第2章 软件项目管理

【章节概述】

本章讲述项目管理与软件开发过程之间的关系，重点讲述软件项目管理的含义及任务目标，并对项目管理中的重要组成部分进行介绍，包括项目计划、质量管理、配置管理、组织管理和风险管理等。

【教学重点与难点】

掌握项目计划的作用和构建步骤；

掌握软件质量模型的含义和软件质量属性。

【引言】

项目管理是在当今各行各业要顺利开展业务必不可少的环节，对于软件项目的开发更是如此。那么，项目管理在软件的开发过程中到底起到一个什么样的作用，它与软件生命周期各阶段的任务目标有什么关系呢？下面我们通过一个例子来了解一下。

以一个拥有汽车装配流水线的汽车生产企业为例，参照如图 2-1 给出的部分汽车生产流水线生产过程中的一些任务，根据你目前对管理的认识，分析一下在汽车生产的过程中，哪些属于汽车生产流水线上的任务，哪些属于生产管理方面的任务？

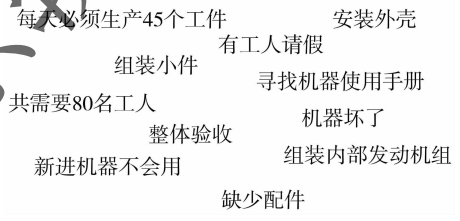

图 2-1 汽车生产流水线上的任务

对于汽车生产流水线上的任务，一般都是能够看到产品的生产过程的环节，能够产生直接经济效益的生产技术活动。例如，安装外壳、组装小件、整体验收、组装内部发动机等。而生产管理活动往往是为了保证正常的生产活动而进行的一些保障性措施。例如，为了保证产品能够按时生产足够的数量，规定了每天必须完成的产品数量，每天必须生产 45 个工件；生产的过程中需要 80 名工人才能保证生产的正常进行；如果有工人请假的话，企业应该提供相应的规章制度能够进行遵守，而不至于对生产产生影响；如果不会使用机器，需要使用到机器的使用手册，那么，应该有一套资料保存的制度，能够让工人在需要时立刻找到它，而不至于在寻找上浪费时间；对于缺少配件、机器坏了要根据企业的规章制度找相关的部门帮助解决等等。

从上面我们看到，所谓的管理，从字面上讲分为“管”和“理”。管，就是规定、指派，设有一定的规章制度需要进行遵守，给人的感情色彩有些强制性；而理则稍稍柔和一点，它强调的是理顺、疏通、协调，将各种关系（部门）协调起来为一个共同的目标来共同努力。

下面，我们就来看一看在软件项目实施过程当中是如何进行“管”和“理”的。

2.1　软件项目管理概述

软件工程管理几乎与软件工程同时于 20 世纪 70 年代中期引起人们的广泛关注。美国国防部立题专门研究的结果发现 70％的项目是因为管理不善而不能很好完成，而并不是因为技术原因。这表明软件管理是影响软件研发的关键因素，而技术则处于次要位置。到了 20 世纪 90 年代中期，软件工程管理不善的问题仍然存在。对美国软件工程实施现状的调查表明软件研发的情况依然很难预测，大约只有 10％的项目能够在规定的费用内按期完成。1995 年，美国共取消了 810 亿美元的软件项目，其中 31％的项目未做完就取消了，53％的软件项目进度通常要延长 50％的时间，通常只有 9％的软件项目能够及时交付并且费用也不超支。

软件工程管理和其他工程管理相比有其特殊性。首先，软件是知识产品，进度和质量都难以度量，生产效率也难以保证。其次，软件系统复杂程度也是超乎想象的。例如，宇宙飞船的软件系统源程序代码多达 2 000 万行，如果按过去的生产效率一个人一年只能写 1 万行代码的话，那么需要 2 000 人年的工作量，这是非常惊人的。正因为软件如此复杂和难以度量，软件工程管理的发展还很不成熟。本章所讲的软件工程管理不仅仅是软件工程过程本身的管理，还包括软件工程过程有关的外部影响因素的管理，是全面的软件工程管理。

软件管理的目标就是保证软件项目的成功，就是保证在规定的时间内，在预算计划内开发出令用户满意的软件产品。软件开发的所有成员都紧紧围绕这三个目标进行工作。

2.1.1　软件项目产品的特点

软件项目是以软件为产品的项目。软件产品的特质决定了软件项目管理和其他领域的项目管理存在着一定的差异。

1.抽象性

软件是脑力劳动的结果，是一种逻辑实体，具有抽象性。在软件项目的开发过程中没有具体的物理制造过程，因而不受物理制造过程的限制，其结束以软件产品交付用户为标志。软件一旦研制成功，就可以大量复制，因此软件产品需要进行知识产权的保护。

2.缺陷检测的困难性

在软件的生产过程中，检测和预防缺陷是很难的，需要进行一系列的软件测试活动以降低软件的错误率。但即使如此，软件缺陷也是难以杜绝的，这就像一些实验科学中的系统误差，只能尽量避免，但不能够完全根除。

3.高度的复杂性

软件的复杂性可以很高。有人甚至认为，软件是目前为止人类所遇到的最为复杂的事

物。软件的复杂性可能来自实际问题的复杂性，也可能来自软件自身逻辑的复杂性。

4.缺乏统一规则

作为一个学科，软件开发是年轻的，还缺乏有效的技术，目前已经有的技术还没有经过很好的验证。不可否认，软件工程的发展带来了许多新的软件技术，例如软件复用、软件的自动生成技术；也研制出了一些有效的开发工具和开发环境，但这些技术在软件项目中采用的比率仍然很低，直到现在，软件开发还没有完全摆脱手工艺的方式，也没有统一的方法，否则它早已通过装配生产线实现了。具有不同经验和学科教育背景的人们为软件开发的方法论、过程、技术、实践和工具的发展做出了贡献，这些多样性也带来了软件开发的多样性。

2.1.2 软件项目管理的内容

IT项目管理是以信息技术为基础的项目管理，它是项目管理的一种特殊形式，是随着信息技术的发展而诞生并不断完善的一种新的项目管理。一般项目管理的科学理论、思想方法和技术在IT项目管理中依然适用，同时由它的特殊性也使其有特殊的管理问题需要研究和讨论。

1.软件项目管理的定义

软件项目管理的概念涵盖了管理软件产品开发所必需的知识、技术及工具。根据美国项目管理协会PMI对项目管理的定义“在项目活动中运用一系列的知识、技能、工具和技术，以满足或超过相关利益者对项目的要求”，我们可以给出软件项目管理的一种定义：在软件项目活动中运用一系列知识、技能、工具和技术，以满足软件需求方的整体要求。

2.软件项目管理的过程

为保证软件项目获得成功，必须清楚其工作范围、要完成的任务、需要的资源、需要的工作量、进度的安排、可能遇到的风险等。软件项目的管理工作在技术工作开始之前就应开始，而在软件从概念到实现的过程中继续进行，且只有当软件开发工作最后结束时才终止。管理的过程分为如下几个步骤。

(1)启动软件项目

启动软件项目是指必须明确项目的目标和范围，考虑可能的解决方案，明确技术和管理上的要求等，这些信息是软件项目运行和管理的基础。

(2)制订项目计划

软件项目一旦启动，就必须制订项目计划。计划的制订以下面的活动为依据：

①估算项目所需要的工作量。

②估算项目所需要的资源。

③根据工作量制订进度计划，继而进行资源分配。

④做出配置管理计划。

⑤做出风险管理计划。

⑥做出质量保证计划。

在软件项目进行的过程中，严格遵守项目计划，对于引起不可避免的变更，要进行适当的控制和调整，但要确保项目计划的完整性和一致性。

(3)评审项目计划

对项目计划的完成程度进行评审,并对项目的执行情况进行评价。

(4)编写管理文档

项目管理人员根据软件合同确定软件项目是否完成。项目一旦完成,则检查项目完成的结果和中间记录文档,并把所有的结果记录下来形成文档而保存。

3.软件项目管理的内容

软件项目管理的内容涉及到上述软件项目管理过程的方方面面,概括起来主要有如下几项:

(1)软件项目范围管理;

(2)软件项目成本管理;

(3)软件项目进度管理;

(4)软件项目配置管理;

(5)软件项目组织管理;

(6)软件项目质量管理;

(7)软件项目风险管理;

(8)软件项目沟通管理;

(9)软件项目集成管理。

软件项目管理的框架,如图 2-2 所示。

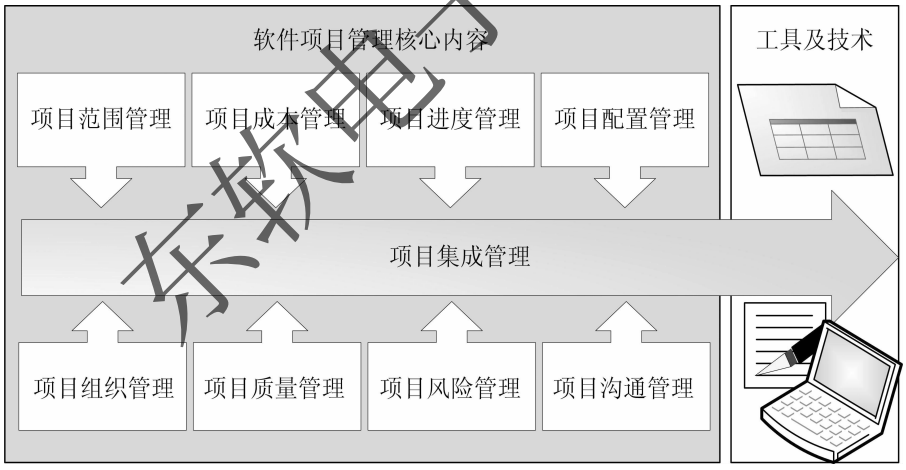

图 2-2　软件项目管理框架

2.2　项目范围管理

项目范围的管理也就是对项目应该包括什么和不应该包括什么进行相应的定义和控制。它包括用以保证项目能按要求的范围完成所涉及的所有过程,包括:确定项目的需求、定义规划项目的范围、范围管理的实施、范围的变更控制管理以及范围核实等。烽火猎头专家亦认为项目范围是指产生项目产品所包括的所有工作及产生这些产品所用的过程。项目

