实施记录包括：

（1）在计划实施过程中，应进行跟踪记录，以便为检查计划、分析实施状况、计划执行状况、调整计划、总结等提供原始资料。

（2）记录工作最好在计划图表上进行，以便检查计划时分析和对比。

（3）记录必须实事求是，不得造假。

（4）流水计划：在计划流水之下绘制实际进度线条。

（5）网络计划：记录实际持续时间。

（6）在计划图上用彩色标明已完成部分。

（7）用切割线记录。

3）做好调度工作

调度工作的任务是掌握计划实施情况，协调关系，排除矛盾，克服薄弱环节，保证作业计划和进度控制目标的实现。

调度工作的内容：

（1）检查计划执行中的问题，找出原因，提出解决措施。

（2）督促供应商按进度计划要求供应资源。

（3）控制施工现场临时设施正常使用，搞好平面管理，发布调度令，检查决议执行情况。

（4）调度工作应以作业计划和现场实际需要为依据，加强预测，信息灵通，及时、准确、灵活、果断，确保工作效率。

（5）在接受监理的工程中，调度工作应与监理单位的协调工作密切结合，调度会应请监理人员参与，监理协调会应视为调度会的一种形式。

4. 进度计划的检查与调整

1）进度计划的检查

（1）检查时间分类：日常检查、定期检查。

（2）检查内容：进度计划中的开始时间、完成时间、持续时间、逻辑关系、实物工程量和工作量、关键线路、总工期，时差利用等。

（3）检查方法：对比法，即计划内容和记录的实际状况进行对比。

（4）检查的结果应写入进度报告，承建单位的进度报告应提交给监理工作师，作为其进度控制、核发进度款的依据。

2）进度计划的调整

（1）通过检查分析，若进度偏离计划不严重，可以通过协调矛盾、解决障碍来继续执行原进度计划。

（2）当项目确实不能按原计划实现时，则应对计划进行必要的调整，适当延长工期或改进实施速度。

（3）新的调整计划应作为进度控制的新依据。

5. 进度计划的分析与总结

1）进度计划的分析与总结

其目的是为了发现问题、总结经验、寻找更好的控制措施，进一步提高控制水平；通过定量分析和定性分析，归纳出卓有成效的控制及原因，为以后的进度控制提供借鉴。

2）项目进度控制的数据收集

（1）实际数据：采集内容包括活动的开始和结束的实际时间、实际投入的人力、使用或投入的实际成本、影响进度的重要原因及分析、进度管理情况。

（2）有关项目范围、进度计划和预算变更的信息：变更可能由建设单位或承建单位引起，也可能是某种不可预见事情的发生引起；一旦变更被列入计划并取得建设单位同意，就必须建立一个新的基准计划，整个计划的范围、进度和预算可能比最初的基准计划不同。

3.2.3　如何实施进度控制

1. 进度控制的目标与范围

1）进度控制的意义

（1）有利于尽快发挥投资效益。

（2）有利于维护良好的管理秩序。

（3）有利于提高企业经济效益。

（4）有利于降低信息系统工程的投资风险。

2）进度控制的目标

（1）总目标：通过各种有效措施保障工程项目在规定的时间内完成，即信息系统达到竣工验收、试运行及投入使用的计划时间。

（2）总目标分解：按单项工程分解；按专业分解；按工程阶段分解；按年、季、月分解。

3）进度控制的范围

（1）纵向范围：在工程建设的各个阶段，对项目建设的全过程控制。

（2）横向范围：在工程建设的各个组成部分，对分项目、子系统的控制。

4）影响进度控制的因素

影响进度控制的因素主要包括：

（1）工程质量的影响。

（2）设计变更的影响。

（3）资源投入的影响。

（4）资金的影响。

(5) 相关单位的影响。

(6) 可见或不可见的风险因素的影响。

(7) 承建单位管理水平的影响。

2. 进度控制的任务、程序与措施

(1) 在准备阶段,项目经理应该参与招标前的准备工作,编制本项目的工作计划,内容包括项目主要内容、组织管理、实施阶段计划、实施进程等;分析项目的内容及项目周期,提出安排工程进度的合理建议;对建设合同中所涉及的产品和服务的供应周期做出详细说明,并建议建设单位做出合理安排;对工程实施计划及其保障措施提出建议,并在招标书中明确;在评标时,应对项目进度安排及进度控制措施进行审查,提出审核意见。

(2) 在设计阶段,根据工程总工期要求,确定合理的设计时限要求;根据设计阶段性输出,由粗而细地制定项目进度计划;协调各方进行整体性设计;提供设计所需的基础资料和数据;协调有关部门保证设计工作顺利进行。

(3) 在实施阶段,根据工程招标和施工准备阶段的工程信息,进一步完善项目进度计划,并据此进行实施阶段进度控制;审查施工进度计划,确认其可行性并满足项目控制进度计划要求;审查进度控制报告,对施工进度进行跟踪,掌握施工动态;在施工过程中,做好对人力、物力、资金的投入控制工作及转换工作,做好信息反馈、对比和纠正工作,使进度控制定期、连续地进行;开好进度协调会,及时协调各方关系,使工程施工顺利进行;及时处理工程延期问题。

(4) 在验收阶段,验收项目并提交验收报告。

3. 进度控制方法

1) 甘特图

甘特图(Gantt chart)又叫横道图、条状图(Bar chart)。甘特图思想比较简单,即以图示的方式通过活动列表和时间刻度形象地表示出任何特定项目的活动顺序与持续时间。甘特图基本上是一条线条图,横轴表示时间,纵轴表示活动(项目),线条表示在整个期间上计划和实际的活动完成情况。它直观地表明任务计划在什么时候进行,及实际进展与计划要求的对比。管理者由此可便利地弄清一项任务(项目)还剩下哪些工作要做,并可评估工作进度,如图 3-5 所示。

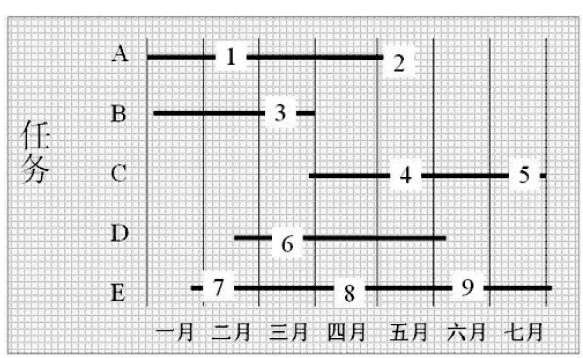

图 3-5 甘特图

2）工程进度曲线（“香蕉”型曲线图）

“香蕉”型曲线是两条S型曲线组合成的闭合曲线。从S型曲线比较法中得知，按某一时间开始的施工项目的进度计划，其计划实施过程中进行时间与累计完成任务量的关系都可以用一条S型曲线表示。对于一个施工项目的网络计划，在理论上总是分为最早和最迟两种开始与完成时间的。因此，一般情况下，任何一个施工项目的网络计划都可以绘制出两条曲线：一条是计划以各项工作的最早开始时间安排进度而绘制的S型曲线，称为ES曲线；另一条是计划以各项工作的最迟开始时间安排进度而绘制的S型曲线，称为LS曲线。两条S型曲线都是从计划的开始时刻开始和完成时刻结束，因此两条曲线是闭合的。一般情况下，其余时刻ES曲线上的各点均落在LS曲线相应点的左侧，形成一个形如香蕉的曲线，故此称为“香蕉”型曲线，如图3-5所示。在项目的实施中，进度控制的理想状况是任一时刻按实际进度描绘的点应落在该“香蕉”型曲线的区域内，如图3-6中所示的R曲线。

“香蕉”型曲线比较法的作用：利用“香蕉”型曲线进行进度的合理安排；进行施工实际进度与计划进度比较；确定在检查状态下，后期工程的ES曲线和LS曲线的发展趋势。

3）网络图计划法

（1）单代号网络图。

用一个圆圈代表一项活动，并将活动名称写在圆圈中。箭线符号仅用来表示相关活动之间的顺序，因其活动只用一个符号就可代表，故称为单代号网络图，如图3-7所示。

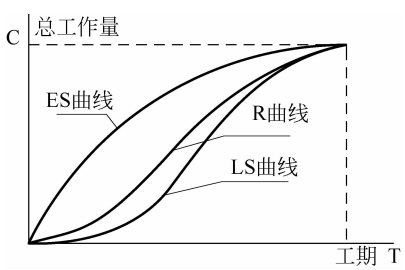

图3-6　“香蕉”曲线图

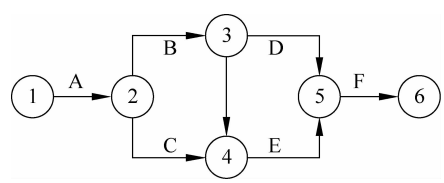

图3-7　单代号网络图

（2）双代号网络图。

双代号网络图是应用较为普遍的一种网络计划形式。它是以箭线及其两端节点的编号表示工作的网络图。双代号网络图中，每一条箭线应表示一项工作。箭线的箭尾节点表示该工作的开始，箭线的箭头节点表示该工作的结束，如图3-8所示。

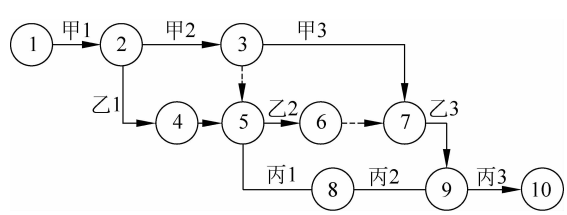

图3-8　双代号网络图

箭线：在双代号网络图中，工作一般使用箭线表示，任意一条箭线都需要占用时间、消耗资源，工作名称写在箭线的上方，而消耗的时间则写在箭线的下方。

虚箭线：是实际工作中不存在的一项虚设工作，因此一般不占用资源、不消耗时间。虚箭线一般用于正确表达工作之间的逻辑关系。

节点：反映的是前后工作的交接点，节点中的编号可以任意编写，但应保证后续工作的节点比前面节点的编号大，且不得有重复。

起始节点：即第一个节点，它只有外向箭线（即箭头离向节点）。

终点节点：即最后一个节点，它只有内向箭线（即箭头指向节点）。

中间节点：既有内向箭线又有外向箭线的节点。

线路：网络图中从起始节点开始，沿箭头方向通过一系列箭线与节点最后达到终点节点的通路，称为线路。一个网络图中一般有多条线路，线路可以用节点的代号来表示。

3.3 项目成本估算与控制

本节重点介绍成本估算、工作量估算和成本控制。

3.3.1 成本估算

1. 成本估算的基本概念

成本估算是项目成本管理的核心，通过成本估算，分析并确定项目的估算成本，并以此为基础进行项目成本预算、开展项目成本控制等管理活动。

软件项目的成本估算是成本管理的核心，是预测开发一个软件系统所需要的总工作量的过程。成本估算贯穿于软件的生命周期。

2. 软件项目成本估算的编制方法

软件开发项目中常用的成本估算方法有自上向下估算法、自下而上估算法、混合估算法、参数估算法和组合估算法。

1）自上向下估算法

自上向下估算法又称为类比估算法，是一种自上而下的估算形式。它使用以前的、相似项目的实际成本作为目前项目成本估算的根据，这是一种专家判断法。项目经理利用以前类似的项目实际成本作为基本依据，通过经验做出判断项目整体成本和各个子任务的成本预算，此方法通常在项目的初期或信息不足时进行。方法较其他方法更节省，但不是很精确，需要项目经理具有较高的水平和经验。

