第3章 电子商务系统开发管理

本章重点介绍电子商务系统开发团队，项目进度控制，项目成本估算与控制，软件质量管理。要求学生了解电子商务系统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 电子商务开发团队

本节重点介绍个人软件过程，团队软件过程，以及项目组构建。

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。软件能力成熟度模型 SW-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，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）。

个人软件过程是一种可用于控制、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，软件开发人员如能接受 PSP 培训，对软件能力成熟度的升级是一个有力的保证。CMM 侧重于软件企业中有关软件过程的宏观管理，面向软件开发单位，PSP 则侧重于企业中有关软件过程的微观优化，面向软件开发人员。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的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，每一级别都包含更低一级别中的所有元素，并增加了新的元素。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。PSP 过程改进模型见图 3-1。

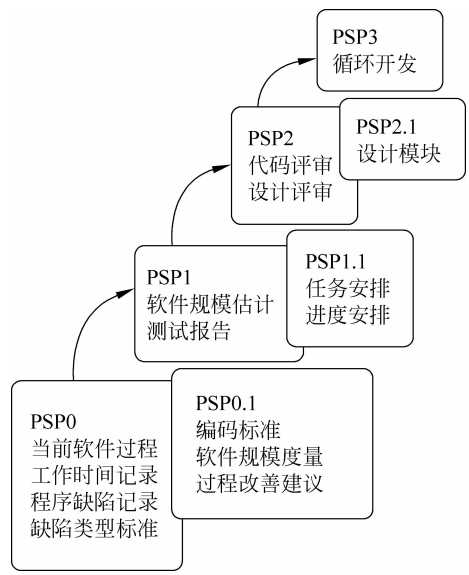

图 3-1　PSP 过程改进模型

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）以及后置处理三个阶段，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，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三个关键过程域，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

为了制订切实可行的计划，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，学习制订好的计划，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，首先制订时间分配计划，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是：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；周活动总结表；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）如何制订阶段计划

为了制订阶段计划，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）如何制订产品计划

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。

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是：搜寻缺陷征兆；从征兆中推断出缺陷的位置；确定程序中的错误；决定如何修复缺陷；修复缺陷；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，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）是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。加上 PSP 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。

团队软件过程是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 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 被设计为最多为 150 人左右的规模。团队软件过程(TSP)加上 PSP 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。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 3～20 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。如果你的组织运用 TSP，它会帮助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 团队软件过程

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)环境，并知晓如何一致创造高质量的产品。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 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，首先需要按照团队软件过程框架定义一个过程。在设计 TSP 时，需要遵循以下 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,开发小组成员如发现过程不合适,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、进展追踪和决策制定等项工作;

(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 8 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,每个成员都要独立相当一个角色。

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,称为开发小组素质度量元;另一组用以度量软件过程的质量,称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 组将制定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。通常无法制订超过三四个月的详细计划。所以，TSP 根据项目情况，每三四个月为一阶段，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 项目组的组建

1. 概述

项目组织机构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(见图 3-2)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。垂直团队的优点是：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。其缺点是：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。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在项目中确立。

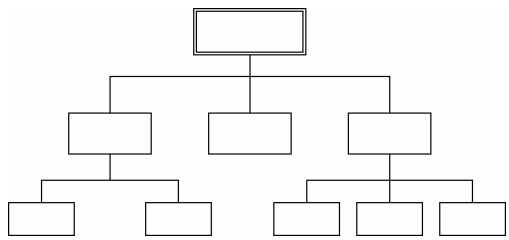

图 3-2　垂直团队组织

（2）水平团队组织。水平团队（见图 3-3）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。其优点是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。其缺点是：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。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。

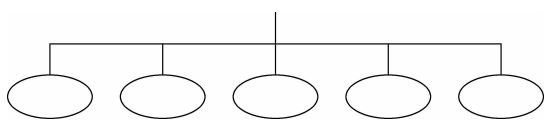

图 3-3　水平团队组织

（3）混合团队组织。混合团队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，支持并处理多个用例中各部分的专家们一起工作。其优点是：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。其缺点是：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。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，每个成员充当一或两个角色。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，同样，如果大多数人员是专家，则采用水平方案。

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)和测试人员。

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，如何实施进度控制。

3.2.1 项目进度概述

1. 概念

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。

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(Duration)内完成规定任务的情况。

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。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对(最低费用相对应的最优工期)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

工程项目进度计划的实施中，控制循环过程如下。

(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以下 4 个。

(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、节点时间要求的具体落实，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　进度控制的 4 个过程

1. 进度控制过程的 4 个阶段

进度控制的 4 个步骤(PDCA)：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(见图 3-4)分别如下。

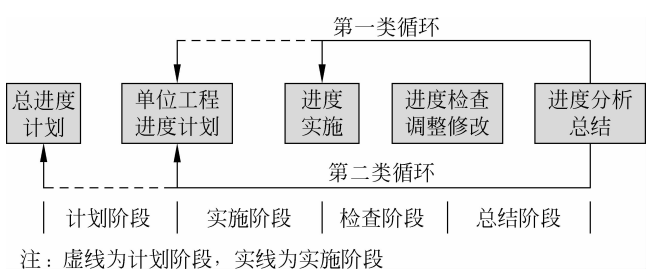

图 3-4　进度控制过程的 4 个阶段

(1) 编制进度计划；

(2) 实施进度计划；

(3) 检查与调整进度计划；

(4) 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，重点是安排工程实施的连续性。

1) 进度计划编制的目的

具体目的包括保证按时获利以补偿已经发生的费用支出，协调资源，使资源需要时可以利用，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，保证项目正常完成。

2) 进度计划编制的要求

具体要求包括：保证项目在合同规定的时间内完成，实现项目目标要求；实施进度安排必须满足连续性和均衡性要求；实施顺序的安排应进行优化，以便提高经济效益；应选择适当的计划图形，满足使用进度计划的要求；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：应对所有大事及其期限要求进行说明；确切的工作程序能够通过工作网络得以说明；进度应该与工作分解结构(Work Breakdown Structure，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

具体内容包括项目综合进度计划、设备(材料)采购工作进度计划、项目实施(开发)进度计划、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：将进度计划具体化为实施作业计划和实施任务书；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；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

具体实施包括：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；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；记录必须实事求是，不得造假；流水计划，在计划流水之下绘制实际进度线条；网络计划，记录实际持续时间；在计划图上用彩色标明已完成部分；用切割线记录；等等。

3）做好调度工作

具体工作包括：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；调度工作的内容包括检查计划执行中的问题，找出原因，提出解决措施；督促供应商按进度计划要求供应资源；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；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，准确、灵活、果断，确保工作效率；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

具体方法包括：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；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；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的事情发生引起；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、进度和预算可能与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

有利于尽快发挥投资效益，有利于维护良好的管理秩序，有利于提高企业经济效益，有利于降低信息系统工程的投资风险。

2）进度控制的目标

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。

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

纵向：在工程建设的各个阶段，对项目建设的全过程控制。

横向：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包括工程质量的影响、设计变更的影响、资源投入的影响、资金的影响、相关单位的影响、可见或不可见的风险因素的影响和承建单位管理水平的影响。

2. 进度控制的任务、程序与方法措施

各阶段主要任务如下。

（1）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）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）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）验收阶段。项目验收，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），见图 3-5。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。基本是一条线条图，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。

2）工程进度曲线（“香蕉”曲线图）

“香蕉”形曲线是两条 S 形曲线组合成的闭合曲线，从 S 形曲线比较法中得知，按某一时

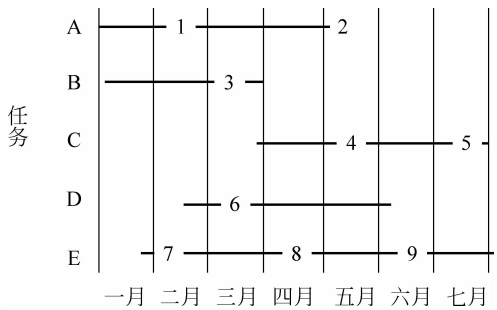

图 3-5　甘特图

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 形曲线表示。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：其一是计划以各项工作的最早开始时间安排进度而绘制的 S 形曲线，称为 ES 曲线；其二是计划以各项工作的最迟开始时间安排进度而绘制的 S 形曲线，称为 LS 曲线。两条 S 形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”形曲线。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”形曲线的区域内，见图 3-6。

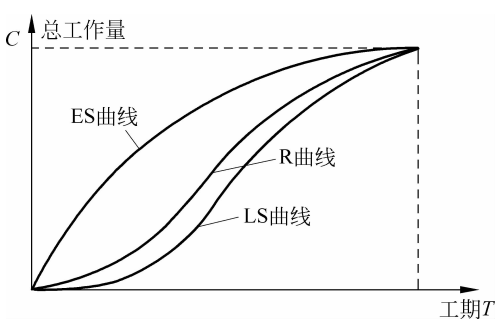

图 3-6　“香蕉”曲线图

“香蕉”形曲线比较法的作用：利用“香蕉”形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，见图 3-7。