干系人必须在项目要产生什么样的产品方面达成共识，也要在如何生产这些产品方面达成一定的共识。

2.2.1 项目范围变更控制

用户需求的不确定、公司上层对产品交付期的严格要求、人员的不合理流动等，使得项目组要制定一个完美的项目需求调研的分析计划几乎成为不可能的奢望。反过来，压缩需求期的工作，减少文档的编写，把相互交流和沟通降到最低限度，又必然造成用户因不满意产品需求的质量，不断修改变化需求，从而使得后期维护成本增加，造成恶性循环。

IT项目经常存在范围蔓延的问题，即项目范围存在不断扩大的趋势。项目范围蔓延是指项目范围在人们不注意的情况下逐步的微量增加，尤其在时间跨度较长的项目中经常出现。范围蔓延会导致项目进度无法控制、项目预算严重超支、项目回款延期、项目所交付的产品差强人意；同时也会导致项目团队对项目失去了方向、士气低落、项目迟迟不能验收、项目成果不能得到验收并付诸使用，使买卖双方两败俱伤。一般来讲，需求的变更通常意味着需求的增加，相对而言需求的减少相对很少，而且处理需求减少方面的问题也比较容易。

2.2.2 项目范围变更原因

范围变更的表现形式多种多样，如客户临时改变对功能需求的想法，项目预算发生变化等。要IT项目中，这些需求范围变更可能来自方案服务商、客户或者产品供应商，也可能来自项目组内部，分析各种需求变更的原因，可以归结为以下4个方面：

(1)范围没有明确就开始细化。范围的细化一般是由需求分析人员根据用户提出描述性的、总结性的短短几句话去细化、提取其中的一个个功能，并给出相应的描述。当客户向需求分析人员提出需求的时候，往往是将自己的想法用自然语言来表述的，这样的表示结果对于真实的需求来说只是某个角度的描述，不能保证这样的需求描述达到百分之百的正确理解。如果用户在需求不明确甚至提不出需求，或者因为项目组对业务不熟悉或者没有与用户密切配合，需求分析工作做得不细致等原因，使得需求范围没有明确就开始了细化工作，当进入项目实施阶段，需求范围发生变化，就需要做很大的改动。

(2)系统实施时间过长。大中型IT项目的建设需要延续一段时间，当客户得出需求范围时，并不能立刻看到系统的运行情况，往往是双方认为理解没有分歧时，开发方就开始工作。在项目漫长的实施过程中，客户由于自身业务发生变化或突然产生新的想法，会不时地对项目提出新的需求；或者当客户拿到差不多可以试用的交付物时，他们会对系统的功能、性能等从一些亲身的感受出发，提出需求变更请求。

(3)用户业务需求改变。由于客户竞争激烈，运行情况不确定，需要随时对业务或环境变化做出反应，用户自然会经常提出需求变更的请求。

(4)系统正常升级。由于开发方自身版本升级、性能改进、设计调整等要求会产生需求变更。

2.2.3 范围变更控制过程

许多IT项目都存在范围蔓延的趋势，无论是项目的开始阶段，还是项目即将结束的阶

段，发生范围变更都是不可避免的。因此，再好的计划也不可能做到一成不变。而项目范围的变更必然对项目产生影响，除了需要对项目范围加以核实以外，关键问题是如何对变更进行有效的控制。

范围变更控制就是指对有关项目范围的变更实施控制。控制好范围变更必须有一套规范的变更管理过程，在发生变更时遵循规范的变更程序来管理变更。其过程如图 2-3 所示。

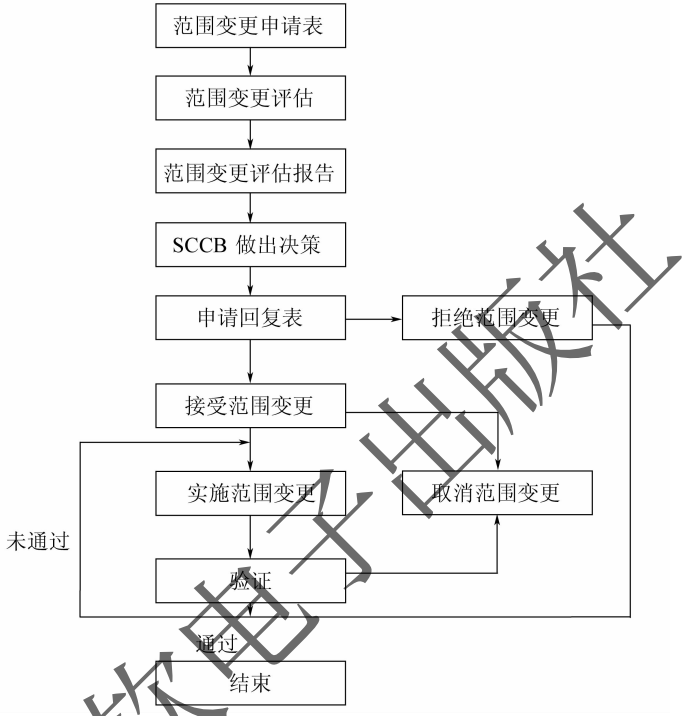

图 2-3　范围变更控制流程

(1)在发生范围变更时，首先需要向变更控制委员会(SCCB)提交范围变更申请表。对于任何范围变更请求，首先要做的是记录下来这个范围变更请求是什么，是由哪一类利益相关者提出来的，以及相应的联系方式。因为有些变更请求如果在本阶段不被接受，也许可以成为以后参考的功能或范围。

(2)变更控制委员会(SCCB)对范围变更进行评估。SCCB 是决定批准或拒绝特定项目中提出的变更请求的团队，通常由有关各方的代表组成，主要完成决定变更哪些需求，变更是否在项目范围之内；评估变更的范围，决定变更是否可以接受；对变更的需求设置优先级，制定版本规定等工作。SCCB 需要对范围变更请求产生的原因进行分析，比如，是由于在项目初期没有明确产品范围而产生的项目变更；或是没有明确项目范围产生的变更；还是由于外部事件所致。在评估时 SCCB 进行范围变更分析，精确理解需求，评估系统对范围变更的接纳程度、变更的代价、变更对系统总体架构，甚至产品发展的影响，为变更控制委员会决定是否批准范围变更提供依据。在范围变更评估分析中，还要进行需求范围稳定性分析。过于频繁的范围变更表明项目进程已经超出了需求变化的范围，项目经理应该考虑项目组织管理方面是不是发生了问题。

(3)SCCB 根据项目现有进度进行项目范围变更的项目进度影响、费用及项目可接受影

响的程度，对项目范围变更排列优先级，提出变更请求采取的应对措施建议，记录风险和相应的风险应对计划。同时与项目赞助人协商项目变更影响、解决变更请求需要的条件、相应的费用变化以及项目赞助人的可接受程度等问题，据此确定是否实施变更。如果拒绝范围变更请求，则范围变更控制过程结束；如果接受范围变更请求，需要将范围变更加入现有项目详细计划中，更新相应的项目文档，通知相应项目有关人员关于项目内容、进度、人员、费用的变更，接下来就应该实施范围变更。

(4)实施范围变更。实施过程主要完成以下任务：

①跟踪所有范围变更影响的工作产品。当确定某一需求范围发生变更时，根据需求跟踪矩阵，修改与范围变更需求有关的各层、各环节需求项，如涉及需求项的设计模型、代码模块、测试用例等；同时完整地跟踪所有范围变更所影响的地方，甚至包括对最终产品以外的影响。例如，因需求变更，版本控制没有相应的记录、产品使用手册没有做相应的修改等。在实施范围变更的过程中，如果发现范围变更不恰当，如出现了未预料到的高额费用等，可以及时取消范围变更。

②确定是否调整需求基线。需求范围变更以后，SCCB决定是否调整需求基线。确定是反映为基线的调整，还是反映为版本的变化。基线变化可以作为产品标准的变化，也可以理解为将发布一个新版本的产品。但是新版本产品并不一定就是新产品，因此SCCB要决定对新需求全面升级还是局部更改，是基线变化还是个别版本变化。

③维护范围变更记录和文档。决定变更基线或提升版本以后，需要做好记录，修改相应的文档。范围变更记录要记录范围变更原因、内容、影响、实现过程和其他相应的变更等。范围变更记录越完整，对于追溯甚至以后可能发生的回退就越有帮助。

④范围变更后还需要进行验证，如果未通过验证的话，要取消范围变更。如果通过验证，则范围变更实施结束。这时需要记录实际项目范围变更带来的影响，总结经验教训。

2.2.4 实施范围变更管理原则

(1)建立需求范围基线，需求范围基线是指是否允许需求变更的分界线。在开发过程中，需求确定并经过评审后(用户参与评审)，可以建立第一个需求基线。随着项目的进展，需求的基线也在变化。此后每次变更并经过评审后，都要重新确定新的需求基线。

(2)制定简单、有效的变更控制流程，并形成文档。在建立了需求基线后，提出的所有变更都必须遵循这个控制流程。同时，这个流程具有一定的普遍性，并对以后的IT项目开发和其他项目都有借鉴作用。