2）自下而上估算法

它包括估算个人工作项和汇总单个工作项成整体项目，单个工作项的大小和估算人员的经验决定估算的精度。如果一个项目有详细工作分解结构，项目经理能够让每个人负责一个工作包，并让他们为那个工作包建立自己的成本估算；然后将所有的估算加起来，产生更高一级的估算，并且最终完成整个项目的估算。

3）混合估算法

混合估算法就是将上述两种估算方法综合使用。在科学计算的基础上，结合项目管理者的经验，做出既科学又符合实际情况的预算。并且可以在科学计算过程中加进参数模型，

通过数据的积累，根据同类项目的管理状况和成本数据建立模型，在遇到同类项目时可直接套用。

4）参数估算法

参数估算法是一种使用项目特性参数建立数据模型来估算成本的方法，是一种统计技术，如回归分析和学习曲线。数学模型可以简单也可以复杂，有的是简单的线性关系模型，有的模型就比较复杂。一般参考历史信息，重要参数必须量化处理，根据实际情况对参数模型按适当比例调整。每个任务必须至少有一个统一的规模单位。

5）组合估算法

这是目前企业软件开发过程中常用的软件成本估算方式，它是一种自下而上估算法和参数估算法的结合模型。其步骤如下：

（1）对任务进行分解。

（2）计算每个任务的估算值 E_i。

（3）计算直接成本：直接成本＝$E_1+E_2\cdots+E_i+\cdots E_n$。

（4）计算估算成本：估算成本＝直接成本＋间接成本。间接成本是指直接成本以外的成本，如安装、培训、预防性维护、备份与恢复的费用，以及运行系统相关的劳务和材料费、管理费、相关补助费用等。资源外包的项目，这些资源包括人员和设备的外包等。

（5）计算总成本：总成本＝估算成本＋风险基金＋税收。其中，风险基金＝估算成本×a%（一般情况 a 为 10～20 左右），税收＝估算成本×b%（一般情况 b 为 5 左右）。

3.3.2　工作量估算

1. 代码行分析法

1）概念和计算方法

代码行（Line Of Code，LOC）分析法是对软件产品的源代码的行数进行测量。但是也有一些问题需要思考：是计算物理行数，还是程序的命令数量？空行是否计算？注释是否计算？预定义文件是否计算？不同版本如何计算？开发过程中的配置脚本、编译脚本是否计算？共享文件（例如共享的开发库文件中的头部文件）如何计算？

一般来说是计算物理行数，不计算空行，不计算注释。对于其他选项，一般计算源文件根目录下的所有文件。所以代码行指的是所有的可执行的源代码行数，包括可交付的工作控制语言（Job Control Language）语句、数据定义、数据类型声明、等价声明、输入/输出格式声明等。常使用的单位有 SLOC（Single Line Of Code）、KLOC（Thousand Lines Of Code）、LLOC（Logical Line Of Code）、PLOC（Physical Line Of Code）、NCLOC（Non-Commented Line Of Code）、DSI（Delivered Source Instruction），其中 SLOC 和 KLOC 比较常用。

2）其他注意点

代码行分析法在某些程序上反映了软件的规模，并且是物理上可测量的。但是在需求、计划、设计阶段因为本身没有代码行，需要其他方法解决。

近来可视化编程工具的大量采用，以及模板库、类库的广泛采用，在程序的结果中有大量自动生成的代码或者复杂的自动配置脚本或资源文件设置，在采用这些工具的项目中，用代码行分析法得到数值的意义已经大大降低。

尽管代码行分析法有很多缺点，但由于其容易使用、操作成本低，还是值得推荐的一种参考和补充手段。

2. 功能点分析法

1) 概述

功能点分析法(Function Point Analysis，FPA)是一种相对抽象的方法，是一种人为设计出的度量方式，主要解决如何客观、公正、可重复地对软件规模进行度量。FPA 在 20 世纪 70 年代由 IBM 的工程师 Allan Albrech 提出，随后被国际功能点用户协会(the International Function Point Users' Group，IFPUG)提出的 IFPUG 方法采用，从系统的复杂性和系统的特性这两个角度来度量系统的规模。功能点可以用于需求文档、设计文档、源代码和测试用例度量，根据具体方法和编程语言的不同，功能点可以转换为代码行。ISO 组织已经把多种功能点估算方法成为国际标准，如加拿大人 Alain Abran 等人提出的全面功能点法(Full Function Points)、英国软件度量协会(United Kingdom Software Metrics Association，UKSMA)提出的 IFPUG 功能点法(IFPUG Function Points)、英国软件度量协会提出的 Mark Ⅱ FPA 功能点法(Mark Ⅱ Function Points)、荷兰功能点用户协会(Netherlands Function Point Users Group，NEFPUG)提出的 NESMA 功能点法以及软件度量共同协会(the Common Software Metrics Consortium，COSMIC)提出的 COSMIC-FFP 方法，这些方法都属于 Albrech 功能点方法的发展和细化。

功能点分析法有一些相对完整的、自成体系的概念，主要包括基础功能部件(Base Function Component，BFC)、BFC 类型、边界、用户、本地化、功能领域、功能规模、功能点规模测量的范围、功能点规模测量过程、功能点规模测量方法、功能性需求、质量需求、技术性需求、数值调整以及调整因子等 15 个关键概念。

2) 功能点计算

项目的功能点数是几个测量参数(用户输入数、用户输出数、用户查询数、文件数、外部接口数)的功能点之和。

(1) 用户输入数：计算每个用户输入，它们向软件提供面向应用的数据。输入应该与查询区分开来，分别计算。

(2) 用户输出数：计算每个用户输出，它们向软件提供面向应用的信息。这里的输出是指报表、屏幕、出错信息等。一个报表中的单个数据项不单独计算。

(3) 用户查询数：一个查询被定义为一次联机输入，它导致软件以联机输出的方式产生实时的响应。每一个不同的查询都要计算。

(4) 文件数：计算每个逻辑的主文件(如数据的一个逻辑组合，它可能是某个大型数据库的一部分或是一个独立的文件)。

(5) 外部接口数：计算所有机器可读的接口(如磁带或磁盘上的数据文件)，利用这些接口可以将信息从一个系统传送到另一个系统。

3) 成本估算公式

(1) 每个测量参数的估算 FP 计数＝估算值×加权因子。

(2) 项目估算 FP＝各参数 FP 计数之和×复杂度调整因子。

(3) 估算工作量＝项目估算 FP÷估算的生产率。

(4) 估算总成本＝日薪×估算工作量。

(5) 单个 FP 估算成本＝估算总成本÷估算 FP。

其中,估算的生产率可以由经验获得。

3. 任务分解法(WBS)

1) 任务分解的目的

(1) 防止遗漏项目的可交付成果。

(2) 帮助项目经理关注项目目标和澄清职责。

(3) 建立可视化的项目可交付成果,以便估算工作量和分配工作。

(4) 帮助改进时间、成本和资源估计的准确度。

(5) 帮助项目团队的建立和获得项目人员的承诺。

(6) 为绩效测量和项目控制定义一个基准。

(7) 辅助沟通清晰的工作责任。

(8) 为其他项目计划的制定建立框架。

(9) 帮助分析项目的最初风险。

2) 分解的原则

(1) 横向分解:指任务分解不能出现漏项,也不能包含不在项目范围之内的任何产品或活动。也就是既覆盖所有的需要,又不做多余的工作。

(2) 纵向分解:指任务分解要足够细,以满足任务分配、检测及控制的目的,也就是细化到不能再细的程度。

3) 分解的方法

(1) 自上而下与自下而上的充分沟通。

(2) 一对一个别交流。

(3) 小组讨论。

4) 分解的标准

(1) 分解后的活动结构清晰。

(2) 逻辑上形成一个大的活动。

(3) 集成所有的关键因素。

(4) 包含临时的里程碑和监控点。

(5) 所有活动全部定义清楚。

(6) 学会分解任务。只有将任务分解得足够细,才能心里有数,才能有条不紊地工作,才能统筹安排时间表。

5) 工作分解结构

以可交付成果为导向对项目要素进行的分解归纳和定义了项目的整个工作范围每下降一层代表对项目工作的更详细定义。在项目管理实践中,工作分解是最重要的内容,是计划过程的中心,也是制定进度计划、资源需求、成本预算、风险管理计划和采购计划等的重要基础,同时也是控制项目变更的重要基础。

6) WBS 的功能

(1) WBS 是一个描述思路的规划和设计工具,它帮助项目经理和项目团队确定和有效

地管理项目的工作。

(2) WBS是一个清晰地表示各项目工作之间的相互联系的结构设计工具。

(3) WBS是一个展现项目全貌、详细说明为完成项目所必须完成的各项工作的计划工具。

(4) WBS定义了里程碑事件,可以向高级管理层和客户报告项目完成情况,作为项目状况的报告工具。

4. 类比估算法

类比估算法(Analogous Estimates)是最简单的成本估算技术,实质上是一种专家判断法。类比估算,顾名思义是通过同以往类似项目相类比得出估算,为了使这种方法更为可靠和实用,进行类比的以往项目不仅在形式上要和新项目相似,而且在实质上也要非常相同。

这种方法简单易行,花费较少,尤其当项目的资料难以取得时,此方法是估算项目总成本的一种行之有效的方法。当然,它也有一定的局限性,进行成本估算的上层管理者根据他们对以往类似项目的经验对当前项目总成本进行估算,但是又有项目的一次性、独特性等特点,在实际生产中,根本不可能存在完全相同的两个项目,因此这种估算的准确性较差。

用这种方法进行整体估算时比较准确,可以避免过分重视一些任务而忽视另外一些任务。但是可能出现下层人员认为分到的估算不足以完成任务却保持沉默的情况。

类比估算法的操作步骤:首先项目的上层管理人员收集以往类似项目的有关历史资料,依据自己的经验和判断估算当前项目的总成本和各分项目的成本;然后将估算结果传递给下一层管理人员,并责成他们对组成项目和子项目的任务和子任务的成本进行估算,并继续向下传送其结果,直到项目组的最基层人员。