（2）双代号网络图。

双代号网络图是应用较为广泛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，见图 3-8。

箭线：在双代号网络中，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

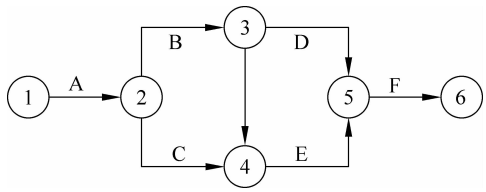

图 3-7 单代号网络图

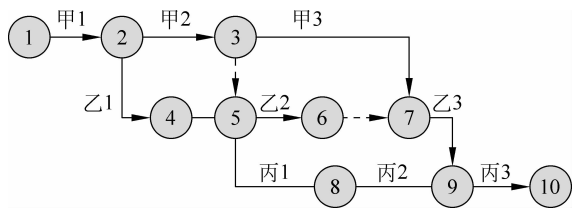

图 3-8 双代号网络图

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，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，接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：即既有内向箭线又有外向箭线的节点。

线路：即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，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存周期。

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，混合估算法，组合估算法。

1）自上向下估算方法

自上向下估算，又称为类比估算法，是一种自上向下的估算形式。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。该方法较其他方法更节省，但不是很精确，需要项目经理的水平和经验较高。

2）自下而上估算方法

自下而上估算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

混合估算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）参数方法

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）组合模型

组合模型是目前企业软件开发过程中常用的软件成本估算方式，它是一种自下而上和参数法的结合模型，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＝$E_1+E_2+\cdots+E_i+\cdots+E_n$。

（4）计算估算成本＝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、相关补助费用及其他等。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＝估算成本＋风险基金＋税收。其中：风险基金＝估算成本×a%（一般情况下 a 为 10～20 左右），税收＝估算成本×b%（一般情况下 b 为 5 左右）。

3.3.2　工作量估算

1. 代码行估算法

1）概念和计算方法

代码行（Line Of Code，LOC）。分析方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，一般为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

代码行分析方法从某种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，用代码行分析方法得到数值的意义已经大大降低。

尽管代码行分析方法有很多缺点，但由于它容易使用，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 于 20 世纪 70 年代由 IBM 的工程师艾伦·艾尔布策（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”、“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作为国际标准，如加拿大人艾伦·艾布恩（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，这些方法都属于艾尔布策功能点方法的发展和细化。

功能点分析方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 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。

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。

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，输出是指报表、屏幕、出错信息，等等。一个报表中的单个数据项不单独计算。

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。

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)。

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

较好的工作分解可以：防止遗漏项目的可交付成果；帮助项目经理关注项目目标和澄清职责；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；帮助改进时间、成本和资源估计的准确度；帮助项目团队的建立和获得项目人员的承诺；为绩效测量和项目控制定义一个基准；辅助沟通清晰的工作责任；为其他项目计划的制定建立框架；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。也就是覆盖所有的需要，也不做多余的工作。

(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

第一步，自上而下与自下而上的充分沟通；第二步，一对一个别交流；第三步，小组讨论。

4）分解的标准

分解时应注意：分解后的活动结构清晰；逻辑上形成一个大的活动；集成了所有的关键因素；包含临时的里程碑和监控点；所有活动全部定义清楚；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组，它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，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。但是，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。

类比估算法（Analogous Estimates）的操作步骤是：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三种时间估计。

（1）最可能的活动时间：正常情况下，完成某项工作的时间（Tm）。

（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（To）。

（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（Tp）。

活动期望时间（Te）的计算公式：$Te=\dfrac{To+4\times Tm+Tp}{6}$

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

例 3-2　表 3-1 列出了一个包含三个活动的某种路径的三种时间估计。

表 3-1　三个活动的三种时间估计

	第一段	第二段	第三段
To	4	1	2
Tm	7	7	11
Tp	16	25	26

$$Te=\frac{4+4\times7+16}{6}+\frac{1+4\times7+25}{6}+\frac{2+4\times11+26}{6}=8+9+12=29$$

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

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存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。

$$L=\mathrm{Ck}\times K^{1/3}\times \mathrm{td}^{4/3} \tag{3-1}$$

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)，td 为开发持续时间(以年计)，Ck 为技术状态常数，它反映“妨碍开发进展的限制”，取值因开发环境而异，见表 3-2。

表 3-2　不同开发环境下 Ck 的取值

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
11 000	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、易于使用的成本估算方法。1981 年由勃姆(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。每个参数可以定为很低，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，则算作一条指令。

(2) MM(度量单位为人月)表示开发工作量。

(3) TDEV(度量单位为月)表示开发进度，由工作量决定。

(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) 三种模型

COCOMO模型用三个不同层次的模型来反映不同程度的复杂性，分别如下。

(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)，则在用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) 三种模式

同时根据不同应用软件的不同应用领域，COCOMO模型划分为如下三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。

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。资源规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，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。应对成本费用进行层层分解。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，同时要确保所有的有关变更都准确地

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,这三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,它的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,存在一些问题等。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业主、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。成本控制的常用工具和技术有如下几种。

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系，对项目成本进行控制。该系统主要包括三个部分：成本变更申请、标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业主、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业主、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

绩效测量主要用于估算实际发生的变化的方法，如挣值分析法等。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，即使功用差异为正值，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 信息系统质量管理

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，质量保证。

3.4.1 产品与服务的质量

1. 质量定义

在不同的时期，国际标准 ISO 对质量的定义也不同。

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 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，大体经历了以下 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，美国贝尔实验室的 W. A. 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(Statistical Quality Control，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 W. Ewards Deming(埃华茨丹明)，Joseph Juran(约瑟夫佐兰)等人提出了全面质量管理(Total Quality Control，TQC)的概念。Deming 被誉为现优质量思想之父。

3. 顾客满意度

1) 顾客满意的定义

在 2000 年新版的 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 6 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，见图 3-9。

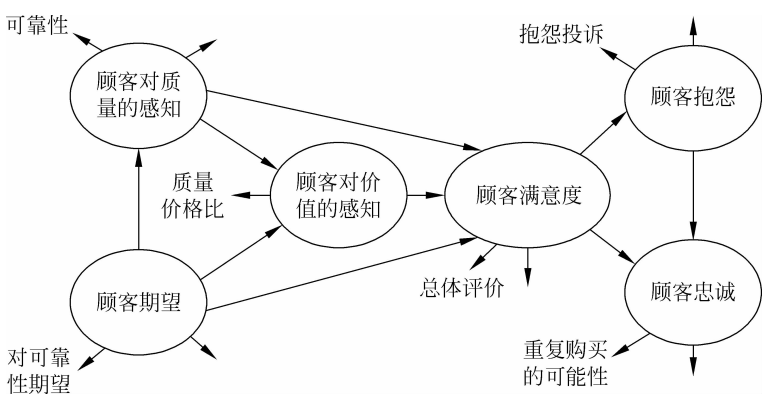

图 3-9　顾客满意度指数模型

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 大尺度、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 软件质量的概念

1. 软件质量定义

在中华人民共和国国家标准 GB/T 11457—1989 中，软件质量 Software Quality 的条目为 2.434，它的定义如下。

(1) 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。

(2) 软件具有所期望的各种属性的组合程度。

(3) 顾客和用户觉得软件满足其综合期望的程度。

(4) 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。

在中华人民共和国国家标准 GB/T 11457—1995 中，软件质量 Software Quality 的条目为 2.454，它的定义如下。

(1) 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。

(2) 软件具有所期望的各种属性的组合程度。

(3) 顾客和用户觉得软件满足其综合期望的程度。

(4) 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—1991 标准，软件质量有 6 个特性：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：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—1991 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：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

美国专家斯蒂芬（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：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 15504、Bootstrap、SPICE、TickIT 等。预计 21 世纪初，软件过程技术将得到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)

SW-CMM(软件生产能力成熟度模型)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 5 级——初始级，可重级，定义级，管理级和优化级。关键过程域(Key Process Area，KPA)包含 5 类目标：实施保证，实施能力，执行活动，度量分析和实施验证。

2. 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，委托美国卡内基梅隆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。

3. 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，表明他具备了质量保证能力。制定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，它们是管理职责，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。

4. 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《软件生存在过程》标准草案，我国根据该草案制定了 GB/T 8566—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：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。

5. 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 15504 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 标准是一个两维的结构：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，组织过程；另一个是能力维，从低到高有 6 个级别，第 0 级不完全级，第 1 级可实施级，第 2 级有管理级，第 3 级可创建级，第 4 级可预测级，第 5 级优化级。

6. 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。建立在 CMU 的 SEI 的工作，ISO 9000 和欧洲空间署的软件工程标准的基础上发展的。1992 年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 BootstrapV 2.3 中。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。

7. 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。

3.4.4 质量保证

1. 质量保证概述

软件质量保证(Software Quality Assurance,SQA)是建立一套有计划,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别和个人;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,QA 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

软件质量保证(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——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 小组完成以下任务。

(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

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，不超过两小时，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。