(3)成立项目变更控制委员会(SCCB)或具有相关职能的类似组织，负责裁定接受哪些变更。

(4)需求变更一定要先申请然后再评估，最后经过与变更级别相当的评审确认。

(5)需求变更后，受影响的软件计划、产品、活动都要进行相应的变更，以保证和更新的需求一致。

(6)妥善保存变更产生的相关文档。

2.2.5 项目范围变更控制

IT项目的生命周期分为启动、计划、实施、控制和收尾5个过程。范围变更的控制不应

该只是项目实施过程考虑的事情，而是要分布在整个项目生命周期。项目中不可避免地会发生范围的变更，无论是在项目的开始阶段还是将要结束阶段，都有可能会发生项目范围的变更，而项目范围的变更会自然地对项目产生影响。所以，怎么样控制项目的范围变更是项目管理的一个重要内容。

项目所处的阶段越早，项目的不确定性越大，项目范围调整或变更的可能性就越大，此时带来的代价比较低。随着项目的进行，不确定性逐渐减小，变更的代价、付出的人力和资源逐渐增加，就会增加决策的难度。为了将项目范围变更的影响降到最小，就需要采用综合变更控制方法。综合变更控制的主要内容有找出影响项目变更的原因从而判断项目变更范围是否已经发生等。

进行综合变更控制的主要依据是项目计划、变更请求和提供了项目执行状况信息的绩效报告。为了保证项目变更的规范和有效实施，通常项目实施组织会采取以下措施：

1.项目启动阶段的需求范围变更预防

任何 IT 项目的范围变更都是不可避免的，只能从项目启动的需求分析阶段就开始积极应对。对一个需求分析做得很好的项目来说，基准文件定义的范围越详细越清晰，客户跟项目经理不断提出需求范围变更的机会就越少。如果需求没做好，基准文件里的范围就会含糊不清，往往要付出许多无谓的代价。如果需求做得好，文档清晰且又有客户签字，那么后期客户提出的变更一旦超出了合同范围，就需要另外收费。这并非要刻意赚取客户的钱财，而是不能让客户养成经常变更范围的习惯，否则对于项目来说后患无穷。

2.项目实施阶段的需求范围变更

成功项目和失败项目的区别就在于项目的整个过程是否可控。项目经理应该树立一个理念——“范围变更是必然的、可控的、有益的”。项目实施阶段的变理控制需要分析变更请求，评估变更可能带来的风险和修改基准文件，特别需要注意以下几点：

(1)范围一定要与投入有联系，如果范围变更的成本由开发方来承担，则项目范围的变更就会频繁发生。所以，在项目的开始，无论是开发方还是出资方都要明确这一条：需求变，项目的投入也要变。

(2)范围的变更要经过出资者的认可，保证关键利益相关者对范围的变更有成本的概念，能够慎重地对待范围的变更。

(3)小的范围变更也要经过正规的范围变更流程。人们往往不愿意为小的范围变更去执行正规的范围变更流程，认为降低了开发效率，浪费了时间。但这种观念经常使得范围逐渐变得不可控，最终导致项目的失败。

(4)精确的需求与范围定义并不会阻止需求的变更。并非对需求定义得越细，就越能避免需求的渐变，这是两个层面的问题。太细的需求定义对需求渐变没有任何效果，因为需求的变化是永恒的，并非需求写细了，它就不会变化了。

(5)注意沟通的技巧。实际情况是用户、开发者都认识到了上面的几点问题，但是由于需求的变更可能来自客户方，也可能来自开发方，因此，作为需求管理者，项目经理需要采用各种沟通技巧来使项目的各方各得其所。

(6)在开发上尽量根据情况采用多次迭代的方式，在每次迭代的同时让客户参与和使用软件，对下一步的开发提出建议，争取在项目前期有效地减少后期可能出现的变更情况。

3.项目收尾阶段的总结

能力的提高往往不是从成功的经验中来，而是从失败的教训中来。范围变更过程的结果之一就是教训的总结。许多项目经理不注意经验教训的总结和积累，即使在项目运作过程中碰得头破血流，也只是抱怨运气、环境和团队配合不好，很少系统地分析总结，或者不知道如何分析总结，以至于同样的问题反复出现。

事实上，项目总结工作应做出现有项目或将来项目持续改进工作的一项重要内容，同时也可以作为对项目合同、设计方案内容与目标的确认和验证。项目总结工作包括对项目中事先识别的风险和没有预料到而发生的变更等风险的应对措施的分析和总结，也包括对项目中发生的变更和项目中发生问题的分析统计的总结。

2.3 项目成本管理

项目成本管理是承包人为使项目成本控制在计划目标之内所作的预测、计划、控制、调整、核算、分析和考核等管理工作。项目成本管理就是要确保在批准的预算内完成项目，具体项目要依靠制定成本管理计划、成本估算、成本预算、成本控制四个过程来完成。项目成本管理是在整个项目的实施过程中，为确保项目在以批准的成本预算内尽可能好的完成而对所需的各个过程进行管理。

2.3.1 成本管理过程

项目成本管理由一些过程组成，要在预算下完成项目，这些过程是必不可少的。这些过程的主要框架为：

(1)资源计划过程：决定完成项目各项活动需要哪些资源(人、设备、材料)以及每种资源的需要量。

(2)成本估计过程：估计完成项目各活动所需每种资源成本的近似值。

(3)成本预算过程：把估计总成本分配到各具体工作。

(4)成本控制过程：控制项目预算的改变。

以上四个过程相互影响、相互作用，有时也与外界的过程发生交互影响，根据项目的具体情况，每一过程由一人或数人或小组完成，在项目的每个阶段，上述过程至少出现一次。同时，以上过程是分开陈述且有明确界线的，实际上这些过程可能是重复的、相互作用的，对此本教材不作详细讨论。

项目成本管理主要与完成活动所需资源成本有关。然而，项目成本管理也考虑决策对项目产品的使用成本的影响。例如：减少设计方案的次数可减少产品的成本，但却增加了今后顾客的使用成本，这个广义的项目成本叫项目的生命周期成本。许多应用领域，未来财务状况的预测和分析是在项目成本管理之外进行的。但有些场合，预测和分析的内容也包括在成本管理范畴，此时就得使用投资收益、有时间价值的现金流、回收期等技巧。

项目成本管理还应考虑项目相关方对项目信息的需求，不同的相关方在不同时间以不同方式对项目成本进行度量。当项目成本控制与奖励挂钩时，就应分别估计和预算可控成

本和不可控成本，以确保奖励能真正反映业绩。

2.3.2　成本管理手段

1.基于预算的目标成本控制方法

在国内企业中间，采取严格的预算管理的企业并不多见。尽管一些企业管理者从各种渠道了解到实行预算管理的种种好处，因而每到年底，他们总会要求财务部门，或者是销售部门，或者是“总经办”这样的部门去为来年做一份预算。然而，由于大家都对怎样做预算一知半解，企业平时又没有积累起做预算所需要的各种数据，以及做预算所需要相应的组织环境，加上时间十分紧迫（通常他们会要求有关人员在 1～7 天内完成）和其他一些原因，他们做出的预算，其实只是做预算者在揣摸领导意图后拿出的一个来年的花钱的计划。而且做这个计划的人通常明明知道这个花钱计划只是做一做，满足老板当前的要求而已。在大多数企业中，很少有人认为预算会是有用的，不是指预算从理论上讲无用，而是在他们的企业没有用。

人们普遍确信的是，“计划没有变化快”。“计划没有变化快”这句话可以有多种理解。一种理解是，老板自己不会按照他要求做出的预算执行，预算做得再好也没有用。一种理解是，我们的企业根本就做不出切实可行、行之有效的预算。还有一种理解是，人际关系太复杂、当家的人太多，即便老板要坚持按预算办事，也不定哪一天就有一人物把他破坏了。可能还有一种解释是，环境因素变化太快，企业发展的变数太多，根本就无法预测两三个月以后的事，所以预算做不出来，做了也一定会不实用。

但是，国外成功企业的经验显示，预算管理是有效的成本控制方法。所谓预算，通俗的讲就是，事前确定好明天花多少钱？哪里花钱？谁来花钱？怎么花钱？谁来控制花钱？要回答这些问题，不仅需要对全盘有把握，而且知道资金从哪里带来（并保证能得到这笔资金），以及知道各种需要购进的东西的未来价格走势。因为是按计划来花钱，自然就不会乱花钱、花冤枉钱。为什么说按事前的计划花钱就不会花冤枉钱呢？因为计划通常是事前经过在各部门的共同参与下，反复讨论协商出来的。

当然，正如世界上没有绝对好的东西一样。基于预算的目标成本控制方法也并非百分之百好用，因为总有一些事情是无法预计的。但这不能否定预算管理的无效，预算一旦执行以后，也不是铁板一块，必要的时候是可以作适当调整的。最重要的是，有预算管理一定会比没有预算管理好。

2.基于标杆的目标成本控制方法

所谓标杆，就是样板，就是别人在某些方面做得比自己好，所以要以别人为楷模来做，甚至比别人做得还要好，或说别人做到了那样的效果，所以我也要求自己达到甚至超过那样的效果。

这里的“别人”有三层意思：其一，它可以是别的企业。当一个企业在某些方面做到某种较好程度时，通常就会有一批企业继而效仿它，比如 A 汽车制造厂由于采用某种新的工艺，促其每台车的生产成本降低了 1%，因此众多的汽车生产企业也纷纷采取这种工艺。又比如，某企业的人均贡献率达到了某种水平，于是一家企业开始研究它是如何达到那个水平的，当这家企业确信自己找到答案时，它便以那家企业为目标，采取措施（不一定跟那个企业