5. PERT时间估计法

PERT网络图起源于其活动时间不确定的项目。解决这一问题要求对每个活动做出3种时间估计。

(1) 最可能的活动时间:正常情况下,完成某项工作的时间(T_m)。

(2) 乐观的活动时间:也就是要进行这个活动所需的最短时间(T_o)。

(3) 悲观的活动时间:也就是要进行这个活动所需的最长时间(T_p)。

活动期望时间(T_e)的计算公式:

$$T_e=\frac{T_o+4\times T_m+T_p}{6}$$

例 3-1 完成某项工作最可能的活动时间 $T_m=16$ 天,乐观的活动时间 $T_o=10$ 天,悲观的活动时间 $T_p=40$ 天,那么活动的期望时间为 $T_e=\frac{10+4\times16+40}{6}=19$(天)。

例 3-2 一个包含3个活动的某种路径的3种时间估计,如表3-1所示。

表 3-1 3个活动的3种时间估计

	第一段	第二段	第三段
T_o	4	1	2
T_m	7	7	11
T_p	16	25	26

期望时间为：

$$T_e = \frac{4+4\times7+16}{6} + \frac{1+4\times7+25}{6} + \frac{2+4\times11+26}{6} = 8+9+12 = 29(\text{天})$$

标准差为：

$$\delta = \left[\left(\frac{16-4}{6}\right)^2 + \left(\frac{25-1}{6}\right)^2 + \left(\frac{26-2}{6}\right)^2\right]^{\frac{1}{2}} = (36)^{\frac{1}{2}} = 6(\text{天})$$

这个案例计算结果显示的期望时间为 29 天，标准差为 6 天，因此项目将在第 23 天和第 35 天之间完成。

任务的领导者、项目经理以及另外 1～3 名团队成员应该对任务时间估计进行讨论。请任务领导者参加讨论的原因是其决定权；项目经理的参与是为了在其他项目时间估计之间进行权衡；其他人则能够带来专业经验与知识。

6. Putnam 模型

1978 年提出的 Putnam 模型是一种动态多变量模型。它假定在软件开发的整个生命期中工作量有特定的分布。这种模型是依据在一些大型项目（总工作量达到或超过 30 个人年）中收集到的工作量分布情况而推导出来的，但也可以应用在一些较小的软件项目中。如下面的公式(3-1)所示：

$$L = C_k \times K^{\frac{1}{3}} \times t_d^{\frac{4}{3}} \tag{3-1}$$

其中：L 为源代码行数（以 LOC 计）；K 为整个开发过程所花费的工作量（以人年计）；t_d 为开发持续时间（以年计）；C_k 为技术状态常数，反映“妨碍开发进展的限制”，其取值因开发环境而异，如表 3-2 所示。

表 3-2　C_k 的典型值开发环境举例

C_k	评价	开 发 环 境
2000	差	没有系统的开发方法，缺乏文档和复审
8000	好	有合适的系统的开发方法，有充分的文档和复审
11000	优	有自动的开发工具和技术

7. COCOMO 模型

1）概述

COCOMO 模型是一种精确的、易于使用的成本估算方法。1981 年由 Boehm 提出，是一种参数化的项目估算方法，参数建模是把项目的某些特征作为参数，通过建立一个数字模型预测项目成本。

在 COCOMO 模型中，工作量调整因子（Effort Adjustment Factor，EAF）代表多个参数的综合效果，这些参数使得项目可以特征化和根据 COCOMO 数据库中的项目规格化。每个参数可以定位很低、低、正常、高、很高。每个参数都作为乘数，其值通常在 0.5～1.5 之间，这些参数的乘积作为成本方程中的系数。

COCOMO 模型具有估算精确、易于使用的特点。在该模型中使用的基本量有以下几个：

(1) DSI：源指令条数，定义为代码行数，包括除注释行以外的全部代码。若一行有两个语句，则算做一条指令。

(2) MM：表示开发工作量，度量单位为人月。

(3) TDEV：表示开发进度，由工作量决定，度量单位为月。

(4) COCOMO 模型重点考虑 15 种影响软件工作量的因素，并通过定义乘法因子，从而准确、合理地估算软件的工作量。

2) 3 种模型

COCOMO 模型用 3 个不同层次的模型来反映不同程度的复杂性。

(1) 基本模型(Basic Model)：是一个静态单变量模型，它用一个以已估算出来的源代码行数(LOC)为自变量的函数来计算软件开发工作量。

(2) 中间模型(Intermediate Model)：在用 LOC 为自变量的函数计算软件开发工作量的基础上，再用涉及产品、硬件、人员、项目等方面属性的影响因素来调整工作量的估算。

(3) 详细模型(Detailed Model)：包括中间 COCOMO 模型的所有特性，但用上述各种影响因素调整工作量估算时，还要考虑对软件工程过程中分析、设计等各步骤的影响。

3) 3 种模式

同时根据不同应用软件的不同应用领域，COCOMO 模型划分为如下 3 种软件应用开发模式。

(1) 组织模式(Organic Mode)：这种应用开发模式的主要特点是在一个熟悉稳定的环境中进行项目开发，该项目与最近开发的其他项目有很多相似点，项目相对较小，而且并不需要许多创新。

(2) 嵌入式应用开发模式(Embedded Mode)：在这种应用开发模式中，项目受到接口要求的限制，接口对整个应用的开发要求非常高，而且要求项目有很大的创新，例如开发一种全新的游戏。

(3) 中间应用开发模式(Semidetached Mode)：这是介于组织模式和嵌入式应用开发模式之间的类型。

但是 COCOMO 模型也存在一些很严重的缺陷，例如：分析时的输入是优先的；不能处理意外的环境变换；得到的数据往往不能直接使用，需要校准；只能得到过去的情况总结，对于将来的情况无法进行校准；等等。

3.3.3 成本控制

1. 成本管理

1) 成本管理概述

成本管理是在项目具体实施过程中，为了确保完成项目所花费的实际成本不超过预算成本而展开的项目成本估算、项目预算、项目成本控制等方面的管理活动。

成本管理主要包括资源计划编制、成本估算、成本预算和成本控制等过程。其中，资源计划编制是确定项目需要的物资资源的种类和数量；成本估算是编制一个为完成项目各活动所需要的资源成本的近似估算；成本预算是将总成本估算分配到各单项工作活动上；成本控制是控制项目预算的变更。

资源计划是成本估算的基础和前提；有了成本估算才可以进行成本预算，并将成本分配到各个单项任务中；然后在项目实施过程中通过成本控制保证项目的成本不超过预算。所以，软件的成本估算是成本管理的中心环节。

2）成本管理的基本原则

（1）合理化原则。成本管理的根本目的在于通过成本管理的各种手段，促进不断降低项目成本，以达到可能实现最低目标成本的要求。但是，项目的成本并非越低越好，应研究成本降低的可能性和合理的成本最低化。一方面应挖掘各种成本降低的潜力，使可能性变为现实；另一方面应从实际出发，制定通过主观努力可能达到的合理的费用水平。

（2）全面管理的原则。成本管理应是全面、全过程、全员参加的管理，而不仅仅是局部的、某些阶段、某些人员参加的管理。

（3）责任制原则。为实行全面成本管理应对成本费用进行层层分解、层层落实，明确各相关者的责任。

（4）管理有效原则。成本管理的有效化就是促使项目以最小的投入，获取最大的产出；以最少的人力和财力，完成较多的管理工作，提高管理效率。

（5）管理科学化原则。成本管理是一种科学管理，应按信息化项目的客观规律，采用科学的方法合理确定项目的成本目标、动态管理费用发生的过程，有效降低成本的支出，最优实现项目的成本目标。

（6）管理动态性原则。信息化项目成本管理具有动态特性，所以项目的成本管理应考虑动态性原则。即项目在进行过程中，成本可能会发生变更，无论是项目供方还是项目需方都应充分考虑项目成本的可变性。

2. 成本控制

项目成本控制是指项目组织为保证在变化的条件下实现其预算成本，按照事先拟订的计划和标准，通过采用各种方法，对项目实施过程中发生的各种实际成本与计划成本进行对比、检查、监督、引导和纠正，尽量使项目的实际成本控制在计划和预算范围内的管理过程。随着项目的进展，根据项目实际发生的成本项不断修正原先的成本估算和预算安排，并对项目的最终成本进行预测的工作也属于项目成本控制的范畴。项目成本控制工作的主要内容包括以下几个方面。

（1）识别可能引起项目成本基准计划发生变动的因素，并对这些因素施加影响，以保证该变化朝着有利的方向发展。

（2）以工作包为单位，监督成本的实施情况，发现实际成本与预算成本之间的偏差，查找出产生偏差的原因，做好实际成本的分析评估工作。

（3）对发生成本偏差的工作包实施管理，有针对性地采取纠正措施，必要时可以根据实际情况对项目成本基准计划进行适当的调整和修改，同时要确保所有的相关变更都准确地记录在成本基准计划中。

（4）将核准的成本变更和调整后的成本基准计划通知项目的相关人员。

（5）防止不正确的、不合适的或未授权的项目变动所发生的费用被列入项目成本预算。

（6）在进行成本控制的同时，应该与项目范围变更、进度计划变更、质量控制等紧密结

合，防止因单纯控制成本而引起项目范围、进度和质量方面的问题，甚至出现无法接受的风险。

有效成本控制的关键是经常、及时地分析成本绩效，尽早发现成本差异和成本执行的无效率，以便在情况变坏之前能够及时采取纠正措施。一旦项目成本失控，在预算内完成项目是非常困难的，如果项目没有额外的资金支持，那么成本超支的后果就是要么推迟项目工期，要么降低项目的质量标准，要么缩小项目的工作范围，这 3 种情况是各方都不愿意看到的。

3. 项目成本控制的依据

1）项目各项工作或活动的成本预算

项目各项工作或活动的成本预算是根据项目的工作分解结构图为每个工作包进行的预算成本分配，在项目的实施过程中，通常以此为标准对各项工作的实际成本发生额进行监控，是进行成本控制的基础性文件。

2）成本基准计划

成本基准计划是按时间分段的费用预算计划，可用来测量和监督项目成本的实际发生情况，并能够将支出与工期进度联系起来，时间是对项目支出进行控制的重要依据。

3）成本绩效报告

成本绩效报告是记载项目预算的实际执行情况的资料，其主要内容包括项目各个阶段或各项工作的成本完成情况、是否超出了预先分配的预算、存在哪问题等。通常用以下 6 个基本指标来分析项目的成本绩效。