3.5 信息系统开发方法

本节重点介绍生命周期法，结构化方法，面向对象方法，以及软件复用和构件技术。

3.5.1 生命周期法

1. 定义

同任何事物一样，一个软件产品或软件系统也要经历孕育、诞生、成长、成熟、衰亡等阶段，一般称为软件生存周期(软件生命周期)。把整个软件生存周期划分为若干阶段，使得每个阶段有明确的任务，使规模大，结构复杂和管理复杂的软件开发变得容易控制和管理。通常，软件生存周期包括可行性分析与开发项计划、需求分析、设计(概要设计和详细设计)、编码、测试、维护等活动，可以将这些活动以适当的方式分配到不同的阶段去完成。

软件生命周期(Systems Development Life Cycle，SDLC)是软件的产生直到报废的生命周期，周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段，这种按时间分层的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。但随着新的面向对象的设计方法和技术的成熟，软件生命周期设计方法的指导意义正在逐步减少。

2. 软件生命周期的几个阶段

1）问题的定义及规划

此阶段是软件开发方与需求方共同讨论，主要确定软件的开发目标及其可行性。

2）需求分析

在确定软件开发可行的情况下，对软件需要实现的各个功能进行详细分析。需求分析阶段是一个很重要的阶段，这一阶段做得好，将为整个软件开发项目的成功打下良好的基础。“唯一不变的是变化本身”。同样，需求也是在整个软件开发过程中不断变化和深入的，因此必须制定需求变更计划来应付这种变化，以保护整个项目的顺利进行。

3）软件设计

此阶段主要根据需求分析的结果，对整个软件系统进行设计，如系统框架设计、数据库设计等。软件设计一般分为总体设计和详细设计。好的软件设计将为软件程序编写打下良好的基础。

4）程序编码

此阶段是将软件设计的结果转换成计算机可运行的程序代码。在程序编码中必须要制定统一、符合标准的编写规范，以保证程序的可读性、易维护性、提高程序的运行效率。

5）软件测试

在软件设计完成后要经过严密的测试，以发现软件在整个设计过程中存在的问题并加以纠正。整个测试过程分为单元测试、组装测试以及系统测试三个阶段进行。测试的方法主要有白盒测试和黑盒测试两种。在测试过程中需要建立详细的测试计划并严格按照测试计划进行测试，以减少测试的随意性。

6）运行维护

软件维护是软件生命周期中持续时间最长的阶段。在软件开发完成并投入使用后，由于多方面的原因，软件不能继续适应用户的要求。要延续软件的使用寿命，就必须对软件进行维护。软件的维护包括纠错性维护和改进性维护两个方面。

7）软件升级

软件升级是指软件开发者在编写软件的时候，由于设计人员考虑不全面或程序功能不完善，在软件发行后，通过对程序的修改或加入新的功能后，以补丁的形式发布的方式。用户把这些补丁更新，即升级完成。软件升级是为了更好地满足用户的需求和防止病毒的入侵。

8）软件报废

软件因不能继续使用或功能不能满足现在的需求而作废。

3. 软件开发流程

软件开发流程(Software Development Process)即软件设计思路和方法的一般过程，包括设计软件的功能和实现的算法和方法、软件的总体结构设计和模块设计、编程和调试、程序联调和测试以及编写、提交程序。

第一步：需求调研分析

系统分析员向用户初步了解需求，然后列出要开发的系统的大功能模块，每个大功能模

块有哪些小功能模块，对于有些需求比较明确相关的界面时，在这一步里面可以初步定义好少量的界面。

系统分析员深入了解和分析需求，根据自己的经验和需求用相关的工具再做出一份文档系统的功能需求文档。这次的文档会清楚利用系统大致的大功能模块，大功能模块有哪些小功能模块，并且还列出相关的界面和界面功能。

系统分析员向用户再次确认需求。

第二步：概要设计

开发者需要对软件系统进行概要设计，即系统设计。概要设计需要对软件系统的设计进行考虑，包括系统的基本处理流程、系统的组织结构、模块划分、功能分配、接口设计、运行设计、数据结构设计和出错处理设计等，为软件的详细设计提供基础。

第三步：详细设计

在概要设计的基础上，开发者需要进行软件系统的详细设计。在详细设计中，描述实现具体模块所涉及的主要算法、数据结构、类的层次结构及调用关系，需要说明软件系统各个层次中的每一个程序（每个模块或子程序）的设计考虑，以便进行编码和测试。应当保证软件的需求完全分配给整个软件。详细设计应当足够详细，能够根据详细设计报告进行编码。

第四步：编码

在软件编码阶段，开发者根据《软件系统详细设计报告》中对数据结构、算法分析和模块实现等方面的设计要求，开始具体的编写程序工作，分别实现各模块的功能，从而实现对目标系统的功能、性能、接口、界面等方面的要求。

第五步：测试

测试编写好的系统。交给用户使用，用户使用后一个一个地确认每个功能。

第六步：软件交付准备

在软件测试证明软件达到要求后，软件开发者应向用户提交开发的目标安装程序、数据库的数据字典、《用户安装手册》、《用户使用指南》、需求报告、设计报告、测试报告等双方合同约定的产物。

第七步：验收

对软件进行功能项测试，业务流测试，容错测试，安全性测试，性能测试，易用性测试，适应性测试，文档测试等。

3.5.2 结构化方法

1. 结构化方法的定义

结构化方法是一种传统的软件开发方法，它是由结构化分析、结构化设计和结构化程序设计三部分有机组合而成的。它的基本思想：把一个复杂问题的求解过程分阶段进行，而且这种分解是自顶向下，逐层分解，使得每个阶段处理的问题都控制在人们容易理解和处理的范围内。

结构化方法的基本要点是：自顶向下、逐步求精、模块化设计。结构化分析方法是以自顶向下，逐步求精为基点，以一系列经过实践的考验被认为是正确的原理和技术为支撑，以

数据流图、数据字典、结构化语言、判定表、判定树等图形表达为主要手段，强调开发方法的结构合理性和系统的结构合理性的软件分析方法。

结构化方法按软件生命周期划分，有结构化分析(SA)、结构化设计(SD)、结构化实现(SP)。其中要强调的是，结构化方法学是一个思想准则的体系，虽然有明确的阶段和步骤，但是也集成了很多原则性的东西，所以学会结构化方法，不是单从理论知识上去了解就足够的，更多的还是要在实践中慢慢理解各准则，慢慢将其变成自己的方法学。

2. 结构化分析

结构化分析是20世纪70年代末，由Demarco等人提出的，旨在减少分析活动中的错误，建立满足用户需求的系统逻辑模型。该方法的要点是：面对数据流的分解和抽象；把复杂问题自顶向下逐层分解，经过一系列分解和抽象，到最底层的就都是很容易描述并实现的问题了。SA方法的分析结果由数据流图、数据词典和加工逻辑说明。

结构化分析就是使用数据流程图、数据字典、结构化语言、判定表和判定树等工具，来建立一种新的、称为结构化说明书的目标文档，也就是需求规格说明书。

结构化分析的步骤：①分析当前的情况，做出反映当前物理模型的DFD；②推导出等价的逻辑模型的DFD；③设计新的逻辑系统，生成数据字典和基元描述；④建立人机接口，提出可供选择的目标系统物理模型的DFD；⑤确定各种方案的成本和风险等级，据此对各种方案进行分析；⑥选择一种方案；⑦建立完整的需求规约。

3. 结构化设计

结构化设计方法给出一组帮助设计人员在模块层次上区分设计质量的原理与技术。它通常与结构化分析方法衔接起来使用，以数据流图为基础得到软件的模块结构。SD方法尤其适用于变换型结构和事务型结构的目标系统。在设计过程中，它从整个程序的结构出发，利用模块结构图表述程序模块之间的关系。结构化设计的结果是概要设计说明书和详细设计说明书。

结构化设计方法的设计原则：使每个模块尽量只执行一个功能(坚持功能性内聚)；每个模块用过程语句(或函数方式等)调用其他模块；模块间传送的参数作数据用；模块间共用的信息(如参数等)尽量少。

结构化设计的步骤：①评审和细化数据流图；②确定数据流图的类型；③把数据流图映射到软件模块结构，设计出模块结构的上层；④基于数据流图逐步分解高层模块，设计中下层模块；⑤对模块结构进行优化，得到更为合理的软件结构；⑥描述模块接口。

4. 结构化系统实现

结构化系统实现是系统开发工作的最后一个阶段。它是将结构化系统设计的成果变成可实际运行的系统的过程。系统实现的主要工作包括：数据库的建立，应用程序设计与编码，程序测试与系统调试，试运行，现场布局调整与系统移入，组织机构调整，系统切换、文档整理与验收(鉴定)。实现阶段形成的文档主要有：数据库源模式清单，程序流程图及源程序清单，系统调试书，使用说明书，维护手册，系统验收(鉴定、评审)书等。