的做法一模一样)试图取得同样的人均贡献率,甚至更高的人均贡献率。以其他企业为标杆,其学习途径主要有三个:一是,通过一定的媒介(电视、报纸、期刊、书籍、网络、管理顾问等)知道某个企业在某一方面或几个方面做得比自己好,因而决意学习它。二是,到那家企业参观学习或由那家企业的人员当面介绍,因而决意学习它。三是,在那家企业工作过的人员带来了那家企业的经验,在本企业推广它。其二,以自身企业过去的某些绩效为标准来作为未来的目标予以控制。比如,在本企业的历史上,最高的人均利润贡献额为50 000元,或者销售费用率仅为8%,于是决意在下一年度以此为目标来予以控制。这一点与基于历史数据的目标成本控制方法是基本一致的。其三,是以本企业的某个部门或某个人创造的某项纪录为目标,要求其他部门或其他人以此为标杆,并力争超越他。比如,某部门连续三个月创造了人均办公用品费用不超过10元的纪录,经分析认为,全公司的其他部门如果努力控制办公用品使用,也能达到这个效果,于是便在全公司倡导或强制性地执行以那个部门的这一结果为标准,来实施降低办用品费用的计划。又比如,某位计件工当月创造了一项较高的生产记录,公司便号召其他人向他学习,也是一种标杆式的管理方法。

3.基于市场需求的目标成本控制方法

基于市场需求的目标控制方法(有时也把它称为“基于决策层意志的成本控制法”,因为这种方法在使用过程中,决策者的意志将起主导作用)。下面是一个典型的基于市场需求的目标成本控制方法的操作案例。

某公司计划开发生产一种新产品——A型涂料,公司技术人员经过攻关,终于研制出了这种涂料的配方。生产这种涂料需要用清铅粉、黑铅粉、粘土和糖浆四种原料,它们所占的比重分别为:35%、45%、14%和6%。该公司通过市场调查发现,该类型涂料具有竞争性的市场价格0.50美元/公斤,公司确定产品投放市场后的目标毛利为0.25美元/公斤。这样一来,A型涂料的目标成本即为0.25美元/公斤(0.50美元/公斤~0.25美元/公斤)。然而该公司通过市场调查得知:上述四种原料的成本分别为0.45美元/公斤、0.18美元公斤、0.15美元/公斤和1.00美元/公斤。因此,A型涂料的成本为:0.45×35%+0.18×45%+0.05×14%+1×6%=0.31美元/公斤。也就是说,这个设计方案虽然在技术上是可行的,但其成本却达不到目标成本的要求。为了实现既定的目标成本,该公司科技人员决定对A型涂料现有的配方进行重新研究调整,以便达到成本目标。通过运用价值工程的原理,公司技术人员发现A型涂料耐高温性能有些过剩,而悬浮稳定性却略显不足。为此,科技人员决定在保证A型涂料必要的功能的前提下改进配方。新配方只用清铅粉、黑铅粉和膨润土三种原料,它们所占的比重分别为15%、80%和5%,而膨润土的成本仅为0.09美元/公斤。这样新的A型涂料配方的成本为:0.45×15%+0.18×80%+0.09×5%=0.27美元/公斤。新配方的成本达到目标成本的要求,可以正式投产。

这一方法已经被众多的企业所采用,即实践证明它是一种十分有效的控制成本的手段。最初,这种方法可能是某企业迫于竞争的无奈而创造出来的,现在也主要在竞争激烈的行业中被广泛采用。但是,实际上在竞争并不激烈的产业中,推行此方法依然可以获得奇特的管理效果。人的潜力是无限的,有时候看似不能达到的目标,如果有一个强权者一定要让人们达到这个目标,有时还真的能够如愿以偿。许多企业往往并不知道自己企业是否存在降低成本的空间,采取这种方法,有时可以把海绵中所有的水都拧干。

4.基于价值分析的成本控制方法

一些优秀的制造业中的大企业都使用这种方法。这类企业往往设有一个专门的部门来负责“降低成本”,他们分析现有的工作、事项、材料、工艺、标准,通过分析他们的价值并寻找相应的替代方案,可以相应地降低成本。比如,某企业的成本管理人员经过认真分析,发现将企业内的保洁工作外包给公司以外的专业保洁公司完成,比企业自己养清洁工成本更低,于是提出议案,公司领导看后认为可行,于是就把公司的保洁工作委托给了一家专业保洁公司。

这种方法在先进的公司中进行是经常的和制度化的,即企业设有专门的人员(通常是工程师)以此为工作职责。但是,几乎所有的公司看似或多或少地使用了这种方法,但其实做得并不专业。具体分析会发现有两种情况:一是,一些企业所进行的价值分析实际上是学其他企业的经验。比如,听到或看到某企业将保洁工作实现外包,降低了保洁和管理成本,于是也来采取外包保洁的管理办法。这实际上是在运用标杆,而不是独立的价值分析的过程和结果。其二,一些企业经常也会特地对某些工作、事项、业务和流程等进行价值分析,并且有时也可能找出一个良好的替代方案,以及实行的效果的确比较理想。然而,这种价值分析的过程更多的是因为某些重要人物心血来潮时的行为。

5.基于经验的成本管理方法

这是一种最为基础的和较低级别的,但是应用最为普遍,在一定的条件下效果也是十分好的一种成本控制法。大多数企业的成本管理都是由此开始的,而其他每一种成本控制方法的最底层部分其实都是由此构成的。

它是管理者借助过去的经验来现实对管理对象进行控制,从而追求较高的质量、效率和避免或减少浪费的过程。比如说,经验告诉我们,在采购的过程中,“货比三家、反复招标、尽量杀价”,可以降低采购成本,于是管理者就要求他们的下属在采购时“货比三家、反复招标、尽量杀价”。又比如,经验告诫我们,对外采购的过程中,如果缺少必要的监督机制,有的采购人员就可能产生自私行为,从而导致企业损失,于是大量的企业常常不惜牺牲效率和成本设置“关卡”来防止采购人员的自私行为。还比如,人们注意到只要对员工盯紧一点,员工的工作效率就会得到相应的提高,于是企业都十分强调对员工行为的监督。

毫无疑问,基于经验的成本管理方法有时是最有效的提高效率、保证质量和控制成本的措施。一个从最基层销售员干起,一直干到营销副总经理职务的管理者,他所管理的销售人员,一般较少有机会犯直接蓄意损害企业利益的错误。然而,经验有时也是不可靠的。一位企业总经理过去管理文化层次较低的工作人员的经验告诉他,加强对犯错员工的处罚可能减少员工犯错,从而减少或避免企业的损失,然而如果他现在管理的工作人员文化层次较高,且都是独生子女一代,那么他的这一经验可能不但不管用,甚至走向反面。

基于经验的成本管理法有时并不管用,一般出于两点原因:一是,经验带有严重的个人色彩,当变化的环境问题超过经验的范围时,经验可能失去效用。二是,经验往往是“就事论事”的,不是系统思维的结果,因此经验在实用过程中可能出现系统性的消极后果,即对具体的对象而言它们有助于控制甚至降低成本,但就总体而言它们则可能无助于控制成本,甚至造成系统性成本上升。此外,实施经验化的成本管理,可能在未来留下历史的阴影。

2.4 项目进度管理

进度是IT项目的一个关键因素，项目进度控制和监督的目的是增强项目进度的透明度，以便当项目进展与项目计划出现严重偏差时，可以采取适当的纠正措施。已经归档和发布的项目计划是项目控制和监督中活动、沟通、采取纠正和预防措施的基础。

2.4.1 影响项目进度的因素

要有效地进行进度控制，必须对影响进度的因素进行分析，事先或及时采取必要的措施，尽量缩小计划进度与实际进度的偏差，实现对项目的主动控制。IT项目中影响进度的因素很多，如人为因素、技术因素、资金因素和环境因素等，其中人为因素是最重要的因素，技术因素归根到底也是人的因素。常见的有以下几种情况：

1.低估IT项目实现的条件

(1)低估技术难度。IT项目的高技术本身，说明其实施中会有很多的技术难度，除了需要高水平的技术人员实施外，还要考虑为解决性能问题而进行科研攻关和项目实验。实际中，项目主管通常会低估项目技术上难度，使得项目不能按照项目计划顺利实施。

(2)低估了多个项目团队参加项目时工作协调的复杂度和难度。IT项目团队成员通常比较强调个人的智慧，强调个性，这就会增加需要团队合作的项目协调工作的复杂度。尤其是对于由很多子项目组成的大项目来说，更会增加项目协调和进度控制上的难度。

(3)低估环境因素。企业项目主管和项目经理也经常低估环境因素，没有充分了解项目情况，从而低估了项目实现的条件。

2.项目参与者的失误

每个项目参与者的行为都会影响到项目进度。若项目计划本身有错误，如计划制定者在系统框架设计上的错误、成本预算编制的错误等，那么执行过程也难免出错，进而对进度产生影响。即使项目计划正确，项目执行上也一样会出现错误，如项目所需款项没有到位、关键人员离职等，都会影响项目进度。此外，如果没有很好地管理项目转包的部分，也会造成进度的延误。

3.不可预见事件的发生

项目会因为一些不可预见的事件，如计划中要采购的设备没有货等，对项目进度产生影响。

通常项目进入实施阶段之后，项目经理所关注的活动都是围绕进度展开的。进度控制的目标与成本控制和质量控制的目标是对立统一的关系，三者互相制约。比如，一般情况下，进度快需要增加投资，而项目的提前完工可能会提高投资效益；进度快有可能会影响质量，而严格控制质量又有可能影响进度，有时也有可能因为严格控制质量而不出现返工，反而加快了进度。这三个目标是一个系统，存在于一个统一体中，因此必须协调和平衡它们之间的关系才能得到全局的最优解。项目经理需要系统地考虑三者之间的制约关系，既要进度快，又要成本省、质量好，使三个目标的控制达到最优。