（1）项目计划作业的预算成本：是按预算价格和预算工作量分配给每项作业活动的预算成本。

（2）累积预算成本：将每一个工作包的总预算成本分摊到项目工期的各个区间，这样计算出截止到某期的每期预算成本汇总的合计数，成为该时点的累积预算成本。

（3）累积实际成本：已完成工作的实际成本，截止到某一时点的每期发生的实际成本额的合计数。

（4）累积盈余量：已完成工作的预算成本，由每一个工作包的总预算成本乘以该工作包的完工比率得到。

（5）成本绩效指数：衡量成本效率的指标，是累积盈余量同累积实际成本的比值，反映了用多少实际成本才完成了一单位预算成本的工作量。

（6）成本差异：累积盈余量同累积实际成本之间的差异。

4）变更申请

变更申请是项目的相关利益者以口头或者书面的方式提出的有关更改项目工作内容和成本的请求，其结果是增加或减少项目成本，有关项目的任何变动都必须经过项目投资者、客户的同意，以获得他们的资金支持。项目管理者要根据变更后的项目工作范围或成本预算来对项目成本实施控制。

5）项目成本管理计划

项目成本管理计划对在项目的实施过程中可能会引起项目成本变化的各种潜在因素进行识别和分析，提出解决和控制方案，为确保在预算范围内完成项目提供一个指导性的

文件。

4. 项目成本控制的方法

有效的成本控制的关键是经常、及时地分析费用绩效，以便在情况变坏之前能够采取纠正措施积极解决它，从而减缓对项目范围和进度的冲击。以下是成本控制的常用工具和技术。

1）成本变更控制系统

这是一种项目成本控制的程序性方法，主要通过建立项目成本变更控制体对项目成本进行控制。该系统主要包括3个部分：成本变更申请、核准成本变更申请和变更项目成本预算。

提出成本变更申请的人可以是项目投资者、客户、项目管理者、项目经理等项目的一切利益相关者。所提出的项目成本变更申请呈交到项目经理或项目其他成本管理人员，然后这些成本管理者根据严格的项目成本变更控制流程，对这些变更申请进行一系列的评估，以确定该项变更所导致的成本代价和时间代价，再将变更申请的分析结果报告给项目投资者、客户，由他们最终判断是否接受这些代价，核准变更申请。变更申请被批准后，需要对相关工作的成本预算进行调整，同时对成本基准计划进行相应的修改。最后，注意成本变更控制系统应该与其他变更控制系统相协调，成本变更的结果应该与其他变更结果相协调。

2）绩效测量

绩效测量是指主要用于估算实际发生的变化的方法，如挣值法等。在费用控制过程中，要把精力放在那些费用绩效指数小于1或费用差异小的工作包上，而且费用绩效指数和费用差异越小越要优先考虑，以减少费用或提高项目进行的效率。在采取措施时主要应针对近期的工程活动和具有较大估计费用的活动上，因为越晚采取行动则造成的损失就可能越大，纠正的可能性也就越小，而费用估算较大的活动，减少其成本的机会也就越多。

具体而言，降低项目费用的方法有很多种，如改用满足要求但成本较低的资源，提高项目团队的水平以促使他们更加有效地工作，或者减少工作包和特定活动的作业范围和要求。

另外，即使费用差异为正值，也不可掉以轻心，而要想办法控制项目费用，让其保持下去，因为一旦费用绩效出现了麻烦，再要使它回到正轨上来往往是很不容易的。

3）挣值法

挣值法是用以分析目标实施与目标期望之间差异的一种方法。挣值法又称为赢得值法或偏差分析法。

挣值法通过测量和计算已完成工作的预算费用与已完成工作的实际费用，将其与计划工作的预算费用相比较得到项目的费用偏差和进度偏差，从而达到判断项目费用和进度计划执行状况的目的。

3.4　软件质量管理

本节重点介绍质量概述、软件质量的概念、软件质量的相关概念、软件质量度量、软件过程和质量保证。

3.4.1 质量概述

1. 质量定义

在不同的时期，国际标准化组织 ISO 对质量的定义也不相同。

ISO 8402—1986 中将质量定义为“反映产品和服务满足明确和隐含需要的能力的特性和特征综合”。

ISO 840—1994 对质量的定义是“反映实体满足明确和隐含需要的能力的特性的总和。”而这里的“实体”是指“可单独描述和研究的事物”。

ISO 8402—2000 对质量的定义为“一组固有特性满足要求的程度”。其中，“要求”是指“明示的、通常隐含的或必须履行的需求或期望”。“通常隐含”是指组织、顾客和其他相关方的惯例或一般习惯，所考虑的要求或期望是不言而喻的。

显然 ISO 8402—2000 的质量不仅包含产品和服务都要满足客户的需求，还应该包括不断增加其竞争力以及有别于竞争对手的特性。现在消费者的质量观有了新的发展，要求得到的不仅仅是产品的功能质量，更多的包括与产品有关的系统服务。“满足顾客的需要”不仅包含产品和服务都要满足顾客的需要，而且还应包括增加其竞争力和有别于竞争性产品和服务的特性。除了基本功能之外，产品质量还应包括品牌、款式、包装、服务、付款方式、超值等内容。

2. 质量的历史

质量管理理论始于 20 世纪初期，质量管理从出现到现在大体经历了 5 个阶段。

(1) 产品质量检验阶段：其特征是对产品的质量进行检验。产品质量的检验只能是一种事后的检查，因此不能预防不合格品的产生。

(2) 以产品为中心的统计质量管理阶段：1924 年，美国 Bell 实验室的 Walter Shewhart，运用概率论和数理统计的原理，首先提出了控制生产过程，预防不合格品产生的思想和方法。即通过小部分样品测试，推测和控制全体产品或工艺过程的质量状况。二次大战以后，逐步形成了统计质量控制(SQC)的方法，用控制图表对生产过程中取得的数据进行统计分析，分析不合格品产生的原因，采取措施，使生产过程保持在不出废品的稳定状态。

(3) 以顾客为中心的质量保证阶段：企业为了保护原有市场并开拓新市场，必然会特别重视顾客的各种需求。为此，企业将花费很大的精力用于调查与搜集顾客对质量的各项要求，进一步将单独顾客的各项需求汇总形成若干个指标组，每项指标都规定了应达到的质量标准，它是企业进行生产需达到的最低要求。

(4) 强调持续改进的质量管理阶段：为了适应企业连续生产与经营的要求，针对软件产品持久改进的特点，质量管理工作在上述关注顾客需求的基础上需要有新的突破。企业在重视以往用户当前需求的同时，需要考虑用户的未来需求以及生产者的长远利益和企业长期维护成本之和。

(5) 全面质量管理阶段：20 世纪 50 年代末，质量管理专家 Edwards Deming，Joseph Juran 等人提出了全面质量管理的(Total Quality Control，TQC)概念。Edwards Deming

被誉为现代质量思想之父。

3. 历史人物与思想

Walter Shewhart 在 20 世纪 20 年代曾是美国 Bell 实验室的一名统计员，后来他成为了统计质量改进的奠基人。Shewhart 模型是解决问题和过程控制的系统方法，它由 4 个步骤组成，用于持续过程的改进。这 4 个步骤分别是计划（Plan）、实施（Do）、检查（Check）和处理（Act），被称作 PDCA 模型或 Shewhart 环。

Edwards Deming 是质量运动中的一个主要人物。他认为，仅仅组织中的每一个人都竭尽全力是不够的，还需要组织中有一致的目标和方向。也就是说，首要的是人们要知道去做什么，组织中所有个体都要有同一个坚定不移的目标，以确保最终获得成功。Deming 关于质量管理的 14 点原理的应用转变了传统的西方管理模式。

Joseph Juran 是质量运动的另一位巨人，他赞同质量的自顶向下方法。Juran 把质量定义为“适用性”，认为质量问题是管理人员的直接职责，管理人员必须经过计划、控制和改进来保证质量。质量三部曲（计划、控制和改进）被称为 “Juran 三部曲”。

Philip Crosby 是质量运动的巨人之一，他的观点影响了能力成熟度模型 CMM。他在 *Quality is Free* 一书中描述了一次性成功原理，即零缺陷（Zero Defect，ZD）程序，他把质量定义为“符合需求”，并且认为人们受到“错误是不可避免”的观点的限制。

Shingo 是零缺陷理论的重要人物。他的理论包括确定过程中的潜在错误源，对这些错误源进行监控。对所有已发现的错误进行因果分析，然后排除其根本原因。这种方法能消除所有可能出现的错误，因此只有一些异常的错误可能会出现，然后将这些异常错误及其起因排除掉。失效模式和影响分析（Failure Mode and Effects Analysis，FMEA）方法就是这种方法的演变。确定系统或子系统可能出现的失效并对其进行分析，把失效的原因、产生的影响以及概率记录下来。

Genichi Taguchi 对质量的定义完全不同，他把质量定义为“一个产品在装运后对社会造成的损失，而不是产品的内在功能造成的损失”。Taguchi 定义了一个损耗函数来作为质量成本的度量：$L(X)=C(X-T)^2+k$。Taguchi 也找到了一种确定过程变量的最佳值的方法，这个最佳值能在保持过程均值正确的同时使得过程中的变动最小。

Kaoru Ishikawa 因为他在质量控制圈（Quality Control Circles，QCC）中所做的工作而闻名。质量控制圈是一个小的雇员组，他们做相似的工作，约定进行定期碰头以确定并分析与工作相关的问题，集体讨论并推举和实施解决方案。问题解答则用于 Pareto 分析、鱼骨图、柱状图、散点图和控制图表这类工具。

Armand Feigenbaum 由于在全面质量控制（TQC）方面所做的工作而闻名。（TQC）涉及应用于组织内所有功能的质量保证，它与全面质量管理（TQM）有所区别：TQC 是要全面控制质量，而 TQM 则体现了涉及整个组织内的全体员工和功能的质量管理和改进原理。

4. 顾客满意度

1）顾客满意的定义

在 2000 年新版的 ISO 9000 族标准中，进一步强调了顾客满意，顾客满意成为评价质量

的标准。有些组织简单地认为“顾客满意”只是在售后服务阶段，通过优质服务就可以做到顾客满意。而实际上，让顾客满意的第一步是从市场调查了解顾客的需求开始，然后在设计、加工、销售、服务的全过程中，努力去满足顾客的需求，以顾客为中心的思想贯穿于产品和/或服务质量形成的全过程。组织要努力实现顾客满意，而且不能停留在顾客满意的水平上，还要继续努力，从顾客满意提高到顾客忠诚。顾客忠诚是指在顾客满意的基础上，对某品牌或组织做出长期购买的承诺，是顾客一种意识与行为的结合。顾客满意一般是指一次性的；顾客对某品牌或组织由满意发展到忠诚后，他会再次购买同一品牌的产品和/或服务。