3.5.3 面向对象方法

1. 面向对象方法概述

面向对象方法(Object-Oriented Method)是一种把面向对象的思想应用于软件开发过程中,指导开发活动的系统方法,简称 OO(Object-Oriented)方法,是建立在“对象”概念基础上的方法学。对象是由数据和容许的操作组成的封装体,与客观实体有直接对应关系,一个对象类定义了具有相似性质的一组对象。而继承性是对具有层次关系的类的属性和操作进行共享的一种方式。所谓面向对象就是基于对象概念,以对象为中心,以类和继承为构造机制,来认识、理解、刻画客观世界和设计、构建相应的软件系统。

用计算机解决问题需要用程序设计语言对问题求解加以描述(即编程),实质上,软件是问题求解的一种表述形式。显然,假如软件能直接表现人求解问题的思维路径(即求解问题的方法),那么软件不仅容易被人理解,而且易于维护和修改,从而会保证软件的可靠性和可维护性,并能提高公共问题域中的软件模块和模块重用的可靠性。面向对象的机能和机制恰好可以使得人们按照通常的思维方式来建立问题域的模型,设计出尽可能自然地表现求解方法的软件。

面向对象方法作为一种新型的独具优越性的新方法正引起全世界越来越广泛的关注和高度的重视,它被誉为“研究高技术的好方法”,更是当前计算机界关心的重点。十多年来,在对 OO 方法如火如荼的研究热潮中,许多专家和学者预言:正像 20 世纪 70 年代结构化方法对计算机技术应用所产生的巨大影响和促进那样,20 世纪 90 年代 OO 方法会强烈地影响、推动和促进一系列高技术的发展和多学科的综合。

2. 由来与发展

OO 方法起源于面向对象的编程语言(Object-Oriented Programming Language,OOPL)。20 世纪 50 年代后期,在用 FORTRAN 语言编写大型程序时,常出现变量名在程序不同部分发生冲突的问题。鉴于此,ALGOL 语言的设计者在 ALGOL60 中采用了以“Begin…End”为标识的程序块,使块内变量名是局部的,以避免它们与程序中块外的同名变量相冲突。这是编程语言中首次提供封装(保护)的尝试。此后程序块结构广泛用于高级语言如 Pascal、Ada、C 语言之中。

20 世纪 60 年代中后期,Simula 语言在 ALGOL 基础上研制开发,它将 ALGOL 的块结构概念向前发展一步,提出了对象的概念,并使用了类,也支持类继承。20 世纪 70 年代,Smalltalk 语言诞生,它取 Simula 的类为核心概念,它的很多内容借鉴于 Lisp 语言。由 Xerox 公司经过对 Smalltalk72、76 持续不断的研究和改进之后,于 1980 年推出商品化,它在系统设计中强调对象概念的统一,引入对象、对象类、方法、实例等概念和术语,采用动态联编和单继承机制。从 20 世纪 80 年代起,人们基于以往已提出的有关信息隐蔽和抽象数据类型等概念,以及由 Modula2、Ada 和 Smalltalk 等语言所奠定的基础,再加上客观需求的推动,进行了大量的理论研究和实践探索。由此,不同类型的面向对象语言(如 Object-C、Eiffel、C++、Java、Object-Pascal 等)逐步地发展和建立起较完整的和如雨后春笋般研制开发出来,包括 OO 方法的概念理论体系和实用的软件系统。

面向对象源出于 Simula，真正的 OOP 由 Smalltalk 奠基。Smalltalk 现在被认为是最纯的 OOPL。正是通过 Smalltalk80 的研制与推广应用，人们注意到 OO 方法所具有的模块化、信息封装与隐蔽、抽象性、继承性、多样性等独特之处，这些优异特性为研制大型软件、提高软件可靠性、可重用性、可扩充性和可维护性提供了有效的手段和途径。20 世纪 80 年代以来，人们将面向对象的基本概念和运行机制运用到其他领域，获得了一系列相应领域的面向对象的技术。面向对象方法在许多领域的应用都得到了很大的发展，已被广泛应用于程序设计语言、形式定义、设计方法学、操作系统、分布式系统、人工智能、实时系统、数据库、人机接口、计算机体系结构以及并发工程、综合集成工程等。1986 年，在美国举行了首届“面向对象编程、系统、语言和应用(OOPSLA'86)”国际会议，使面向对象受到世人瞩目，其后每年都举行一次，这进一步标志 OO 方法的研究已遍及全世界。

3. 面向对象的基本概念

(1) 对象，是要研究的任何事物。从一辆车到一个车库，从一条信息到一个数据库，甚至航天飞机或空间站都可看作对象，它不仅能表示有形的实体，也能表示无形的(抽象的)规则、计划或事件。对象由数据(描述事物的属性)和作用于数据的操作(体现事物的行为)构成一独立整体。从程序设计者来看，对象是一个程序模块，从用户来看，对象为他们提供所希望的行为。对内的操作通常称为方法。一个对象请求另一对象为其服务的方式是通过发送消息。

(2) 类，是对象的模板。即类是对一组有相同数据和相同操作的对象的定义，一个类所包含的方法和数据描述一组对象的共同属性和行为。类是在对象之上的抽象，对象则是类的具体化，是类的实例。类可有其他类，也可有其他类，形成类层次结构。

(3) 消息，是对象之间进行通信的一种规格说明。一般它由三部分组成：接收消息的对象、消息名及实际变元。

(4) 封装，是一种信息隐蔽技术，它体现于类的说明，是对象的重要特性。封装使数据和加工该数据的方法(函数)封装为一个整体，以实现独立性很强的模块，使得用户只能见到对象的外特性(对象能接收哪些消息，具有哪些处理能力)，而对象的内特性(保存内部状态的私有数据和实现加工能力的算法)对用户是隐蔽的。封装的目的在于把对象的设计者和对象的使用者分开，使用者不必知晓行为实现的细节，只须用设计者提供的消息来访问该对象。

(5) 继承性，是子类自动共享父类之间数据和方法的机制。它由类的派生功能体现。一个类直接继承其他类的全部描述，同时可修改和扩充。继承具有传递性。继承分为单继承(一个子类只有一个父类)和多重继承(一个类有多个父类)。类的对象是各自封闭的，如果没有继承性机制，则类对象中数据、方法就会出现大量重复。继承不仅支持系统的可重用性，而且还促进系统的可扩充性。

(6) 多态性。对象根据所接收的消息而做出动作。同一消息为不同的对象接收时可产生完全不同的行动，这种现象称为多态性。利用多态性用户可发送一个通用的信息，而将所有的实现细节都留给接收消息的对象自行决定，如此，同一消息即可调用不同的方法。例如，Print 消息被发送给一图或表时调用的打印方法与将同样的 Print 消息发送给一正文文件而调用的打印方法会完全不同。多态性的实现受到继承性的支持，利用类继承的层次关

系，把具有通用功能的协议存放在类层次中尽可能高的地方，而将实现这一功能的不同方法置于较低层次，这样，在这些低层次上生成的对象就能给通用消息以不同的响应。在 OOPL 中可通过在派生类中重定义基类函数（定义为重载函数或虚函数）来实现多态性。

4. 面向对象技术

OO 方法是程序设计新范型、系统开发的新方法学。作为一门新技术，OO 方法可支持种类不同的系统开发，已经或正在许多方面得以应用。

近十多年来，除了面向对象的程序设计以外，OO 方法已发展应用到整个信息系统领域和一些新兴的工业领域，包括：用户界面、应用集成平台、面向对象数据库（OODB）、分布式系统、网络管理结构、人工智能领域以及并发工程、综合集成工程等。人工智能是和计算机密切相关的新领域，在很多方面已经采用面向对象技术，如知识的表示、专家系统的建造、用户界面等。人工智能的软件通常规模较大，用面向对象技术有可能更好地设计并维护这类程序。

20 世纪 80 年代后期形成的并发工程，其概念要点是在产品开发初期（即方案设计阶段）就把结构、工艺、加工、装配、测试、使用、市场等问题同期并行地启动运行，其实现必须有两个基本条件：一是专家群体，二是共享并管理产品信息（将 CAD、CAE、CIN 紧密结合在一起）。显然，这需要面向对象技术的支持。目前，一些公司采用并发工程组织产品的开发，已取得显著效益：波音公司用以开发巨型 777 运输机，比开发 767 节省了一年半时间；日本把并发工程用于新型号的汽车生产，和美国相比只用一半的时间。产业界认为它们以后的生存要依靠并发工程，而面向对象技术是促进并发工程发展的重要支持。

综合集成工程是开发大型开放式复杂系统的新的工程概念。与并发工程相似，专家群体的组织和共享信息，是支持这一新工程概念的两大支柱。由于开放式大系统包含人的智能活动，建立数学模型非常困难，而 OO 方法能够比较自然地刻画现实世界，容易达到问题空间和程序空间的一致，能够在多种层次上支持复杂系统层次模型的建立，是研究综合集成工程的重要工具。面向对象技术对于并发工程和综合集成工程的作用，一方面说明了这一新技术应用范围的宽广，同时也说明了它的重要影响，更证明了面向对象技术是一门新兴的值得广泛重视的技术。