4.项目状态信息收集的情况

离开了信息，对项目进行成功的控制就是无源之水、无本之木。由于项目经理的经验或素质原因，对项目状态信息收集掌握不足，会造成项目的及时性、准确性、完整性比较差。比如，某些项目团队成员报喜不报忧，在软件程序的编制过程中，可能会先编制一些表面的东西，给领导造成比较乐观的感觉，而实际上只是一个“原型系统”或演示系统。如果项目经理或者管理团队没有及时地发现这种情况，将对项目的进度造成严重的影响。

5.计划变更调整的及时性

IT 项目不是一个一成不变的过程，开始时的项目计划可以制定得比较粗一些，随着项目的进展，特别是需求明确以后，项目的计划就可以进一步地明确，这时候应该对项目计划进行调整修订，通过变更手续取得利益相关者的共识。计划应该随着项目的进展而逐渐细化、调整和修正。没有及时调整计划或者是随意的、不负责任的计划是难以控制的。

IT 项目计划的制定需要在一定条件的限制和假设之下采用渐近明细的方式，随着项目的进展进行不断细化、调整、修正和完善。对于较为大型的 IT 项目的工作分解结构可采用二次甚至多次 WBS 方法。由于需求的功能点和设计的模块或组件之间并不是一一对应的关系，所以只有在概要设计完成以后才能准确地得到详细设计或编码阶段的二次 WBS，根据代码模块或组件的合理划分而得出的二次 WBS 才能在详细设计、编码阶段乃至测试阶段起到有效把握和控制进度的作用。有些项目的需求或设计做得不够详细，无法对工作任务的分解、均衡分配和进度管理起到参考作用，因此要随着需求的细化和设计的明确，对项目的分工和进度进行及时的调整，使项目的计划符合项目的变化，使项目的进度符合项目的计划。

2.4.2　项目进度控制

项目进度控制应该以项目进度计划为依据。项目经理通过各种有效沟通途径，获取项目进度的实际信息，比照进度计划基准，识别进度方面是否存在偏差。如果发现偏差，要分析偏差形成的原因，以及该偏差对项目总体影响程度，同时要确定偏差是否可以接受。如果利益相关者认为偏差可以接受，则可以调整项目进度计划，反之，则需要制定具体的措施，将偏差纠正到可以接受的范围之内。纠正措施合并到项目的总体计划中，一起进行跟踪管理。

由此可以看出，项目进度偏差是项目进度控制的重要依据。如果不能及时有效地管理项目进度偏差，可能会对项目造成很大的影响。比如，项目延期引起客户满意度的下降，影响了和客户的继续合作；人力资源投入时间延长，增加了项目成本，并间接影响到其他项目的实施；项目延期导致团队情况消极，士气低落，生产率下降；不能及时收回款项，降低资金流动速度；公司名誉受到影响等。这些结果往往比进度偏差本身严重得多。

1.识别偏差

在进行 IT 项目进度控制时，应检查是否存在偏差，通常从分析项目关键路径上的任务是否存在偏差开始，然后检查项目近关键路径上的任务，再检查其他任务的进展情况。通常项目团队使用项目进度管理一览表来管理项目实施过程中出现的进度偏差，如表 2-1 所示。在表的最左边，将出现偏差的任务分为三类：关键路径上任务、近关键路径上任务和非关键路径上任务。在每一类中，按照进度偏差的大小进行排序。

表 2-1 项目进度管理一览表

项目名称：										
项目编号：							项目经理：			
本文件所依据计划基准文件名称及编号：										
文件编号：						发布日期：		发布人：		
WBS编号	任务名称	计划日期	实际日期	进度偏差	偏差原因	影响	偏差是否可以接受，若不能接受，提供纠正措施	计划解决日期	实际解决日期	责任人
关键路径上任务										
近关键路径上任务										
非关键路径上任务										

2.分析偏差原因

识别了进度偏差之后，项目经理应该与存在进度偏差的任务的负责人一起分析偏差原因。如果任务责任人对偏差原因没有清晰的认识，项目经理应该帮助其分析探讨。通常将鱼刺图作为分析原因的工具，如图 2-4 所示。

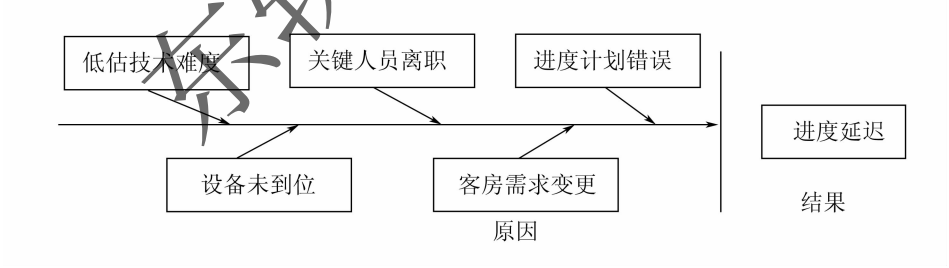

图 2-4 进度偏差原因分析

在 IT 项目中，造成项目进度延误的原因主要包括项目团队内部原因、项目执行组织的原因、客户原因和外部原因 4 个方面。

项目团队内部原因主要包括：进度计划本身存在漏洞；人员技能达不到预想水平；团队士气不高、内部沟通不畅；协作配合不力；项目技术不成熟等。项目执行组织的原因，如组织中管理层或者其他职能部门的不支持，使得项目所需资源不能及时到位；流程不合理影响项目进展等。客户原因，如客户没有按期准备所需事项；客户配合不力；客户需求更改过多等。外部原因，如分包商配合不力导致项目延误；以及一些不可控因素导致进度拖延等。

只有真正了解进度偏差发生的原因，才能确定对偏差的态度，决定是否需要采取措施。

3.确定对既发偏差的态度

项目经理和团队应该重点关注项目进度管理一览表中排在前面的偏差。如果接受了某一项进度偏差，并因此调整进度计划时，项目的关键路径和近关键路径通常会发生变化，相应地，位于关键路径和近关键路径上的任务也会发生变化。

并非所有的偏差都需要采取纠正措施。比如，一些发生在非关键路径上的偏差，如果其小于此路径上的总浮动时间，不会直接导致项目整体进度的延误，这时可以接受此偏差，不采取纠正措施。当然如果发生在非关键路径上的偏差大于此路径上的浮动时间，那么该路径就成了项目的关键路径，这时如果不采取措施就会影响项目的总进度。

一般来说，项目本身通常留有一定的进度冗余。当存在较大的进度冗余时，在进度偏差不超过冗余的情况下，利益相关者可以容忍一定的进度偏差。但是如果进度偏差发生在项目实施早期，项目团队为了给以后的实施留出足够的冗余来抵御不可预见的风险，往往会纠正早期出现的进度偏差。

接受偏差一定要取得客户、公司管理层等关键利益相关者的认可，并及时与相关任务的负责人进行协调，避免潜在的人力资源冲突。

如果不接受偏差，就要及时制定纠正措施。例如，把非关键路径上的资源转移到关键路径上；投入更多的人力资源或加班赶进度，提高生产效率；使用项目中预留的时间冗余；更换不称职的团队成员；与客户主动沟通争取客户的谅解和配合；将组织内部问题升级以取得管理层注意和支持等，都是常常使用的措施。当面对具体的进度偏差时，应该尽可能地找出所有可能的纠正措施，项目经理与项目团队成员、利益相关者、专家等一起从技术可行性、经济可行性、是否易于维护、是否会对项目其他任务造成影响等角度来进行筛选，并选择合适的纠正措施。

4.关注进度正偏差

当项目团队成员的工作效率提高、技术改进时，进度会出现正偏差，表明项目进度超前，这时应该提出表扬，并推广其经验。但并不是所有的进度正偏差都是积极的，尤其是当出现较大幅度的正偏差时，一定要谨慎分析原因。比如，项目进度计划的编制是否有误；项目范围是否有所遗漏；项目的质量是否得以保证；项目进度报告是否有误；是否投入了过多的资源导致成本的迅速上升等。这些因素都预示了项目的潜在风险，需要采取必要的措施。

5.调整项目进度计划

在项目实施过程中，难免会调整进度计划。通常对项目进度计划进行调整，一般有以下几种情况：

(1)客户由于业务或需求变化提出变更申请，经批准后对进度计划进行相应调整。

(2)项目团队由于客观条件变化或者设计了更好的方案，提出变更申请，经批准后对进度计划进行相应调整。

(3)计划本身存在缺陷需要修正，应及时调整计划，并尽最大可能减少调整计划造成的负面影响。

(4)关键利益相关者接受项目进度出现的偏差，对进度计划进行相应的调整。

(5)纠正项目其他方面的偏差引起项目进度的偏差，经批准后对进度计划进行相应调整。

(6)随着项目的实施和内容的细化,对进度计划进行调整,以增加更加详细的内容。

对项目进度计划的调整,可能引起项目关键路径发生转移。因此每当对计划进度调整之后,项目经理者需要重新确定关键路径。

2.5 项目配置管理

软件开发过程有许多的资料,例如需求分析说明、设计说明、源代码、可执行码、用户手册、测试用例、测试结果等文档;还有合同、计划、会议记录、报告等管理文档。软件开发过程中出现变更也是不可避免的。如何有序高效地产生、存放、查找和利用如此庞大且不断变动的资料,确保在需要的时候能够及时获得正确的资料,尽可能少地出现混乱和差错成为软件工程项目十分突出的问题。软件配置管理正是为解决这个问题而提出的,它为软件开发提供了一套管理办法和活动原则,是软件开发过程质量保证活动的重要一环。