2）顾客满意度指数模型

顾客满意度指数模型中的六大要素分别是顾客期望、顾客对质量的感知、顾客对价值的感知、顾客满意度、顾客抱怨、顾客忠诚，如图 3-9 所示。

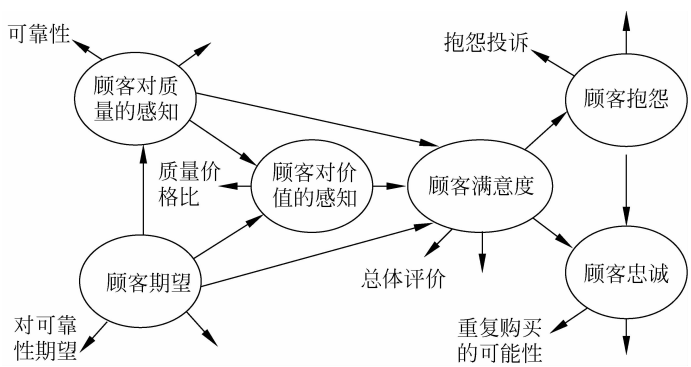

图 3-9　顾客满意度指数模型

其中，顾客期望、顾客对质量的感知、顾客对价值的感知决定着顾客满意程度，是系统的输入变量；顾客满意度、顾客抱怨、顾客忠诚是结果变量。

顾客满意是软件开发项目的主要目的之一，而顾客满意目标要得以实现，需要建立顾客满意度度量体系和指标对顾客满意度进行度量。顾客满意度指标（Customer Satisfaction Index，CSI）以顾客满意研究为基础，对顾客满意度加以界定和描述。项目顾客满意度量的要点在于确定各类信息、数据、资料来源的准确性、客观性、合理性、有效性，并以此建立产品、服务质量的衡量指标和标准。企业顾客满意度度量的标准会因为各企业的经营理念、经营战略、经营重点、价值取向、顾客满意度调查结果等因素而有所不同。比如，NEC 于 2002 年 12 月开始实施的 CSMP 活动的度量尺度包括共感性、诚实性、革新性、确实性和迅速性，其中，将共感性和诚实性作为 CS 活动的核心姿态，而将革新性、确实性和迅速性作为提供商品和服务中不可或缺的尺度。每个尺度包括两个要素，各要素包括两个项目，共计 5 个尺度、10 个要素和 20 个项目。例如，共感性这一尺度包括“了解顾客的期待”、“从顾客的立场考虑问题”这两个要素；“了解顾客的期待”这一要素又包括“不仅仅能胜任目前的工作还能意识到为顾客提供价值而专心投入”、“对顾客的期望不是囫囵吞枣而是根据顾客的立场和状况来思考‘顾客到底需要什么’并加以应对”这两个项目。

3.4.2　软件质量概述

1. 软件质量定义

在我国国家标准 GB/T 11457—89，软件质量 Software Quality 的条目为 2.434，它的定义是：

a. 软件产品中能满足给定需求的性质和特性的总和。例如，符合规格说明。

b. 软件具有所期望的各种属性的组合程度。

c. 顾客和用户觉得软件满足其综合期望的程度。

d. 软件的合成特性。它确定软件在使用中将满足顾客预期要求的程度。

在我国国家标准 GB/T 11457—1995，软件质量(Software Quality)的条目为 2.454，它的定义是：

a. 软件产品中能满足给定需求的性质和特性的总和。例如，符合规格说明。

b. 软件具有所期望的各种属性的组合程度。

c. 顾客和用户觉得软件满足其综合期望的程度。

d. 确定软件在使用中将满足顾客预期要求的程度。

2. 软件质量特性

1) 软件质量要素

1977 年 McCall 率先提出了软件质量要素包含的内容。后来，ISO 将其转为 ISO/IEC 9126—91 标准，软件质量有 6 个特性，即功能性、可靠性、易使用性、效率、可维护性和可移植性。

2) 软件质量的二级特性指标

McCall 认为，软件的质量由以下 11 个要素构成，包括正确性、可靠性、效率、完整性、可使用性、可维护性、可测试性、灵活性、可移植性、重复使用性和连接性。

而 ISO/IEC 9126—91 则将软件质量做了进一步刻画细分，并且有助于描述各个软件特性之间的关系。软件质量特性由下列 21 个二级质量特性所决定，即合适性、精确性、互操作性、依从性、安全性、成熟性、容错性、易恢复性、易理解性、易学性、易操作性、时间特性、资源特性、易分析性、易改变性、稳定性、易测试性、适应性、易安装性、遵循性和易替换性。

3) McCall 三层次质量度量模型

McCall 认为，软件质量要素是软件的质量特征，软件质量属性是软件质量的评价准则，评价准则还需要定量的度量。质量要素、评价准则和度量构成了 McCall 的三层次质量度量模型。在这个层次模型中，度量处于模型的最低层，它是由质量保证人员根据开发过程的特征用评分的方式对质量准则做出的定量评价。

3. 软件用户满意度

美国专家 Stephen H. Kan 在《软件质量工程的度量与模型》(*Metrics and Models in Software Quality Engineering*)中认为，企业的顾客满意度要素如表 3-3 所示。

表 3-3 顾客满意度要素

顾客满意度要素	顾客满意度要素的内容
技术解决方案	质量、可靠性、有效性、易用性、价格、安装、新技术
支持与维护	灵活性、易达性、产品知识
市场营销	解决方案、接触点、信息
管理	购买流程、请求手续、保证期限、注意事项
交付	准时、准确、交付后过程
企业形象	技术领导、财务稳定性、执行印象

作为企业的顾客满意度的基本构成单位，项目的顾客满意度会受到项目要素的影响，主要包括开发的软件产品、开发文档、项目进度以及交期、技术水平、沟通能力、运用维护等。具体而言，可以细分为如表 3-4 所示的度量要素，并根据这些要素进行度量。

表 3-4 顾客满意度要素细分

顾客满意度项目	顾客满意度度量要素
软件产品	功能性、可靠性、易用性、效率性、可维护性、可移植性
开发文档	文档的构成、质量、外观、图表以及索引、用语
项目进度以及交期	交期的根据、进度迟延情况下的应对、进展报告
技术水平	项目组的技术水平、项目组的提案能力、项目组的问题解决能力
沟通能力	事件记录、式样确认、Q&A
运用维护	支持、问题发生时的应对速度、问题解决能力

3.4.3 软件质量的相关概念

1. 基本概念

要讨论软件质量问题，有些概念是不能回避的。隐错(bug)、缺陷(defect)、错误(error)、失效(failure)、故障(fault)、可靠性(reliability)等词汇是与质量密切相关的概念，有些概念的含义非常接近，在使用这些词汇时很容易混淆，因此有必要进行讨论。下面就从这几个基本术语的定义着手，介绍这些词汇的含义，同时对这些术语之间的关系和差别等进行讨论。

国家标准 GB/T 11457—89“软件工程术语”是等同采用 IEEE STD 729—1983 [38] 制定的，其权威性是一般刊物或文章上的见解所无法比拟的。随后，国家标准 GB/T 11457—1995“软件工程术语”代替了 GB/T 11457—89。在新的 GB/T 11457—1995 标准中，错误、故障、失效等概念都有相应的定义，下面给出其标准的条目、条目号和内容。

1) 隐错(bug)

在 GB/T 11457—89 旧的标准中它的条目为 2.53，它引导读者去参见 2.196 条。2.196 条是故障、缺陷(fault)的定义。

a. 功能部件不能执行所要求的功能的意外。

b. 在软件中表示 2.175(b.)关于错误的解释。如果遇到，它可能引起失效。

在GB/T 11457—1995新的标准中它的条目为2.54，它引导读者去参见2.198条。2.198条是故障、缺陷(fault)的定义：

a. 功能部件不能执行所要求的功能。

b. 在软件中表示2.176b关于错误的解释。如果遇到，它可能引起失效。

2) 缺陷(defect)

在GB/T 11457—89旧的标准中它的条目为2.131，它引导读者去参见2.196条。2.196条是故障、缺陷(fault)的定义，其内容同上。

在GB/T 11457—1995新的标准中它的条目为2.131，它同样引导读者去参见2.198条。2.198条是故障、缺陷(fault)的定义，其内容同上。

Humphrey在《个体软件过程》(2001)书中对软件缺陷的阐述是“软件缺陷是指程序中存在的错误，例如语法错误、拼写错误、标点符号错误或者是一个错误的程序语句，是任何影响到程序完整而有效地满足用户要求的东西。缺陷可能出现在程序中和设计中，甚至在需求、规格说明或其他文档中；缺陷可能是冗余的语句、不正确的程序语句或是被忽略的程序部分。事实上，缺陷是任何影响到程序完整而有效地满足用户要求的东西。因此，一个缺陷是客观的事物，是可以标识、描述和统计的。软件缺陷与软件错误又不同。软件错误是指人们的期望和系统实际具有的状态或行为之间的偏差。缺陷是静态的，而错误则包括静态和动态的，但并不是所有的缺陷都能产生错误。软件缺陷基本上来源于程序员的疏忽大意”。

3) 错误、出错、误差(error)

在GB/T 11457—89旧的标准中它的条目为2.175，它的定义是：

a. 计算、观察、测量的值或条件和实际的、规定的或理论上的值或条件不符合。

b. 导致产生会有故障软件的人的行动。例如遗漏或误解软件说明书中用户的需求，不正确的翻译或遗漏设计规格说明书中的需求。这不是本词的优先选用的用法。

在GB/T 11457—1995新的标准中它的条目为2.176，它的定义是：

a. 计算、观察、测量的值或条件和实际的、规定的或理论上的值或条件不符合。

b. 导致产生含有缺陷的软件的人的行动。例如遗漏或误解软件说明书中用户的需求，不正确的翻译或遗漏设计规格说明书中的需求。

4) 失效(failure)

在GB/T 11457—89旧的标准中它的条目为2.190，它的定义是：

a. 功能部件执行其功能的能力的终结。

b. 系统或系统组成部分丧失了在规定的限度内执行所要求的功能的能力。当遇到故障情况时就可能出现失效。

c. 程序操作背离了程序需求。

在GB/T 11457—1995新的标准中它的条目为2.192，它的定义是：

a. 功能部件执行其功能的能力的丧失。

b. 系统或系统组成部分丧失了在规定的限度内执行所要求的功能的能力。当遇到故障情况时系统就可能失效。

c. 程序操作背离了程序需求。

比较：两个定义只在第一点上有区别。一个是“终结”能力，一个是“丧失”能力，显然，后者用词更准确。

5) 故障、缺陷(fault)

在 GB/T 11457—89 旧的标准中它的条目为 2.196，它的定义在前文的 1)中已介绍。

在 GB/T 11457—1995 新的标准中它的条目为 2.198，它的定义在前文的 1)中已介绍。

比较：两个定义只在第一点上有区别。显然，后者用语更准确。

6) 可靠性(reliability)