5. 面向对象方法的基本步骤

（1）分析确定在问题空间和解空间出现的全部对象及其属性；

（2）确定应施加于每个对象的操作，即对象固有的处理能力；

（3）分析对象间的联系，确定对象彼此间传递的消息；

（4）设计对象的消息模式，消息模式和处理能力共同构成对象的外部特性；

（5）分析各个对象的外部特性，将具有相同外部特性的对象归为一类，从而确定所需要的类；

（6）确定类间的继承关系，将各对象的公共性质放在较上层的类中描述，通过继承来共享对公共性质的描述；

（7）设计每个类关于对象外部特性的描述；

（8）设计每个类的内部实现（数据结构和方法）；

(9) 创建所需的对象(类的实例),实现对象间应有的联系(发消息)。

6. OOA 方法

面向对象的分析方法(Object-Oriented Analysis,OOA)是在一个系统的开发过程中进行了系统业务调查以后,按照面向对象的思想来分析问题。OOA 与结构化分析有较大的区别。OOA 所强调的是在系统调查资料的基础上,针对 OO 方法所需要的素材进行的归类分析和整理,而不是对管理业务现状和方法的分析。

1) 处理复杂问题的原则

用 OOA 方法对所调查结果进行分析处理时,一般依据以下几项原则:

(1) 抽象(Abstraction)是指为了某一分析目的而集中精力研究对象的某一性质,它可以忽略其他与此目的无关的部分。在使用这一概念时,要承认客观世界的复杂性,也知道事物包括多个细节,但此时并不打算去完整地考虑它。抽象是科学地研究和处理复杂问题的重要方法。抽象机制被用在数据分析方面,称为数据抽象。数据抽象是 OOA 的核心。数据抽象把一组数据对象以及作用其上的操作组成一个程序实体。使得外部只知道它是如何做和如何表示的。在应用数据抽象原理时,系统分析人员必须确定对象的属性以及处理这些属性的方法,并借助于方法获得属性。在 OOA 中属性和方法被认为是不可分割的整体。抽象机制有时也被用在对过程的分解方面,被称为过程抽象。恰当的过程抽象可以对复杂过程的分解和确定以及描述对象发挥积极的作用。

(2) 封装(Encapsulation),即信息隐蔽,指在确定系统的某一部分内容时,应考虑到其他部分的信息及联系都在这一部分的内部进行,外部各部分之间的信息联系应尽可能的少。继承(Inheritance)是指能直接获得已有的性质和特征而不必重复定义它们。OOA 可以一次性地指定对象的公共属性和方法,然后再特化和扩展这些属性及方法为特殊情况,这样可大大地减轻在系统实现过程中的重复劳动。在共有属性的基础之上,继承者也可以定义自己独有的特性。

(3) 相关(Association),指把某一时刻或相同环境下发生的事物联系在一起。

(4) 消息通信是指在对象之间互相传递信息的通信方式。

(5) 组织方法。在分析和认识世界时,可综合采用如下三种组织方法(Method of Organization):①特定对象与其属性之间的区别;②整体对象与相应组成部分对象之间的区别;③不同对象类的构成及其区别等。

(6) 比例(Scale)是一种运用整体与部分原则,辅助处理复杂问题的方法。

(7) 行为范畴(Categories of Behavior)是针对被分析对象而言的,它们主要包括:①基于直接原因的行为;②变性行为;③功能查询性行为。

2) OOA 方法的基本步骤

在用 OOA 具体地分析一个事物时,大致上遵循如下 5 个基本步骤。

第一步,确定对象和类。这里所说的对象是对数据及其处理方式的抽象,它反映了系统保存和处理现实世界中某些事物的信息的能力。类是多个对象的共同属性和方法集合的描述,包括如何在一个类中建立一个新对象的描述。

第二步,确定结构(Structure)。结构是指问题域的复杂性和连接关系。类成员结构反映了泛化-特化关系,整体-部分结构反映整体和局部之间的关系。

第三步，确定主题(Subject)。主题是指事物的总体概貌和总体分析模型。

第四步，确定属性(Attribute)。属性就是数据元素，可用来描述对象或分类结构的实例，可在图中给出，并在对象的存储中指定。

第五步，确定方法(Method)。方法是在收到消息后必须进行的一些处理方法。方法要在图中定义，并在对象的存储中指定。对于每个对象和结构来说，那些用来增加、修改、删除和选择一个方法本身都是隐含的(虽然它们是要在对象的存储中定义的，但并不在图上给出)，而有些则是显式的。

7. OOD 方法

面向对象的设计方法(Object-Oriented Design，OOD)是 OO 方法中一个中间过渡环节。其主要作用是对 OOA 分析的结果做进一步的规范化整理，以便能够被 OOP 直接接受。在 OOD 的设计过程中，要展开的主要有如下几项工作。

(1) 对象定义规格的求精过程。对于 OOA 所抽象出来的对象和类以及汇集的分析文档，OOD 需要有一个根据设计要求整理和求精的过程，使之更能符合面向对象编程(OOP)的需要。这个整理和求精过程主要有两个方面：一是要根据面向对象的概念模型整理分析所确定的对象结构、属性、方法等内容，改正错误的内容，删去不必要和重复的内容等。二是进行分类整理，以便于下一步数据库设计和程序处理模块设计的需要。整理的方法主要是进行归类，对类和对象、属性、方法和结构、主题进行归类。

(2) 数据模型和数据库设计。数据模型的设计需要确定类和对象属性的内容、消息连接的方式、系统访问、数据模型的方法等。最后每个对象实例的数据都必须落实到面向对象的库结构模型中。

8. OOP 方法

面向对象编程(Object-Oriented Programming，OOP)是一种计算机编程架构。OOP 的基本原则是程序由单个能够起到子程序作用的单元或对象组合而成。OOP 的基本思想是把组件的实现和接口分开，使组件具有多态性。OOP 具有重用性、灵活性和扩展性。为了实现整体运算，每个对象都能够接收信息、处理数据和向其他对象发送信息。OOP 的关键是组件。它是数据和功能一起在运行着的计算机程序中形成的单元，组件在 OOP 计算机程序中是模块和结构化的基础。

9. OOT 方法

OOT 是面向对象的测试(Object-Oriented Test)。OOA Test 和 OOD Test 是对分析结果和设计结果的测试，主要是对分析设计产生的文本进行，是软件开发前期的关键性测试。OOP Test 主要针对编程风格和程序代码实现进行测试，其主要的测试内容在面向对象单元测试和面向对象集成测试中体现。面向对象单元测试是对程序内部具体单一的功能模块的测试，如果程序是用 C++语言实现，主要就是对类成员函数的测试。面向对象单元测试是进行面向对象集成测试的基础。面向对象集成测试主要对系统内部的相互服务进行测试，如成员函数间的相互作用，类间的消息传递等。面向对象集成测试不但要基于面向对象单元测试，更要参见 OOD 或 OOD Test 结果(详见后面叙述)。面向对象系统测试是基于面

向对象集成测试的最后阶段的测试，主要以用户需求为测试标准，需要借鉴 OOA 或 OOA Test 结果。

3.5.4　软件复用和构件技术

1. 软件复用

1）复用定义

软件复用(Software Reuse)就是将已有的软件成分用于构造新的软件系统，以缩减软件开发和维护的花费。无论对可复用构件原封不动地使用还是做适当的修改后再使用，只要是用来构造新软件，则都可称作复用。被复用的软件成分一般称作可复用构件。软件复用是提高软件生产力和质量的一种重要技术。早期的软件复用主要是代码级复用，后来扩大到包括领域知识、开发经验、项目计划、可行性报告、体系结构、需求、设计、测试用例和文档等一切有关方面。对一个软件进行修改，使它运行于新的软硬件平台不称为复用，而称为软件移植。

软件复用的主要思想是，将软件看成是由不同功能部分的“组件”所组成的有机体，每一个组件在设计编写时可以被设计成完成同类工作的通用工具，这样，如果完成各种工作的组件被建立起来以后，编写一特定软件的工作就变成了将各种不同组件组织连接起来的简单问题，这对于软件产品的最终质量和维护工作都有本质性的改变。

2）复用级别

(1) 代码的复用。包括目标代码和源代码的复用。其中目标代码的复用级别最低，历史也最久，当前大部分编程语言的运行支持系统都提供了链接(Link)、绑定(Binding)等功能来支持这种复用。源代码的复用级别略高于目标代码的复用，程序员在编程时把一些想复用的代码段复制到自己的程序中，但这样往往会产生一些新旧代码不匹配的错误。想大规模地实现源程序的复用只有依靠含有大量可复用构件的构件库。如“对象链接及嵌入”技术，既支持在源程序级定义构件并用以构造新的系统，又使这些构件在目标代码的级别上仍然是一些独立的可复用构件，能够在运行时被灵活地更新组合为各种不同的应用。