配置管理包括项目源代码、文档的版本控制与管理;标识软件配置项,建立产品基准库;对配置项的修改加以系统的控制等。软件有一种进化的本性,从一个软件产品的定义一直到它被停止使用的过程中会经历许多变更。每一次变更的结果就是该产品的一个新版本。配置管理的目的就是在初始、评估、以及执行这些变更的同时维护产品的完整性。它提供了一个理性的框架来处理包括用户需求和资源限制的非理性世界。配置管理的目的就是标识每一软件项、管理并控制软件项的变更、便于追踪各软件项和后期维护。

2.5.1 配置管理的意义

在质量体系的诸多支持活动中,配置管理处在支持活动的中心位置。它有机地把其他支持活动结合起来,形成一个整体,相互促进,相互影响,有力地保证了质量体系的实施。

同发达国家相比,我国的软件企业在开发管理上,过分依赖个人的作用,没有建立起协同作战的氛围,没有科学的软件配置管理流程;在技术上只重视系统和数据库及开发工具的选择,而忽视配置管理工具的选择,导致即使有配置管理的规程,也由于可操作性差而搁浅。以上种种原因导致开发过程中普遍存在如下一些问题:

(1)开发管理松散。部门主管无法确切得知项目的进展情况,项目经理也不知道各开发人员的具体工作,项目进展随意性很大。

(2)项目之间沟通不够。各个开发人员各自为政,编写的代码不仅风格各异,而且编码和设计脱节。开发大量重复代码,留下大量难维护的代码。

(3)文档与程序严重脱节。软件产品是公司的宝贵财富,代码的重用率相当高,如何建好知识库,用好知识库对开发优质高效的产品,具有重大的影响。如果程序既无像样的文档,开发风格又不统一,这会给系统维护与升级带来极大的困难。

(4)测试工作不规范。传统的开发方式中,测试工作只是人们的一种主观愿望,根本无法提出具体的测试要求,测试工作往往是走过场,测试结果既无法考核又无法量化,当然就无法对以后的开发工作起指导作用。

(5)施工周期过长,且开发人员必须亲临现场。由于应用软件的特点,各个不同的施工

点有不同的要求，开发人员要手工地保持多份不同的拷贝，即使是相同的问题，但由于在不同地方提出，由不同人解决，其做法也不同，程序的可维护性越来越差。

通过科学的配置管理能够大大地改善软件开发的环境与软件开发的效益，能够节约费用、缩短开发周期，有利于知识库的建立及规范管理，从而保证软件开发工程能够在规定的时间内保质、保量地完成，开发出易于维护、易于扩展的软件。有效的配置管理可以帮助我们提高软件产品质量、提高开发团队工作效率。很多软件企业已经逐渐认识到配置管理的重要性，在国外一些成熟的配置管理工具的帮助下，制订相应的配置管理策略，取得了很好的成效。

软件配置管理的目标是：规划软件配置管理活动；经由选择的软件工作产品能够被识别、控制及可获取；对被识别的软件工作产品的调整进行控制；相关的组和个人能获知软件基准的情况和具体内容。

软件配置管理方面的工作包括：在指定的时间及时确定软件的配置（比如软件产品和它们的描述）；在整个软件生命周期中系统地控制这些配置的调整，并维持其完整性和可跟踪性；被置于软件配置管理之下的工作产品包括发布给用户的软件产品（比如软件需求文档和代码）以及创建软件产品所必需的内容（比如编译器）。

2.5.2　配置管理的实施过程

软件配置管理活动在整个开发活动中是一项支持性、保障性的工作，它本身并不直接为企业产出可以直接赢利的工作成果；而配置管理每一项活动都需要消耗企业的人力资源，有些还需要购置专门的工具来支持活动的进行，这些都会导致企业生产成本的增加。所以，在计划实施配置管理时，要小心地界定每一项活动。取舍的标准是：从事这项活动是不是真正有助于实施活动的成功？它对于提高产品的质量有多大的帮助？能否帮助开发团队更高效率地工作？配置管理的实施要注意以下几个方面的问题。

1.评估开发团队当前配置管理现状

对于本团队管理现状的评估，可以自己进行，也可以引入外部专业咨询人员来完成。引入外部专业咨询人员进行评估有两个好处：一是通常这样的咨询人员有比较丰富的配置管理实施经验，评估工作可以进行得更好更细致，而且通常咨询人员会在评估结果的基础上提出实施的建议；二是引入外部人员，通常评估结果会比内部自我评估更客观。坏处是要花钱。不管以何种方式进行，评估这个步骤的工作是一定要仔细进行的。有了评估的结果，才谈得上改进。

2.定义实施的范围

对于没有正式实施过软件配置管理的开发团队，在配置管理方面存在的问题可能会比较多，经过评估，会找出来很多需要改进的点。那么，怎样来计划改进的工作步骤呢？原则就是利用管理学中的黄金法则抓住团队最头疼的几个问题，努力想办法解决这些问题。能找出 20％对软件开发带来 80％的困扰和痛苦的问题，然后集中 80％的精力来解决这些问题。流程改进是一个持续的历程，一个阶段会有一个阶段改进的重点，抓住重点、做出成绩，才是有效的改进之道。

3.计划资源要素

具体来说，配置管理实施主要需要两方面的资源要素：一是人力资源；二是工具。人力

方面，因为配置管理是一个贯穿整个软件生命周期的基础支持性活动，所以配置管理会涉及到团队中比较多的人员角色，比如，项目经理、配置管理员、开发人员、测试人员、集成人员、维护人员等。但是，工作在一个良好的配置管理平台上并不需要开发人员、测试人员等角色了解太多的配置管理知识，所以，配置管理实施的主要人力资源集中在配置管理员上。配置管理员对一个实施了配置管理，建立了配置管理工作平台的团队来说，是非常重要的。整个开发团队的工作成果都在他的掌管之下，他负责管理和维护的配置管理系统，如果出现问题的话，轻则影响团队其他成员的工作效率，重则可能出现丢失工作成果、发布错误版本等严重的后果。在国外一些比较成熟的开发组织中，对于配置管理员都很重视。在选拔配置管理员的时候，也有相当高的要求，比如，有一定的开发经验，对于系统（操作系统、网络、数据库等方面）比较熟悉，掌握一定的解决问题的技巧，在个人性格上，要求比较稳重、细心。

在配置管理员这个资源配置方面，要注意后备资源的培养。在大家越来越重视配置管理的大环境下，经验丰富的配置管理员会成为抢手的人才；而配置管理员的离开可能会给团队的工作进度带来一定的影响，所以聪明的管理者会为自己留好备份。

选择什么样的配置管理工具一直是大家关注的热点问题，配置管理工作更强调工具的支持。在配置管理工具的选型上，可以综合考虑下面的一些因素。一般来说，如果经费充裕的话，采购商业的配置管理工具会让实施过程更顺利一些，商业工具的操作界面通常更方便一些，通常与流行的集成开发环境（IDE）也会有比较好的集成，实施过程中出现与工具相关的问题也可以找厂商解决。但如果经费有限，不防采用自由软件（如 CVS 之类的工具）。如果准备选择商业配置管理工具，就应当重点考虑下面几个因素：

（1）工具的市场占有率。大家都选择的东西通常会是比较好的东西；而且市场占有率高也通常表明该企业经营状况会好一些，被人收购或者倒闭的可能性小一点。

（2）工具本身的特性，如稳定性、易用性、安全性、扩展能力等。在投资以前应当仔细地对工具进行试用和评估。比较容易忽略的是工具的扩展能力（Scalability）。

（3）厂商的支持能力。工具使用过程中出现的问题，有些是因为使用不当引起的，有些则是工具本身的毛病。这样的问题会影响到开发团队的工作进度，要确保找到厂商的专业技术人员帮助解决这些问题。

配置管理工具不是用一次两次的工具，因此，选择配置管理工具其实是选择和哪个厂商来建立一种长期的关系，所以一定要慎重选择。

4.建立有关的数据库

（1）建立代码知识库实现对程序资源进行版本管理和跟踪。保存开发过程中每一过程版本，这样大大提高了代码的重用率，还便于同时维护多个版本和进行新版本的开发，防止系统崩溃，最大限度地共享代码。同时项目管理人员可以查看项目开发日志；测试人员可以根据开发日志和不同版本对软件进行测试；工程人员可以得到不同的运行版本，并且供外地施工人员存取最新版本，无需开发人员新临现场。科学地应用知识库可以大大提高开发效率，避免了代码覆盖、沟通不够、开发无序的混乱局面。如果利用了公司原有的知识库，则更能提高工作效率，缩短开发周期。

（2）建立业务及经验库。通过配置管理可形成完整的开发日志及问题集合，以文字方式伴随开发的整个过程，不依某个人的转移而消失，有利于公司积累业务经验，无论对版本整

改或升级，都具有重要的指导作用。

(3)建立代码对象库。软件代码是软件开发人员脑力劳动的结晶，也是软件公司的宝贵财富，长期开发过程中形成的各种代码对象就像一个个零件坯一样，是快速生居系统的组成部分。许多组织的现实是一旦某个开发人员离开工作岗位，其原来所做的代码便基本成为垃圾，无人过问。究其原因，就是没有专门对各人的有用对象进行管理，把其使用范围扩大到公司一级，进行规范化，并加以说明和普及。建立代码对象库就能解决这些问题。

5.建立开发管理规范

把版本管理档案挂接在公司内部的 Web 服务器上，工程人员可获取所需的最新版本。开发人员无需下现场，现场工程人员通过对方系统管理员收集反馈意见，直接提交到公司内部开发组的项目经理，开发组内部讨论决定是否修改，并作出书面答复。这样可以同时管理多个项目点，克服开发人员分配到各个项目点、分散力量、人员不够的弊端，同时节约大量的旅差费用。规范管理能带来以下好处：量化工作量考核，传统的开发管理中，工作量一直是难以估量的指标，靠开发人员自己把握，随意性相当大；靠管理人员把握，主观性又太强；规范测试，测试工作人员根据每天的修改细节描述对每一天的工作做具体的测试，对测试人员也具有可考核性，这样环环相扣，大大减少了其工作的随意性；加强协调与沟通，通过文档共享及其特定机制与电子邮件的集成，大大加强了项目成员之间的沟通，做到有问题及时发现、及时修改、及时通知，但又不额外增加很多的工作量。