在 GB/T 11457—89 旧的标准中它的条目为 2.378，它的定义是：

在规定的时间间隔内和条件下，一项目实现所要求的功能的能力。

在 GB/T 11457—1995 新的标准中它的条目为 2.392，它的定义是：

在规定的时间间隔内和条件下，一项目配置所要求的功能的能力。

比较：尽管它们只有两个字的差别，但是，“实现”与“配置”的含义完全不同，前者指软件的功能，后者指用户需求的要求。

7) 软件可靠性(software reliability)

在 GB/T 11457—89 旧的标准中它的条目为 2.436，它的定义是：

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，该概率是系统输入和系统使用的函数，也是软件中存在的错误的函数。系统输入将确定是否会遇到已存在的错误(如果错误存在的话)；

b. 在规定的时间周期内所述条件下程序执行所要求的功能的能力。

在 GB/T 11457—1995 新的标准中它的条目为 2.454，它的定义是：

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，该概率是系统输入和系统使用的函数，也是软件中存在的缺陷的函数。系统输入将确定是否会遇到已存在的缺陷(如果缺陷存在的话)；

b. 在规定的时间周期内所述条件下程序执行所要求的功能的能力。

比较：前后仅有“错误”与“缺陷”两个词的不同，这说明，新标准的定义更严格，因为缺陷一词具有法律意义。

8) 比较

从以上定义可以看出，无论是新版本还是旧版本，“故障、缺陷、隐错(通俗翻译为‘臭虫’)是同义词，软件的故障是由人的行动产生的，也就是说，是在开发过程中由于人的某些疏忽造成了软件中的错误或者说导致软件中存在故障、缺陷、臭虫。它们隐蔽在软件中，如果在软件运行中被意外触发将导致软件失效。所以软件错误和软件故障具有相同或十分近似的含义，在 GB/T—11457 中的其他条目中也没有刻意对这两个词汇加以区别。”

为了真正理解以上几个概念的差别和联系，应该从英语单词的含义上着手。实际上，有几个概念是非常相近的。这一点从两个“软件工程术语”国家标准中可以看出。比如，隐错(bug)、缺陷(defect)和故障(fault)，都是指“毛病”或“缺点”，但是它们还是有区别的。bug

在《新英汉词典》(上海译文出版社,1979)中第一个解释是“臭虫”,第四个解释是机器等上的缺陷、瑕疵;defect 着重某种欠缺而影响到质量;而 fault 多指性格上的“弱点”或行为上的“过失”以及过失的责任(《英语常用词用法词典》,北京大学西语系英语专业编,商务印书馆,1983)。《新实际英语用法大词典》(王文昌,上海外语教育出版社,1997)对 error 和 mistake 作了解释:它们都是普通用语,可通用。但 error 较正式,更多指偏离规范或标准的错误,如逻辑或计算错误,而 mistake 指生活中的任何错误。由此可以看出,这就是“软件工程术语”国家标准中有对 error 解释而没有对 mistake 解释的原因。换句话说,mistake 一词一般不在“软件工程术语”中使用。失效(failure)与以上的几个词差别比较大,《新英汉词典》的解释是失败、缺乏、不足、(机器)失灵、衰退等。

2. 不合格与缺陷

在 2000 年版 ISO 9000 族标准中,不合格的定义为“未满足要求”。“要求”涵盖了明示的需求与期望、通常隐含的需求与期望和必须履行的需求与期望。

新版的 ISO 9000 族标准对不合格的定义做了较大的更改,删去了旧定义中的以“规定”为判罚依据的办法,明确要求以满足顾客需要为宗旨,反映出新版标准对质量提出了更高的需要,包括引导消费的超前需求,才是合格产品的质量要求,否则就是不合格。从这个发展趋势看,产品质量已从“满足标准规定”,发展到“让顾客满意”,到“超越顾客的期望”的新阶段。另外,新定义的通用性更强,它不仅适用于硬件产品,也适用于服务业,同时又适用于过程质量和体系质量的评定。例如服务业如饭店,虽然有饭菜的制作规范,但仍应满足不同顾客的口味爱好、风俗习惯的要求,以此作为合格与否的评定依据,不能按旧标准那样只要符合某种具体的规范,不管是否符合顾客的要求都是合格的。今后,凡是没有满足顾客要求的体系,都属于“未满足要求”的不合格之类。

在 2000 年版 ISO 9000 族标准中,缺陷是“未满足与预期或规定用途有关的要求”,缺陷有法律内涵,与产品责任有关,要慎用。

3. 接近零不合格

1) 零缺陷

20 世纪 60 年代初,在马丁公司(一家为美国军方提供武器的公司)担任过质量部主任的 Philip B. Crosby,为降低导弹的次品率,首先提出零缺陷(zero defect)的概念,在 1979 年写了一本著名的畅销书《质量不花钱》(Quality is free),该书将“第一次就完全做对”这一口号传播到世界各地,Crosby 本人也在世界质量界一举成名。在零缺陷的质量管理理论中,缺陷概念与 ISO 9000 中的缺陷概念是不同的。零缺陷质量管理中的缺陷是传统的缺陷概念,简单地说,就是有毛病、不符合标准与规范。

2) 零不合格过程

产品与服务要完全满足要求。为实现这一目标,要对过程进行严格控制,从而使过程的输出为零不合格。

3）接近零不合格

零不合格过程是一种理想的质量保证状态，从统计的角度看，不存在完全绝对的零不合格过程，只能通过过程的持续改进，向零不合格过程不断逼近。这就是接近零不合格过程概念的由来。即：

不合格⇒零不合格⇒零不合格过程⇒接近零不合格过程

接近零不合格过程的质量控制理论所研究的就是如何对过程进行科学严格的控制，以保证过程不断向零不合格过程逼近。接近零不合格过程的质量控制理论是质量科学的最新分支，是21世纪必须面对、也必将得到解决的课题。“接近零不合格过程”首先由清华大学经济管理学院孙静博士提出。

3.4.4 软件质量度量

1. 历史发展

软件度量的历史几乎与软件工程的历史一样长。软件度量理论始于20世纪60年代末期，20世纪70年代得到逐渐发展，随着20世纪80年代、90年代这一工作的全面开展，其主要框架和一些重要结论基本形成。

软件度量学的概念最初由Rubey和Hurtwick于1968年首次提出。最早的软件度量方法是度量代码行数LOC(Line of Codes)，这是一种应用时间最长、最普遍的度量方法，同时也是受到许多批评的方法，直到今天还在许多企业中应用。

软件质量度量方法和技术经历了几个重要的时期和阶段。1976年，Boehm第一次提出了软件质量度量的层次模型。1978年，Walters和McCall等人提出了从软件质量要素、准则到度量的3个层次式的模型。1985年，ISO建议软件质量模型由三层组成：高层为软件质量需求评价准则，中层为软件质量设计评价准则，低层为软件质量度量评价准则。1990年上海软件中心在ISO/TC97/SC7基础上进一步提出了SSC软件质量评价体系。

1978年，Boehm等人出版了《软件质量特性》一书，提出了定量地评价软件质量的概念，给出了60个质量度量公式，以及用于评价软件质量的方法，并且首次提出了软件质量度量的层次模型。Boehm等人认为，软件产品的质量基本上可从软件的可使用性、可维护性和可移植性三方面考虑。可使用性分为可靠性、效率和人工工程3个方面，反映用户的满意程度；可维护性可以从可测试性、可理解性、可修改性3个侧面进行度量，反映公司本身的满意程度；可移植性被划分为第三层。在20世纪70年代，软件业很重视软件移植，因为当时的硬件系统尚不成熟，主流系统还不明显，而到了20世纪80年代之后，由于主流硬件系统已经基本形成，软件移植已不再如以前那样重要了，取而代之的是软件的重用性，因为它是软件价值的反映，是未来的开发者对该软件的满意程度。

不久McCall等人提出了从软件质量要素(factor)、准则(criteria)到度量(metric)的三层次软件质量度量模型，并且定义了11个软件质量要素，给出了各要素的关系，在要素下面定义了几个评价准则。McCall等人认为，要素是软件质量的反映，而软件属性可用做评价准则，定量化地度量软件属性，从而反映软件质量的优劣。McCall定义的11个质量要素分

别为正确性、可靠性、效率、完整性、可使用性、可维护性、可测试性、灵活性、可移植性、重复使用性及连接性。自从 McCall 等在 1978 年完成他们最初的工作后，几乎关于计算的每一个方面都发生了根本的改变，但提供软件质量指标的属性仍然没有改变。

ISO 的三层次模型与 McCall 等人的模型相似，ISO 的高层、中层和低层分别与 McCall 等人模型中的要素、评价准则和度量相对应。根据 ISO 的观点，高层和中层应建立国际标准以便在国际范围内推广应用 SQM 技术，而低层可由各公司、单位视实际情况制定。

ISO/IEC 9126 将软件质量特性减少到 6 个，并将 21 个子特性定义在附录中，为标准的实施留有充分的余地，其质量特性为功能性、可靠性、易使用性、效率、可维护性和可移植性。

IEEE Std 1061—1992 指定了软件质量度量方法学标准。它建立一个框架，类似于 McCall 的三层模型，但是框架中的要素和子要素允许增加、删除和修改。第一层是建立质量需求，定义质量属性，然后质量要素被分配给质量属性。如果需要，还要分配子要素给每一个要素。要素是面向管理者和用户的。第二层是描述面向软件的标定质量的子要素，它们通过将每个要素分解为可测量的软件属性而得到。子要素是独立的软件属性，因此可以对应于多个要素。在第三层中，子要素被分解为能测量系统产品和过程的度量。它的 6 个要素与 ISO/IEC 9126 的 6 个质量特性的定义几乎一字不差，子要素则与 ISO/IEC 9126 中的定义有一些区别。

我国软件行业协会上海分会于 1989 年制定的 SSC 模型将要素减少到 6 个，去除了安全性、灵活性和连接性，增加了可移植性，将正确性改为功能性，可使用性改为易用性，即包括功能性、可靠性、易用性、效率、维护性和可移植性 6 个质量特性作为质量模型的顶层，同时设置了二十几个质量子特性和一百多个质量度量，每个度量又由若干个用于收集的数据项组成(度量元)，该模型可针对八类软件产品的不同开发阶段实施质量度量。

2. ISO/IEC 9126 标准

ISO/IEC 9126(1991)：软件产品评估—质量特性及其使用指南纲要。该标准就是为支援此种需求而发展出来的，在标准中定义了 6 种质量特性，并且描述了软件产品评估过程的模型。ISO/IEC 9126 所定义的软件质量特性可用来指定客户及使用者在功能性与非功能性方面的需要。ISO/IEC 9126 软件质量模型是一种评价软件质量的通用模型，包括 3 个层次。