(2) 设计的复用。设计结果比源程序的抽象级别更高，因此它的复用受实现环境的影响较少，从而使可复用构件被复用的机会更多，并且所需的修改更少。这种复用有三种途径，第一种途径是从现有系统的设计结果中提取一些可复用的设计构件，并把这些构件应用于新系统的设计；第二种途径是把一个现有系统的全部设计文档在新的软硬件平台上重新实现，也就是把一个设计运用于多个具体的实现；第三种途径是独立于任何具体的应用，有计划地开发一些可复用的设计构件。

(3) 分析的复用。这是比设计结果更高级别的复用，可复用的分析构件是针对问题域的某些事物或某些问题的抽象程度更高的解法，受设计技术及实现条件的影响很少，所以可复用的机会更大。复用的途径也有三种，即从现有系统的分析结果中提取可复用构件用于新系统的分析；用一份完整的分析文档作输入产生针对不同软硬件平台和其他实现条件的多项设计；独立于具体应用，专门开发一些可复用的分析构件。

(4) 测试信息的复用。主要包括测试用例的复用和测试过程信息的复用。前者是把一个软件的测试用例在新的软件测试中使用，或者在软件做出修改时在新的一轮测试中使用。

后者是在测试过程中通过软件工具自动地记录测试的过程信息，包括测试员的每一个操作、输入参数、测试用例及运行环境等一切信息。这种复用的级别，不便和分析、设计、编程的复用级别做准确的比较，因为被复用的不是同一事物的不同抽象层次，而是另一种信息，但从这些信息的形态看，大体处于与程序代码相当的级别。

3）OO 方法对软件复用的支持

支持软件复用是人们对面向对象方法寄托的主要希望之一，也是这种方法受到广泛重视的主要原因之一。面向对象方法之所以特别有利于软件复用，是由于它的主要概念及原则与软件复用的要求十分吻合。

面向对象方法从面向对象的编程发展到面向对象的分析与设计，使这种方法支持软件复用的固有特征能够从软件生命周期的前期阶段开始发挥作用，从而使 OO 方法对软件复用的支持达到了较高的级别。与其他软件工程方法相比，面向对象方法的一个重要优点是，它可以在整个软件生命周期达到概念、原则、术语及表示法的高度一致。这种一致性使得各个系统成分尽管在不同的开发与演化阶段有不同的形态，但可具有贯穿整个软件生命周期的良好映射。这一优点使 OO 方法不但能在各个级别支持软件复用，而且能对各个级别的复用形成统一的、高效的支持，达到良好的全局效果。做到这一点的必要条件是，从面向对象软件开发的前期阶段——OOA 就把支持软件复用作为一个重点问题来考虑。运用 OOA 方法所定义的对象类具有适合作为可复用构件的许多特征，OOA 结果对问题域的良好映射，使同类系统的开发者容易从问题出发，在已有的 OOA 结果中发现不同粒度的可复用构件。

2. 构件技术

1）定义

构件(Component)是系统中实际存在的可更换部分，它实现特定的功能，符合一套接口标准并实现一组接口。构件代表系统中的一部分物理实施，包括软件代码(源代码、二进制代码或可执行代码)或其等价物(如脚本或命令文件)。

构件是面向软件体系架构的可复用软件模块。构件是可复用的软件组成成分，可被用来构造其他软件。它可以是被封装的对象类、类树、一些功能模块、软件框架、软件构架(或体系结构)、文档、分析件、设计模式等。

1995 年，Ian. oraham 给出的构件定义如下：构件是指一个对象(接口规范、或二进制代码)，它被用于复用，接口被明确定义。构件是作为一个逻辑紧密的程序代码包的形式出现的，有着良好的接口。像 Ada 的 Package、Smalltalk80 和 C++的 Class 和数据类型都可属于构件范畴。但是，操作集合、过程、函数即使可以复用也不能成为一个构件。开发者可以通过组装已有的构件来开发新的应用系统，从而达到软件复用的目的。软件构件技术是软件复用的关键因素，也是软件复用技术研究的重点。

采用构件软件不需要重新编译，也不需要源代码并且不局限于某一种编程语言。该过程叫做二进制复用(Binary Reuse)，因为它是建立在接口而不是源代码级别的复用之上的。虽然软件构件必须遵守一致的接口，但是它们的内部实现是完全自动的。因此，可以用过程语言和面向对象语言创建构件。由于构件技术是由基于面向对象技术而发展起来的，与面向对象的设计中的对象相类似，它们都是针对软件复用，都是被封装的代码，但它们之间仍

存在很大差异。

2）软件构件的属性

（1）有用性（Usefulness）：构件必须提供有用的功能。

（2）可用性（Usability）：构件必须易于理解和使用。

（3）质量（Quality）：构件及其变形必须能正确工作。

（4）适应性（Adaptability）：构件应该易于通过参数化等方式在不同语境中进行配置。

（5）可移植性（Portability）：构件应能在不同的硬件运行平台和软件环境中工作。

3）构件的特点

（1）自描述。构件必须能够识别其属性、存取方法和事件，这些信息可以使开发环境将第三方软件构件无缝地结合起来。

（2）可定制。允许提供一个典型的图形方式环境，软件构件的属性只能通过控制面板来设置。

（3）可集成。构件必须可以被编程语言直接控制。构件也可以和脚本语言或者与从代码级访问构件的环境连接，这个特点使得软件构件可以在非可视化开发项目中使用。

（4）连接机制。构件必须能产生事件或者具有让程序员从语义上实现相互连接的其他机制。

3.6　信息系统开发模型

本节重点介绍瀑布模型，快速原型模型，增量开发模型，螺旋模型和喷泉模型。

3.6.1　瀑布模型

瀑布模型（Waterfall Model）是一个项目开发架构，开发过程是通过设计一系列阶段顺序展开的，从系统需求分析开始直到产品发布和维护，每个阶段都会产生循环反馈，因此，如果有信息未被覆盖或者发现了问题，那么最好“返回”上一个阶段并进行适当的修改，项目开发进程从一个阶段“流动”到下一个阶段，这也是瀑布模型名称的由来，见图 3-10。其主要包括软件工程开发、企业项目开发、产品生产以及市场销售等构造瀑布模型。

1970 年，温斯顿·罗伊斯（Winston Royce）提出了著名的“瀑布模型”，直到 20 世纪 80 年代早期，它一直是唯一被广泛采用的软件开发模型。

瀑布模型核心思想：瀑布模型核心思想是按工序将问题化简，将功能的实现与设计分开，便于分工协作，即采用结构化的分析与设计方法将逻辑实现与物理实现分开。将软件生命周期划分为制定计划、需求分析、软件设计、程序编写、软件测试和运行维护 6 个基本活动，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落。

瀑布模型的优缺点：瀑布模型的优点是为项目提供了按阶段划分的检查点；当前一阶段完成后，只需要去关注后续阶段；可在迭代模型中应用瀑布模型。瀑布模型的缺点是在项目各个阶段之间极少有反馈；只有在项目生命周期的后期才能看到结果；通过过多地强制完成日期和里程碑来跟踪各个项目阶段；瀑布模型的突出缺点是不适应用户需求的变化。

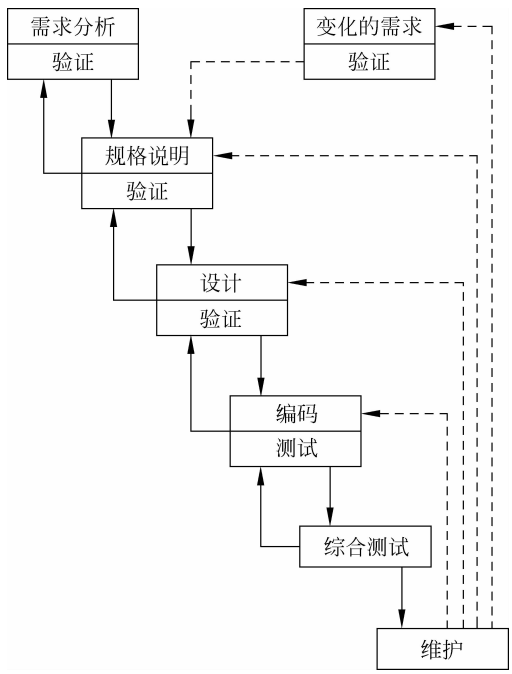

图 3-10 瀑布模型图

3.6.2 快速原型模型

1. 模型概述

快速原型法就是在系统开发之初，尽快给用户构造一个新系统的模型(原型)，反复演示原型并征求用户意见，开发人员根据用户意见不断修改完善原型，直到基本满足用户的要求进而实现系统，这种软件开发方法就是快速原型法，见图 3-11。原型就是模型，而原型系统就是应用系统的模型。它是待构筑的实际系统的缩小比例模型，但是保留了实际系统的大部分性能。这个模型可在运行中被检查、测试、修改，直到它的性能达到用户需求为止。因而这个工作模型很快就能转换成原样的目标系统。