6.建立基准

经过正式评审和认可的一组配置项(文档和其他软件工作产品)，可以作为进一步开发的基础，并且只有通过正式的更改控制规程才能被更改。

7.更改控制

在合同阶段与客户明确系统更改的控制方法，以确保系统的成功实施。因客户的业务需求变更而进行更改时应有客户的确认，以防开发人员的软件项的随意更改。在各项目开发结束后，所有代码和文档备份到专用的代码备份服务器归档。后期的维护作为新任务的开始，定期整理维护活动产生的结果，追加到原项目的备份中去，同时更新配置状态报告。

在初期的软件开发过程中人们常常忽略文档的管理，往往认为程序是软件的核心，而文档则是可有可无的，对文档不重视也是产生软件危机的一个原因。所幸的是，现在人们对文档的重要性已经有某种程度的认识。文档是用来表示对活动、需求、过程或结果进行描述、定义、规定、报告或认证的书面信息。它们描述和规定了软件设计和实现的细节，说明使用软件的操作命令，是软件使用、升级和维护的最重要的依据。文档是软件的一部分，没有文档的软件是一个不完整的软件。软件在整个软件生存期中，各种文档作为半成品或是最终成品，会不断地生成、修改或补充。为了最终得到高质量的产品，达到质量要求，必须加强对文档的管理。在文档管理方面要注意以下几点。

(1)软件开发小组应设一位文档保管人员，负责集中保管本项目已有文档的两套主文本。两套文本内容完全一致，其中的一套可按一定手续，办理借阅。

(2)软件开发小组的成员可根据工作需要在自己手中保存一些个人文档。这些一般都应是主文本的复制件，并注意和主文本保持一致；在做必要的修改时，也应先修改主文本。

(3)开发人员个人只保存着主文本中与他工作相关的部分文档。

(4)如果有新文档取代旧文档时,管理人员应及时注销旧文档。如果文档内容有变动时,管理人员应随时修订主文本,使其及时反映更新了的内容。

(5)项目开发结束时,文档管理人员应收回开发人员的个人文档。发现个人文档与主文本有差别时,应立即着手解决。这常常是未及时修订主文本而造成的。

(6)在软件开发过程中,可能发现需要修改已完成的文档,特别是规模较大的项目,主文本的修改必须特别谨慎。修改之前要充分估计修改可能带来的影响,并且要按照提议、评议、审核、批准和实施等步骤进行严格的控制。在整个软件生存期中,各种文档作为半成品或是最终成品,会不断地生成、修改或补充。为了最终得到高质量的产品,达到质量要求,必须加强对文档的管理。

2.5.3 配置控制

为配合整个软件开发过程的管理,保证各阶段成果有完备、一致、可追踪性和技术状态的可控制性,在整个产品实现过程中标识、组织和控制修改项,需要在软件产品开发和维护过程中实施软件配置管理活动。

软件配置管理(Software Configuration Management,SCM)的真正含义可以从以下角度理解和掌握:

(1)《ISO/IEC12207(1995)信息技术——软件生存期过程》:配置管理过程是在整个软件生存期中实施管理和技术规程的过程,它标识、定义系统中软件项并制订基线;控制软件项的修改和发布;记录和报告软件项的状态和修改申请;保证软件项的完整性、协调性和正确性;以及控制软件项的存储、装载和交付。

(2)《ISO9000-3(1997)质量管理和质量保证标准——第3部分:ISO9001:1994在计算机软件开发、供应、安装和维护中的使用指南》:软件配置管理是一个管理学科,它对配置项的开发和支持生存期给予技术上和管理上的指导。配置管理的应用取决于项目的规模、复杂程度和风险大小。

(3)巴比齐(W. Babich):软件配置管理能协调软件开发,使得混乱减少到最小。软件配置管理是一种标识、组织和控制修改的技术,目的是最有效地提高生产率。

(4)《GB/T11457(1995)软件工程术语》:软件配置管理是标识和确定系统中配置项的过程,在系统整个生存周期内控制这些项的投放和变动,记录并报告配置的状态和变动要求,验证配置项的完整性和正确性。

总之,软件配置管理是指通过在软件生命周期的不同时间点上对软件配置进行标识,并对这些被标识的软件配置项的更改进行系统控制,从而达到保证软件产品的完整性和可溯性的过程。

为了达到上述目的,SCM必须完成下面4项工作:

(1)配置标识。配置标识是软件生命周期中选择定义各类配置项、建立各类基线、描述相关软件配置项及其文档的过程。首先,软件被分成一系列软件配置项,一旦各配置项和它们各自应包含的内容被选定,就制订一套框架方案,包括对代码、数据、文档进行命名。然后,对这些配置项的功能、性能和物理特性生成描述文档。在配置管理系统中,基线就是一个配置项或一组配置项在其生命周期的不同时间点上通过正式评审而进入正式受控的一种

状态，是产品中所有模块的配备（版本集）。

（2）配置控制。即控制对配置项的修改，要对配置项的变更申请进行初始化、评估、协调、实现，包括将通过和实现的变更加入到基线中的更改控制过程。更改控制确保各类变更被正式地初始化、分类、评估、批准/或不批准。获批准的变更请求将得到正式的实现、记录和验证。

（3）配置状态发布。配置状态发布是跟踪对软件更改的过程，它保证对正在进行和已完成的变更进行记录、监视并通报。

（4）配置评审。确认受控软件配置项满足需求并就绪。配置评审是验证一个可发布的软件基线是否包含了它应包括的所有内容，通常包括两类评审：功能配置评审（FCA）和物理配置评审（FCA）。FCA 确认软件已通过测试并满足基线规定的需求说明，即确保配置的正确性；FCA 确认将发布的软件包含了所有必需的组成部分，包括代码、文档、数据等，确保配置的完整性。

模块代码通常以 3 种形式存在：源代码（现在常使用 C++、Java 或 Ada 等高级程序语言编写）、目标代码（通过编译源代码生成）和可执行载入映像（目标代码与运行时例程结合），如图 2-5 所示。程序员可使用每个模块的多种不同版本。软件配置是指一个软件产品在软件生存周期各个阶段所产生的各种形式（机器可读或人工可读）和各种版本的文档、程序及其数据的集合。该集合中的每一个元素称为该软件产品软件配置中的一个配置项（Configuration Item）。

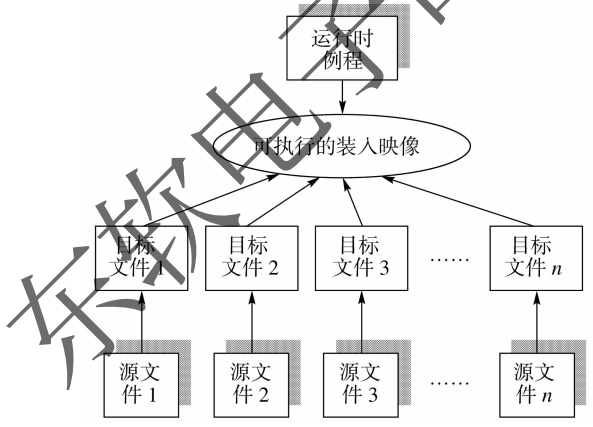

图 2-5　模块代码的 3 种存在形式

需要指出，“配置”和“配置项”是两个不同的概念。“配置”是在技术文档中明确说明并最终组成软件产品的功能或物理属性。因此，它包括了即将受控的所有产品特性、相关文档、软件版本、变更文档、软件运行的支持数据，以及其他一切保证软件一致性的组成要素。相对于硬件类配置，软件产品的“配置”包括更多的内容并具有易变性。

受控软件经常被划分为各类“配置项”，这类划分是进行软件配置管理的基础和前提。“配置项”是逻辑上组成软件系统的各组成部分，比如一个软件产品包括几个程序模块，每个程序模块及其相关文档和支撑数据可能被命名为一个配置项。如果一个产品同时包括硬件和软件部分，那么配置项也同时包括硬件和软件部分。一个纯软件的配置项通常也称之为软件配置项。软件和硬件的配置管理有一些类似的地方，但因为软件更易于修改，所以软件配置管理是一个更应该系统化的过程。

接受SCM过程控制的软件受控“配置项”应包括一切可能对软件产品的完整性和一致性造成影响的组成要素，比如项目文档、产品文档、代码、支撑数据、项目编译建立环境和项目运行环境等。所有这些可以由同一套SCM过程统一管理。

一般地，软件开发过程从概念演绎和需求分析开始，然后是设计、各配置项的编码或写作、集成测试，最后是用户手册的编写等。软件配置管理包括了软件生命周期的时间分散点上对各配置项进行标识并对它们的修改进行控制的过程。在一个开发阶段结束或一组功能开发完成后，要对相应的配置项进行基线化并形成各类基线。

在进行软件测试时，为了重现模块在某组测试数据上的问题，必须使用一个配置控制工具（否则需要查看二进制的可执行载入映像才能指出错误源头）来确定是模块哪个变种版的哪个修订版进入了该产品失效的哪个配置版本。所以，处理多个版本时必须解决两个问题：区分版本以便将每个模块的正确版本编译并链接到产品中；对一个可执行映像还要确定每个组件的哪个版本进入它了。