第一层质量要素：描述和评价软件质量的一组属性，包括功能性、可靠性、易用性、效率性、可维护性和可移植性等质量特性。

第二层衡量标准：衡量标准的组合反映某一软件质量要素，包括精确性、稳健性、安全性、通信有效性、处理有效性、设备有效性、可操作性、培训性、完备性、一致性、可追踪性、可见性、硬件系统无关性、软件系统无关性、可扩充性、公用性、模块性、清晰性、自描述性、简单性、结构性和文件完备性等。

第三层量度标准：可由各使用单位自定义。根据软件的需求分析、概要设计、详细设计、编码、测试、确认、维护与使用等阶段，针对每一个阶段制定问卷表，以此实现软件开发过

程的质量度量。

3. 应用举例

例 3-3 下面是作者对《现浇钢筋混凝土矩形清水池结构 CAD》大型软件的评价过程。

1) 软件质量准则

根据国际标准化协会颁布的“软件质量特性”国际标准(ISO/IEC 9126—91)和国家标准(GB/T 12260—96),将软件质量准则归纳为以下 7 个最基本的因素和 3 个选用的质量特性,分别为功能度准则、可靠性准则、可维护性准则、易使用性准则、可移植性准则、时间经济性准则、资源经济性准则、保密性、可再用性和可连接性,如表 3-5 所示。

表 3-5 软件质量评价表

质量设计标准	质量评价标准	工程质量评价标准的定义	评价	
			一 次	两 次
功能度	正确性	程序功能与需求说明书一致	4 3 2 1	4 3 2 1
	完整性	完成说明书中的全部功能要求	4 3 2 1	4 3 2 1
可靠性	可靠性	无故障、正常、连续地运行	4 3 2 1	4 3 2 1
	健壮性	输入出错时,程序正常运行	4 3 2 1	4 3 2 1
可维护性	可测试性	测试的难易性	4 3 2 1	4 3 2 1
	理解性	程序清晰,易读,易懂	4 3 2 1	4 3 2 1
	可修改性	交付使用后,对程序修改的难易	4 3 2 1	4 3 2 1
易使用性	简单性	操作和输入简单	4 3 2 1	4 3 2 1
	一致性	整个程序与用户界面的风格一致	4 3 2 1	4 3 2 1
	灵活性	用户界面形式多样	4 3 2 1	4 3 2 1
	反馈性	程序对用户的响应	4 3 2 1	4 3 2 1
	易学性	功能提出,输入和操作提示,运行提示,出错提示	4 3 2 1	4 3 2 1
可移植性	硬件独立性	输入输出接口为逻辑接口	4 3 2 1	4 3 2 1
	软件独立性	独立于其他软件系统	4 3 2 1	4 3 2 1
	规范性	国家标准	4 3 2 1	4 3 2 1
	可追踪性	确定程序的功能与内部模块及模块内的逻辑对应关系的难易程度,以及确定源程序的某个部分在本模块中和在整个程序中涉及范围的难易程度	4 3 2 1	4 3 2 1
	模块化	模块结构,功能单一	4 3 2 1	4 3 2 1
	结构化	符合软件工程结构化编程规定	4 3 2 1	4 3 2 1
时间经济性		只测试系统运行的速度(秒)	4 3 2 1	4 3 2 1
资源经济性		测试系统的磁盘容量、运行空间	4 3 2 1	4 3 2 1
保密性			4 3 2 1	4 3 2 1
可再用性		软件重用	4 3 2 1	4 3 2 1
可连接性		与其他软件接口的难易性	4 3 2 1	4 3 2 1

2）评价等级

如果采用比较简单的平均值评价，将软件的质量好坏分 4 个级别：4—优，3—良，2—中，1—差。

质量得分之和平均值：

$$M=(4\times A_4+3\times A_3+2\times A_2+A_1)/E$$

其中 M 为该准则项质量得分的平均值，A_i 是得分为 i 的评价项目个数($1\leqslant i\leqslant 4$)，评价项目数总和 $E=A_4+A_3+A_2+A_1$。具体用法和含义在下文介绍。

3）评价打分

软件质量分析如下：

(1) 功能度准则：正确性 4 分，完整性 4 分。

(2) 可靠性准则：可靠性 3 分，健壮性 4 分。

(3) 可维护性准则：可测试性 4 分，可理解性 4 分。

(4) 易使用性准则：简单性 4 分，一致性 3 分，灵活性 4 分，反馈性 4 分，易学性 3 分。

(5) 可移植性准则：硬件独立性 4 分，软件独立性 2 分，规范性 4 分，可追踪性 4 分，模块化 4 分，结构化 4 分。

(6) 时间经济性准则 4 分。

(7) 资源经济性准则 4 分。

(8) 保密性 3 分。

(9) 可再用性 2 分。

(10) 可连接性 3 分。

4）计算结果

软件评价前 3 条最重要，中间的 4 条比较重要，最后 3 条可选。

前 3 条的综合评价：$M_1=(4\times 9+3\times 2)/11=41/11=3.7272$

前 7 条的综合评价：$M_2=(4\times 24+3\times 2+2)/27=103/27=3.8148$

前 10 条的综合评价：$M_3=(4\times 24+3\times 4+2)/30=111/30=3.7$

3.4.5　软件过程

20 世纪 50 年代，关于软件质量，人们考虑更多的是机器码的对错和汇编语言正确与否；20 世纪 60 年代，软件危机出现，软件失败率高，错误率高，程序员最关心的是如何减少失败，减少错误，降低成本；20 世纪 70 年代，软件工程中的生命周期方法被广泛应用，人们更注意时间、费用和质量协调问题；20 世纪 80 年代，软件的复杂性增加，CMM 方法出现，人们的注意力转向软件过程控制；20 世纪 90 年代后，工业化的软件过程技术和质量保障技术已经成为发展软件产业的重要支柱。软件过程随着软件组织的特点不同和商业目标不同，特别是在网络环境下，经常处于动态的调整和定义与重定义状态。所以过程技术必须支持过程的动态定义和过程流的动态重组。软件过程流本质上由工作流组成。过程改善的关键是可以明确标识当前状态，并明确改进的方向。

国际上软件过程方面代表性技术有 CMU-SEI 提出的 CMM、PSP、TSP、CMMI、ISO 9000、ISO 12207、ISO 15504、BOOTSTRAP、SPICE、TickIT 等。21 世纪初，软件过程技术得到了进一步的重视和发展。软件度量学的最终目的是服务于软件质量控制与评价。首

先，它必须确定度量评价标准，为软件质量保证和管理奠定定量的基础。

1. CMM(软件生产能力成熟度模型)

CMM 是 SW-CMM(软件生产能力成熟度模型)的简称，1987 年由 SEI 提出。它是目前国际上最流行也是最实用的软件生产过程标准，它为软件企业的过程能力提供了一个阶梯式的进化框架，共有五级，即初始级、可重级、定义级、管理级和优化级。关键过程域(KPA)包含五类目标，即实施保证、实施能力、执行活动、度量分析和实施验证。

2. PSP(个人软件过程)

个人软件过程是由美国 Carnegie Mellon 大学软件工程研究所(CMU/SEI)的 Watts S. Humphrey 领导开发的，于 1995 年它的推出，可以说是由定向软件工程走向定量软件工程的一个标志。关于 PSP，前文已详述，这里不再重复。

3. TSP(团队软件过程)

TSP 对团队软件过程的定义、度量和改革提出了一整套原则、策略和方法，把 CMM 要求实施的管理与 PSP 要求开发人员具有的技巧结合起来，以按时交付高质量的软件，并把成本控制在预算的范围之内。在 TSP 中讲述了如何创建高效且具有自我管理能力的工程小组、工程人员如何才能成为合格的项目组成员、管理人员如何对群组提供指导和支持、如何保持良好的工程环境使项目组能充分发挥自己的水平等软件工程管理问题。

4. CMMI(能力成熟度集成模型)

CMMI 是能力成熟度集成模型，它是 CMM 模型的最新版本。早期的能力成熟度模型是一种单一的模型，较多地用于软件工程。随着应用的推广与模型本身的发展，该方法演绎成为一种被广泛应用的综合性模型，因此改名为 CMMI 模型。早期的 CMM 是美国国防部出资，委托美国 Carnegie Mellon 大学软件工程研究院开发出来的工程实施与管理方法。CMMI 在世界各地得到了广泛的推广与接受。

5. 软件企业 ISO 9000 质量管理体系标准

ISO 9001 是 ISO 9000 标准族中的一个很重要的质量保证标准，也是软件机构推行质量认证工作的一个基础标准。该标准于 1994 年由国际标准化组织公布，我国已及时地将其转化为国家推荐标准，并给予编号：GB/T 19001—1994。

这一标准明确规定了质量体系的要求，如果产品开发、生产者或称供方达到了这些要求，表明其具备了质量保证能力。制订这一标准的主要目的在于，通过防止从产品设计到售后服务的所有阶段中出现不合格，使得用户满意。

ISO 9001 标准在 20 个方面规定了质量体系要素。这 20 个方面分别是管理职责，质量体系，合同评审，设计控制，文件和资料的控制，采购，顾客提供产品的控制，产品标识和可追溯性，过程控制，检验和试验，检验、测量和试验设备的控制，检验和试验状态，不合格品的控制，纠正和预防措施，搬动、储存、包装、防护和交付，质量记录控制，内部质量审核，培训，服务，统计技术。

6. ISO/IEC 12207

ISO 生存周期分为 5 个阶段，即需求、设计、实现、测试和维护。1991 年 9 月 IEEE 标准化委员会制定的《软件生存周期过程开展标准》就这一点做了说明。接着 ISO/IEC 于 1994 年制定出《软件生存周期过程》标准草案，我国根据该草案制定了 GB/T8566—1995《信息技术、软件生存周期过程》国家标准。ISO/IEC 组织于 1995 年 8 月 1 日又发布了 ISO/IEC 12207 第一版“信息技术—软件生存周期过程”国际标准。该标准正文对 MIS 生存周期过程进行了全面的描述。

标准中的过程被分成三大类，即主要过程、支持过程和组织过程。主要过程是生命周期中的原动力，它们是获取、供应、开发、运行和维护。支持过程包括文档、配置管理、质量保证、验证、确认、联合评审、审计和问题解决。在其他过程中可以使用支持过程。组织过程有管理、基础设施、改进和培训。一个组织可以使用组织过程来建立、控制和改进生命周期过程。

7. ISO/IEC TR 15504

1991 年 6 月，国际标准化组织 ISO/IEC JTC1 的 SC7 分技术委员会通过一项研究计划，旨在调查制定软件过程评估的国际标准的需求。1993 年 7 月该国际评估标准的制定工作正式开始，国际标准项目代号为 ISO/IEC 15504 并由 SC7 的一个工作组具体负责。随后在 1995 年 6 月经过投票表决后，该标准的工作草案发布。世界各地的用户相继进行试用并提供了试用反馈报告。在 1996 年 9 月又颁布了工作草案最终版。