2. 原型法的三个层次

第一层包括联机的屏幕活动，这一层的目的是确定屏幕及报表的版式和内容、屏幕活动的顺序及屏幕排版的方法。

第二层是第一层的扩展，引用了数据库的交互作用及数据操作，这一层的主要目的是论证系统关键区域的操作，用户可以输入成组的事务数据，执行这些数据的模拟过程，包括出错处理。

第三层是系统的工作模型，它是系统的一个子集，其中应用的逻辑事务及数据库的交互作用可以用实际数据来操作，这一层的目的是开发一个模型，使其发展成为最终的系统规模。

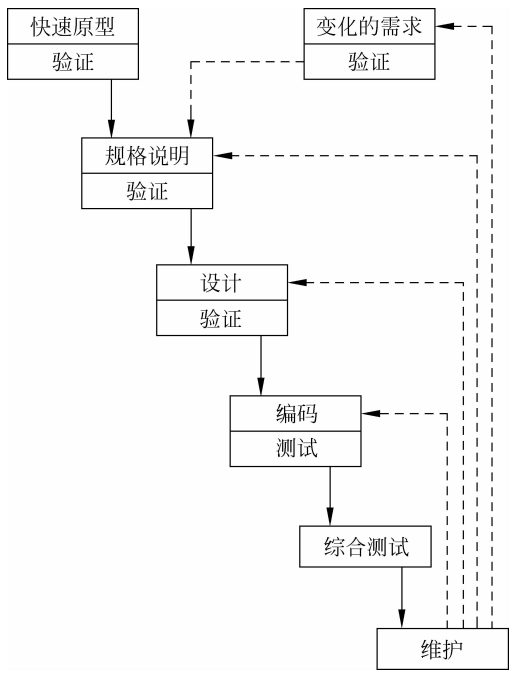

图 3-11　快速原型法模型

3. 模型优缺点

原型法的主要优点在于它是一种支持用户的方法，使得用户在系统生存周期的设计阶段起到积极的作用；它能减少系统开发的风险，特别是在大型项目的开发中，由于对项目需求的分析难以一次完成，应用原型法效果更为明显。原型法的概念既适用于系统的重新开发，也适用于对系统的修改；原型法不局限于仅对开发项目中的计算机方面进行设计，第三层原型法是用于制作系统的工作模型的。快速原型法要取得成功，要求有像第 4 代语言(4GL)这样的良好开发环境/工具的支持。原型法可以与传统的生命周期方法相结合使用，这样会扩大用户参与需求分析、初步设计及详细设计等阶段的活动，加深对系统的理解。近年来，快速原型法的思想也被应用于产品的开发活动中。

原型法的主要缺点是所选用的开发技术和工具不一定符合主流的发展；快速建立起来的系统结构加上连续的修改可能会导致产品质量低下。

3.6.3　增量开发模型

增量模型也称为渐增模型，如图 3-12 所示。使用增量模型开发软件时，把软件产品作为一系列的增量构件来设计、编码、集成和测试。每个构件由多个相互作用的模块构成，并且能够完成特定的功能。使用增量模型时，第一个增量构件往往实现软件的基本需求，提供最核心的功能；第二个增量构件提供更完善的编辑和文档生成功能；第三个增量构件实现拼写和语法检查功能；第四个增量构件完成高级的页面排版功能。

把软件产品分解成增量构件时，应该使构件的规模适中，规模过大或过小都不好。最佳分解方法因软件产品特点和开发人员的习惯而异。分解时唯一必须遵守的约束条件是，当

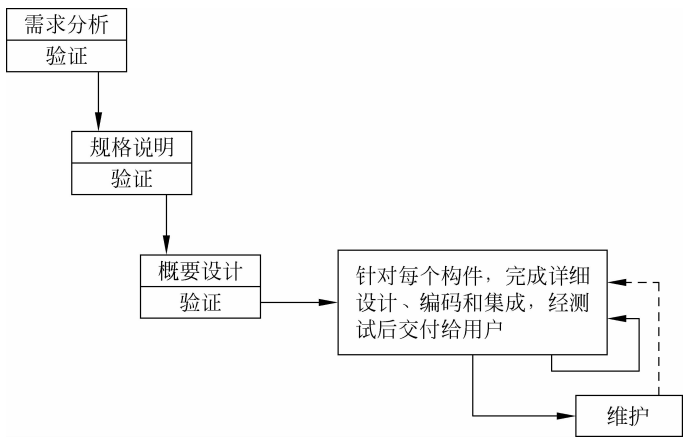

图 3-12　增量模型

把新构件集成到现有软件中时，所形成的产品必须是可测试的。

采用瀑布模型或快速原型模型开发软件时，目标都是一次就把一个满足所有需求的产品提交给用户。增量模型则与之相反，它分批地逐步向用户提交产品，整个软件产品被分解成许多个增量构件，开发人员一个构件接一个构件地向用户提交产品。从第一个构件交付之日起，用户就能做一些有用的工作。显然，能在较短时间内向用户提交可完成部分工作的产品，是增量模型的一个优点。

增量模型的另一个优点是，逐步增加产品功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给客户组织带来的冲击。

使用增量模型的困难是，在把每个新的增量构件集成到现有软件体系结构中时，必须不破坏原来已经开发出的产品。此外，必须把软件的体系结构设计得便于按这种方式进行扩充，向现有产品中加入新构件的过程必须简单、方便，也就是说，软件体系结构必须是开放的。但是，从长远观点看，具有开放结构的软件拥有真正的优势，这样的软件的可维护性明显好于封闭结构的软件。

因此，尽管采用增量模型比采用瀑布模型和快速原型模型需要更精心的设计，但在设计阶段多付出的劳动将在维护阶段获得回报。如果一个设计非常灵活而且足够开放，足以支持增量模型，那么，这样的设计将允许在不破坏产品的情况下进行维护。事实上，使用增量模型时开发软件和扩充软件功能（完善性维护）并没有本质区别，都是向现有产品中加入新构件的过程。

从某种意义上说，增量模型本身是自相矛盾的。它一方面要求开发人员把软件看作一个整体，另一方面又要求开发人员把软件看作构件序列，每个构件本质上都独立于另一个构件。除非开发人员有足够的技术能力协调好这一明显的矛盾，否则用增量模型开发出的产品可能并不令人满意。

如图 3-12 所示的增量模型表明，必须在开始实现各个构件之前就全部完成需求分析、规格说明和概要设计的工作。由于在开始构建第一个构件之前已经有了总体设计，因此风险较小。图 3-13 描绘了一种风险更大的增量模型：一旦确定了用户需求之后，就着手拟定第一个构件的规格说明文档，完成后规格说明组将转向第二个构件的规格说明，与此同时设

计组开始设计第一个构件……用这种方式开发软件，不同的构件将并行地构建，因此有可能加快工程进度。但是，使用这种方法将冒构件无法集成到一起的风险，除非密切地监控整个开发过程，否则整个工程可能毁于一旦。

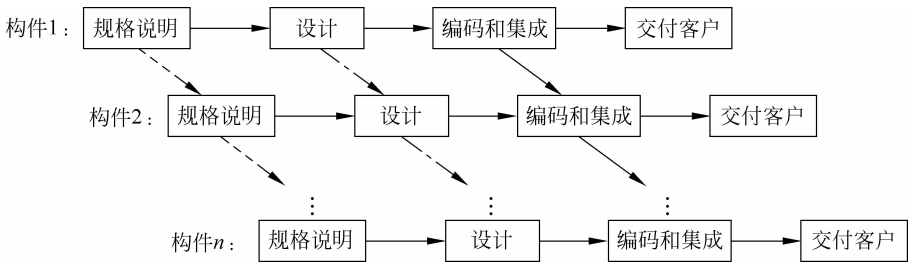

图 3-13　风险更大的增量模型

3.6.4　螺旋模型

1. 模型概述

1988 年，巴利·玻姆(Barry Boehm)提出了软件系统开发的“螺旋模型”，它将瀑布模型和快速原型模型结合起来，强调了其他模型所忽视的风险分析，特别适合于大型复杂的系统。

螺旋模型采用一种周期性的方法来进行系统开发。这会导致开发出众多的中间版本。使用它，项目经理在早期就能够为客户实证某些概念。该模型是快速原型法，以进化的开发方式为中心，在每个项目阶段使用瀑布模型法。这种模型的每一个周期都包括需求定义、风险分析、工程实现和评审 4 个阶段，由这 4 个阶段进行迭代。软件开发过程每迭代一次，软件开发又前进一个层次，见图 3-14。

2. 采用螺旋模型的软件过程

螺旋模型的基本做法是在“瀑布模型”的每一个开发阶段前引入一个非常严格的风险识别、风险分析和风险控制，它把软件项目分解成一个个小项目。每个小项目都标识一个或多个主要风险，直到所有的主要风险因素都被确定。螺旋模型强调风险分析，使得开发人员和用户对每个演化层出现的风险有所了解，继而做出应有的反应，因此特别适用于庞大、复杂并具有高风险的系统。对于这些系统，风险是软件开发不可忽视且潜在的不利因素，它可能在不同程度上损害软件开发过程，影响软件产品的质量。减小软件风险的目标是在造成危害之前，及时对风险进行识别及分析，决定采取何种对策，进而消除或减少风险的损害。

螺旋模型沿着螺线进行若干次迭代，图 3-14 中的 4 个象限代表了以下活动：①制定计划，确定软件目标，选定实施方案，弄清项目开发的限制条件；②风险分析，分析评估所选方案，考虑如何识别和消除风险；③实施工程，实施软件开发和验证；④客户评估，评价开发工作，提出修正建议，制定下一步计划。

螺旋模型由风险驱动，强调可选方案和约束条件从而支持软件的重用，有助于将软件质量作为特殊目标融入产品开发之中。

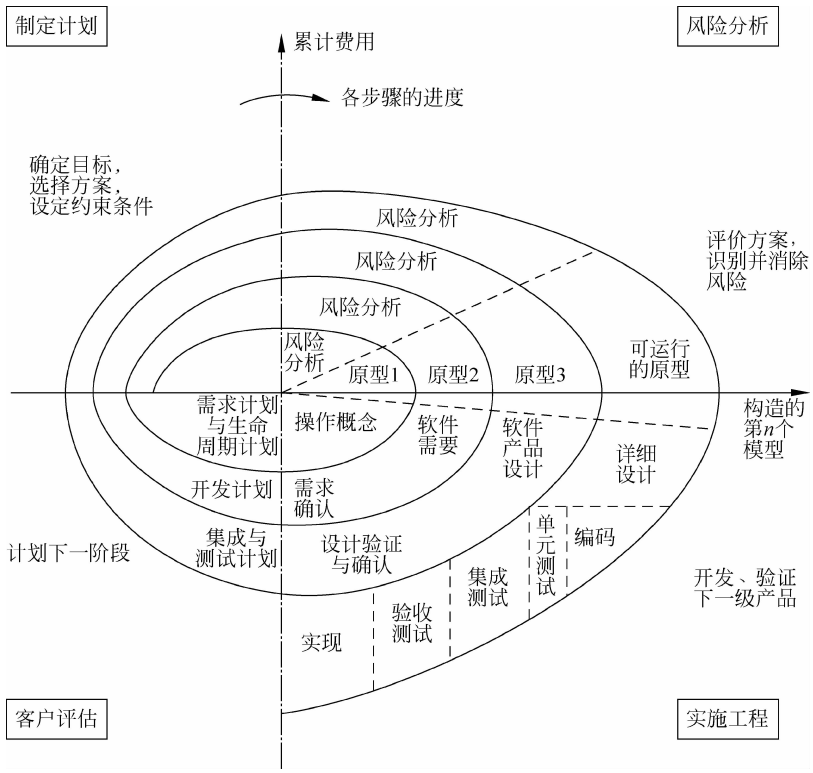

图 3-14 螺旋模型

3.6.5 喷泉模型

1. 模型概述

喷泉模型(Fountain Model)是一种以用户需求为动力，以对象为驱动的模型，主要用于描述面向对象的软件开发过程。

该模型认为软件开发过程自下而上周期的各阶段是相互迭代和无间隙的。软件的某个部分常常被重复工作多次，相关对象在每次迭代中随之加入渐进的软件成分。无间隙指在各项活动之间无明显边界，如分析和设计活动之间没有明显的界限。由于对象概念的引入，表达分析、设计、实现等活动只用对象类和关系，从而可以较为容易地实现活动的迭代和无间隙，使其开发自然地包括复用。

喷泉模型不像瀑布模型那样，需要分析活动结束后才开始设计活动，设计活动结束后才开始编码活动。该模型的各个阶段没有明显的界限，开发人员可以同步进行开发。其优点是可以提高软件项目开发效率，节省开发时间，适应于面向对象的软件开发过程。由于喷泉模型在各个开发阶段是重叠的，因此在开发过程中需要大量的开发人员，不利于项目的管理。此外，这种模型要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入各种信息、需求与资料的情况。

2. 模型应用解释

迭代是软件开发过程中普遍存在的一种内在属性。经验表明，软件过程各个阶段之间的迭代或一个阶段内各个工作步骤之间的迭代，在面向对象范型中比在结构化范型中更常见。如图 3-15 所示的喷泉模型是典型的面向对象生命周期模型。

“喷泉”这个词体现了面向对象软件开发过程迭代和无缝的特性。图中代表不同阶段的圆圈相互重叠，这明确表示两个活动之间存在交迭；而面向对象方法在概念和表示方法上的一致性，保证了在各项开发活动之间的无缝过渡。事实上，用面向对象方法开发软件时，在分析、设计和编码等各项开发活动之间并不存在明显的边界。图 3-15 中在一个阶段内的向下箭头代表该阶段内的迭代(或求精)。图 3-15 中较小的圆圈代表维护，圆圈较小象征着采用了面向对象范型之后维护时间缩短了。

为避免使用喷泉模型开发软件时开发过程过分无序，应该把一个线性过程(例如，快速原型模型或图 3-15 中的中心垂线)作为总目标。但是，同时也应该记住，面向对象范型本身要求经常对开发活动进行迭代或求精。

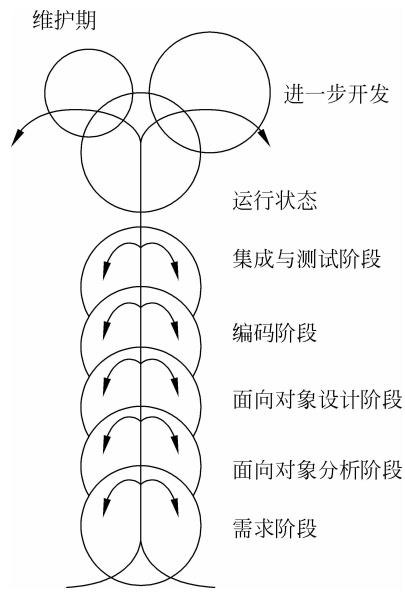

图 3-15　喷泉模型

习题

1. 名词解释

(1) PSP；(2)甘特图；(3) 成本管理；(4) TSP；(5) SQA；(6) FPA；(7) OO 方法；(8) OOA 方法；(9) 软件复用；(10) 喷泉模型。

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 形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 形曲线表示。　(　　)

(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 生存周期分为 4 个阶段，即需求、设计、实现和测试。　(　　)

(7) 软件过程为一个为建造高质量软件所需完成的任务的框架,即形成软件产品的一系列步骤,包括中间产品、资源、角色及过程中采取的方法、工具等范畴。 ( )

(8) 构件代表系统中的一部分物理实施,包括软件构件框架或其等价物。 ( )

(9) 瀑布模型核心思想是按工序将问题化简,将功能的实现与设计分开,便于分工协作,即采用结构化的分析与设计方法将逻辑实现与物理实现分开。 ( )

(10) 使用增量模型开发软件时,把软件产品作为一个整体来设计、编码、集成和测试。 ( )

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)

(3) 软件工程涉及_____,_____,_____,系统平台,标准,设计模式等方面。

(4) 结构化方法是一种传统的软件开发方法,它是由_____、_____和结构化程序设计三部分有机组合而成的。

(5) 结构化分析就是使用数据流程图、_____、_____、_____和判定树等工具,来建立一种新的、称为结构化说明书的目标文档,也就是需求规格说明书。

(6) 瀑布模型将软件生命周期划分为制定计划、_____、_____、_____、软件测试和运行维护 6 个基本活动,并且规定了它们自上而下、相互衔接的固定次序,如同瀑布流水,逐级下落。

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 1/4

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) 软件开发流程(Software Development Process)即软件设计思路和方法的一般过程,包括以下哪几项?( )

A. 设计软件的功能和实现的算法和方法　B. 软件的总体结构设计和模块设计

C. 成本预算和效益分析　　D. 编程和调试

E. 程序联调和测试以及编写

(6) 结构化设计方法的设计原则遵循哪几条？(　　)

A. 以类和继承为构造机制　　B. 使每个模块尽量只执行一个功能

C. 每个模块用过程语句调用其他模块　　D. 模块间传送的参数作数据用

(7) 在 OOD 的设计过程中，要展开的主要有如下哪几项工作？(　　)

A. 对象定义规格的求精过程　　B. 需求分析和详细设计

C. 数据模型和数据库设计　　D. 成本核算

(8) 软件构件的属性有哪些？(　　)

A. 有用性　　B. 可用性　　C. 质量　　D. 适应性

E. 可移植性

(9) 软件构件的特点有哪些？(　　)

A. 自描述　　B. 可继承　　C. 可集成　　D. 连接机制

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。

(3) 软件生命周期有哪几个阶段？

(4) 简述结构化分析的步骤。

(5) 简述面向对象程序设计基本步骤。

(6) 简述 OOA 方法的基本步骤。

(7) 简述软件复用的几个级别。

(8) 简述快速原型模型原型法的三个层次。

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。

(3) 软件开发流程的步骤。

(4) 解释面向对象方法里的对象、类、消息、封装、继承性、多态性这些基本概念。

(5) 画出螺旋模型图，并给出意义解释。