ISO/IEC TR 15504 过程评估标准，鼓励软件组织使用一致的、可靠的、可证明的方法评估他们的过程状态，并用评估结果来持续地改进软件过程，以提高产品质量。

ISO/IEC TR l5504 为软件过程评估提供了一个框架，并为实施评估以确保各种级别的一致性和可重复性提出了一个最小需求。该需求有助于保持评估结果前后一致，并提供证据证明其级别、验证与需求相符。

ISO/IEC TR 15504 标准是一个二维的结构：一个是过程维，包括客户—供应者过程、工程过程、支持过程、管理过程和组织过程；另一个是能力维，从低到高有 6 个级别，第 0 级是不完全级，第 1 级是可实施级，第 2 级是有管理级，第 3 级是可创建级，第 4 级是可预测级，第 5 级是优化级。

8. BOOTSTRAP

软件过程评估和改进方法 BOOTSTRAP 是由欧共体的 Esprit 项目下的多国工业和研究联合团体的 Bootstrap Institute 开发的，是建立在 CMU/SEI 的工作以及 ISO 9000 和欧洲空间署的软件工程标准的基础上发展的。1992 年底发表第一版的评估指导（问卷、支持材料、获取数据和评分工具）。欧洲质量管理基金会的整体质量模型（Total Quality Model）的一些思想也纳入到 1995 年公布的 BOOTSTRAP V2.3。目前已经积累了大量的经验。

BOOTSTRAP 评估的主要目标是生成一个改进行动计划。BOOTSTRAP 问卷中把软件生产单位/项目的过程质量属性分成三大组：组织、方法和技术。BOOTSTRAP 对每个级别和每个质量属性评分（0-无，1-及格，2-中，3-良，4-优），试图真实地标志一个组织或一个

项目的行为。

9. SPICE

SPICE 的全名是软件过程改进和能力确定(Software Process Improvement Capability dEtermination),它和软件过程评估 SPA 一同起着类似于 CMM 的作用。由于存在众多的软件过程改进方法和标准,1991 年英国建议 ISO/IEC 为软件过程管理建立了一套国际标准,建立软件过程评估标准,以调和现存各种标准。1992 年 ISO/IEC 批准成立工作组 WG10 开发软件过程评估的国际标准,并创建 SPICE 项目。SPICE 的目标是建立一种过程能力的度量方法。选用的方法是为度量特定过程的实现和使之制度化,作为一种过程度量,而不是组织度量。1994 年第一次完成实践指南基线(Baseli Practices Guide,BPG),1995 年批准多部分标准 ISO/IEC 15504 第一版。

10. TickIT

TickIT 方案是在 1991 年为实现以客观、独立的控制证据保证软件质量的目的而展开的,该方案专为涉及软件开发的组织提供普遍认可的 ISO 9000 认证。它的目标是专门针对软件开发提供 ISO 9000 质量保证。

3.4.6 软件质量保证

1. 软件质量保证概述

软件质量保证(SQA)是指建立一套有计划、有系统的方法来向管理层保证拟定出的标准、步骤、实践和方法能够正确地被所有项目所采用。软件质量保证的目的是使软件过程对于管理人员来说是可见的。它通过对软件产品和活动进行评审和审计来验证软件是合乎标准的。软件质量保证组在项目开始时就一起参与建立计划、标准和过程。这些将使软件项目满足机构方针的要求。

软件质量保证的目标:使工作有计划进行;客观地验证软件项目产品和工作是否遵循恰当的标准、步骤和需求;将软件质量保证工作及结果通知给相关组织和个人;高级管理层接触到在项目内部不能解决的不符合类问题。

2. SQA 的工作内容和工作方法

1) 计划

针对具体项目制定 SQA 计划,确保项目组正确执行过程。制定 SQA 计划应当注意如下几点:

(1) 有重点:依据企业目标以及项目情况确定审计的重点。

(2) 明确审计内容:明确审计哪些活动、哪些产品。

(3) 明确审计方式:确定怎样进行审计。

(4) 明确审计结果报告的规则:审计的结果报告给谁。

2) 审计/证实

依据 SQA 计划进行 SQA 审计工作,按照规则发布审计结果报告。

注意审计一定要有项目组人员陪同，不能搞突然袭击；双方要开诚布公，坦诚相对；审计的内容是否按照过程要求执行了相应活动，是否按照过程要求产生了相应产品。

3）问题跟踪

对审计中发现的问题，要求项目组改进，并跟进直到解决。

3. SQA 的素质

（1）过程为中心：应当站在过程的角度来考虑问题，只要保证了过程，SQA 就尽到了责任。

（2）服务精神：为项目组服务，帮助项目组确保正确执行过程。

（3）了解过程：深刻了解企业的工程，并具有一定的过程管理理论知识。

（4）了解开发：对开发工作的基本情况了解，能够理解项目的活动。

（5）沟通技巧：善于沟通，能够营造良好的气氛，避免审计活动成为一种找茬活动。

4. SQA 活动

SQA 是一种应用于整个软件过程的活动，它包含：一种质量管理方法；有效的软件工程技术（方法和工具）；在整个软件过程中采用的正式技术评审；一种多层次的测试策略；对软件文档及其修改的控制；保证软件遵从软件开发标准；度量和报告机制。

SQA 与两种不同的参与者相关，这两种参与者是做技术工作的软件工程师和负责质量保证的计划、监督、记录、分析及报告工作的 SQA 小组。

软件工程师通过采用可靠的技术方法和措施，进行正式的技术评审，执行计划周密的软件测试来考虑质量问题，并完成软件质量保证和质量控制活动。

SQA 小组的职责是辅助软件工程小组得到高质量的最终产品。SQA 小组完成的工作如下：

（1）为项目准备 SQA 计划。该计划在制定项目规定、项目计划时确定，由所有感兴趣的相关部门评审，包括：需要进行的审计和评审；项目可采用的标准；错误报告和跟踪的规程；由 SQA 小组产生的文档；向软件项目组提供的反馈数量；等等。

（2）参与开发项目的软件过程描述。评审过程描述以保证该过程与组织政策、内部软件标准、外界标准以及项目计划的其他部分相符。

（3）评审各项软件工程活动，对其是否符合定义好的软件过程进行核实。记录、跟踪与过程的偏差。

（4）审计指定的软件工作产品，对其是否符合事先定义好的需求进行核实。对产品进行评审，识别、记录和跟踪出现的偏差；对是否已经改正进行核实；定期将工作结果向项目管理者报告。

（5）确保软件工作及产品中的偏差已记录在案，并根据预定的规程进行处理。

（6）记录所有不符合的部分并报告给高级领导者。

5. 正式技术评审（FTR）

正式技术评审（FTR）是一种由软件工程师和其他人进行的软件质量保障活动。

1）目标

（1）发现功能、逻辑或实现的错误。

（2）证实经过评审的软件的确满足需求。

（3）保证软件的表示符合预定义的标准。

（4）得到一种一致的方式开发的软件。

（5）使项目更易管理。

2）评审会议

3～5人参加，不超过2小时，由评审主席、评审者和生产者参加，必须做出下列决定中的一个：工作产品可不可以不经修改而被接受；由于严重错误而否决工作产品；暂时接受工作产品。

3）评审总结报告、回答

评审什么？由谁评审？结论是什么？

评审总结报告是项目历史记录的一部分，标识产品中存在问题的区域，作为行政条目检查表以指导生产者进行改正。

4）评审指导原则

（1）评审产品，而不是评审生产者。注意客气地指出错误，气氛轻松。

（2）不要离题，限制争论。有异议的问题不要争论但要记录在案。

（3）对各个问题都发表见解。问题解决应该放到评审会议之后进行。

（4）为每个要评审的工作产品建立一个检查表。应为分析、设计、编码、测试文档都建立检查表。

（5）分配资源和时间。应该将评审作为软件工程任务加以调度。

（6）评审以前所做的评审。

6．检验项目内容

1）需求分析

需求分析→功能设计→实施计划。

检查：开发目的；目标值；开发量；所需资源；各阶段的产品作业内容及开发体制的合理性。

2）设计

结构设计→数据设计→过程设计。

检查：产品的计划量与实际量；评审量；差错数；评审方法；出错导因及处理情况；阶段结束的判断标准。

3）实现

程序编制→单元测试→集成测试→确认测试。检查内容除上述外，加测试环境及测试用例设计方法。

4）验收

说明书检查；程序检查。

习题 3

1. 名词解释

(1) PSP。
(2) 甘特图。
(3) 成本管理。
(4) TSP。
(5) SQA。
(6) FPA。

2. 判断题

(1) 代码复查就是从头到尾阅读源代码，并从中发现错误。（　　）
(2) 项目组织机构是项目型组织，是指那些一切工作都围绕项目进行、通过项目创造价值并达成自身战略目标的组织。（　　）
(3) 进度计划是表示各项工程的实施方式、成本核算以及调度安排的计划。（　　）
(4) 甘特图是由两条 S 型曲线组合成的闭合曲线，其计划实施过程中进行时间与累计完成任务量的关系都可以用一条 S 型曲线表示。（　　）
(5) 成本估算是项目成本管理的核心，通过成本估算，分析并确定项目的估算成本，并以此为基础进行项目成本预算，开展项目成本控制等管理活动。（　　）
(6) ISO 生命周期分为 4 个阶段，即需求、设计、实现和测试。（　　）

3. 填空题

(1) 进度控制的 4 个步骤包括_____、_____、_____、_____。
(2) 软件项目工作量估算的方法包括_____、_____、_____、_____。（填 4 个即可）

4. 选择题（多选）

(1) 软件过程质量的基本度量元有哪些？（　　）
A. 设计工作量应大于编码工作量
B. 设计评审工作量在设计工作量当中要少于四分之一
C. 代码评审工作量应占一半以上的代码编制的工作量
D. 每万行源程序在编译阶段发现的差错不应超过 10 个
(2) 项目组织机构的类型包括以下几种？（　　）
A. 集成团队组织　　B. 垂直团队组织
C. 水平团队组织　　D. 混合团队组织
(3) 成本管理的基本原则有哪些？（　　）
A. 合理化原则　　B. 全面管理的原则

C. 责任制原则　　D. 管理有效原则

(4) 工程项目进度计划的实施中,控制循环过程包括几项?(　　)

A. 事前进度控制　　B. 项目进度控制

C. 过程进度控制　　D. 事后进度控制

5. 简答题

(1) 进度控制的目标和范围是什么?

(2) 简单介绍 COCOMO 模型。

6. 论述题

(1) 进度控制的图形方法有哪几种? 请简单介绍一下。

(2) 请介绍一下软件开发项目中常用的几种成本估算方法。