解决这个问题首先要有版本控制工具。许多操作系统支持版本控制（尤其是大型计算机的操作系统）。对于不支持版本控制的操作系统可以使用一个单独的版本控制工具。在版本控制中，每个文件的名称常常包含两个部分：文件名本身和修订版本号。程序员借此可以准确指明为完成某任务需要哪一个修订版。

版本控制在管理模块的多个版本和整体产品方面的帮助很大，但是在维护V个变种版本时，如果其中一个发现了错误，则全部V个变种版都需要恢复。较好的处理方法是只存储一个变种版，然后其他任何变种版都根据从最初的版本到那个变种版本所做改变的存储列表。这些差异的列表被称为“增量”，所以，存储的只是一个变种版和V－1个增量，通过访问最初的变种版并应用不同的增量就可以得到其他变种版。然而对最初的变种版的任何修改都会自动应用到所有其他的变种版中。

使用配置控制工具，不仅可以自动管理多个变种版，而且还能处理小组开发和维护时出现的其他问题。例如，在产品维护期间的配置控制中，如果多个程序员同时维护一个产品，就会出现修复错误的变更不同步问题。对于团队维护来说，应该每次只允许一个用户修改模块。配置控制不仅在维护阶段需要，在产品开发过程中的实现阶段和集成阶段也需要。管理者为了充分监控开发过程，有可能就会使用配置控制来管理，这样就能掌握每个模块的状态。一个可用的软件配置状态报告与清单如表2-2、表2-3所示。

表2-2 软件配置状态报告

项目名称		项目编号		项目周期		开发环境	
报告期间		所属部门		报告编制		流水号	

表2-3 软件配置状态清单

序号	变更记录流水号	已变更配置项	相关变更	变更状态	变更原因	变更时间	变更者	备注

常用的软件配置管理工具如下：

(1)Source Integrity(SI)：版本管理工具

(2)Track Integrity：问题跟踪、变更管理工具

(3)Rational 公司的产品

(4)ClearCase：版本控制工具

(5)ClearQuery：变更管理工具

(6)微软的 Studio Package 中带的 VSS

(7)UNIX 版本控制工具

①SCCS：源代码控制系统

②RCS：修订版控制系统

③CVS：并行版本控制系统

(8)较早被使用的版本管理工具：PVCS

此外，还使用辅助工具来进行配置管理。配置库系统是实现配置管理的一种辅助自动化工具，它提供对配置项的增量式存储和对各类流程(比如变更控制流程、配置状态发布)的支持。项目应根据其规模和配置管理的复杂程度使用专门的配置管理系统建立配置库系统。

为了体现对配置项的分层控制，在逻辑上可将配置库分为 3 类：基线库、开发库和产品库。

(1)基线库。包含通过评审的各类基线及变更统计数据。

(2)开发库。即程序员的工作空间，始于某一基线，为某一目的的阶段性开发服务，最终通过正式的评审过程归并到某一基线，回归到基线库。

(3)产品库。保存各基线的静态复件。基线库进入发布阶段形成产品库，可以在产品数据库中形成相应的复件。

通常情况下，用软件配置管理工具的分支和合并功能实现基线库、开发库和产品库的分离。版本树主干作为基线库，在进行功能增强或错误、缺陷修改时，建立相应的版本分支作为开发库，修改完成后归并到基线库中。

上面提到的基线按照软件开发的不同阶段，可以分为下面几种类型：

(1)功能基线(Functional Baseline)。是指在系统分析与软件定义阶段结束时，经过正式评审和批准的系统设计规约书中对待开发系统的规约；或是指经过项目委托单位和项目承办单位双方签字同意的协议书或合同中所规定的对待开发软件系统的规约；或是由下级申请经上级同意或直接由上级下达的项目任务书中所规定的对待开发软件系统的规约。功能基线是最初批准的功能配置标识。

(2)指派基线(Allocated Baseline)。是指在软件需求分析阶段结束时，经过正式评审和批准的软件需求的规约。指派基线是最初批准的指派配置标识。

(3)产品基线(Product Baseline)。是指在软件组装与系统测试阶段结束时，经过正式评审和批准的有关所开发的软件产品的全部配置项的规约。产品基线是最初批准的产品配置标识。

2.5.4 配置管理报表

在系统的运行与维护过程中，还要注意一些常用的配置管理报表及其格式。

1.软件问题报告单(SPR)

在系统的运行与维护阶段对软件产品的任何修改建议，或在软件开发的任一阶段中对前面各个阶段的阶段产品的任何修改建议，都应填入软件问题报告单。软件问题报告单的格式如表 2-4 所示。

表 2-4　　软件问题报告单(SPR)

<table>
<tr><td colspan="9" rowspan="3">软件问题报告单</td><td colspan="3">登记号(A)</td><td colspan="4"></td></tr>
<tr><td colspan="3">登记日期(B)</td><td colspan="4">年　月　日</td></tr>
<tr><td colspan="3">发现日期(C)</td><td colspan="4">年　月　日</td></tr>
<tr><td colspan="2">项目名(D)</td><td colspan="3"></td><td colspan="2">子项目名(E)</td><td colspan="2"></td><td colspan="3">代号(F)</td><td colspan="4"></td></tr>
<tr><td>阶段名</td><td>软件定义</td><td>需求分析</td><td>概要设计</td><td>详细设计</td><td>编码测试</td><td>组装测试</td><td>安装验收</td><td>运行维护</td><td>状态</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="2" rowspan="2">报告人</td><td colspan="2">姓名</td><td colspan="4"></td><td colspan="3">电话</td><td colspan="4"></td></tr>
<tr><td colspan="2">地址</td><td colspan="12"></td></tr>
<tr><td colspan="2">问题(G)</td><td colspan="14">例行程序　程序　数据库　文档　改进</td></tr>
<tr><td colspan="2">子例行程序/子系统:(H)</td><td colspan="3"></td><td colspan="3">修改版本号:(I)</td><td colspan="3"></td><td colspan="2">媒体(J)</td><td colspan="2"></td></tr>
<tr><td colspan="2">数据库:(K)</td><td colspan="3"></td><td colspan="3">文档:(L)</td><td colspan="3"></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td colspan="2">测试实例:(M)</td><td colspan="3"></td><td colspan="3">硬件:(N)</td><td colspan="3"></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td colspan="16">问题描述/影响:(O)</td></tr>
<tr><td colspan="16">附注及修改建议:(P)</td></tr>
</table>

(1)配置管理人员填写内容

表中 A、B、C、P 和状态等项目是由负责修改控制的配置管理人员填写的，其他各项即 D、E、F、G、H、I、J、K、N 和 O 各项是由发现问题的人或申请配置管理的人填写的，除此之外他还可能要填写 L 和 M 三项内容。前 4 项内容的意义如下：

①A 是由配置管理人员确定的登记号，一般按报告问题的先后顺序编号。

②B 是由配置管理人员登记问题报告的日期。

③C 是发现软件问题的日期。

④P 是填写若干补充信息和修改建议。

(2)配置管理状态

状态一栏分成 7 种情况,分别为:软件问题报告正被评审,已确定采取什么行动;软件问题报告已由指定的开发人员去进行维护工作;修改已经完成、测试好,正准备释放给主程序库;主程序库已更新,主程序库修改的重新测试尚未完成;已经进行了复测,但发现问题仍然存在;已经进行了复测,已经顺利完成所做的修改,软件问题报告单被关闭(维护已完成);留待以后关闭,因问题不是可重产生的,或者是属于产品改善方面的,或者只具有很低的优先级等。

(3)配置管理申请人员填写的内容

在软件问题报告单中,属于配置管理申请人填写的各项内容的意义如下:

①D、E 两项是项目和子项目的名称,F 是该子项目的代号。这应按配置标识的规定来命名代号。

②阶段名和报告人的姓名、住址和电话等的含义显而易见。

③G 表示问题属于哪一方面,是程序的问题还是例行程序的问题,是数据库的问题还是文档的问题,是功能适应性修改还是性能改进性修改问题,也可能是它们的某种组合。

④H 表示子例行程序/子系统,即要指出出现问题的子例行程序名字。如果不知是哪个子例行程序,可标出子系统名,总之,尽可能给出细节。

⑤I 是修订版本号,指出出现问题的子例行程序版本号。

⑥J 是媒体,表示包含有问题的子例行程序的主程序库存储媒体的标识符。

⑦K 是数据库,表示当发现问题时所使用的数据库标识符。

⑧L 是文档号,表示有错误的文档的编号。

⑨M 表示出现错误的主要测试实例的标识符。

⑩N 是硬件,表示发现问题时所使用的计算机系统的标识。

⑪O 是问题描述/影响,填写问题特征的详细描述。如果可能则写明实际存在的问题,还要给出该问题对将来测试、界面软件和文档等的影响。

2.软件修改报告单(SCR)

对软件产品或其阶段产品的任何修改,都必须经过评审、批准后才能重新投入运行称为阶段产品释放。这一过程用软件修改报告单(Software Change Report)给予记录。软件修改报告单的格式如表 2-5 所示。当收到了软件问题报告单之后,配置管理人员便要填写软件修改报告单。软件修改报告单要指出修改类型、修改策略和配置管理状态,它是供配置控制小组进行审批的修改申请报告。

表中各项内容的含义如下:

(1)A 是登记号,它是配置修改小组收到软件修改报告单时所作的编号。

(2)B 是配置管理人员登记软件修改报告单的登记日期。

(3)C 是已经准备好软件修改报告单,可以对它进行评审的时间。

(4)D、E 和 F 的意义是软件修改报告单的编号。如该编号中提出的问题只是部分解决,则在填写时要在该编号后附以字母 P(PART 表示部分之意)。

(5)H 指出是程序修改、文档更新、数据库修改还是它们的组合,如果仅是指出用户文档的缺陷则在解释处做上记号。

(6)I 是修改的详细描述。如果是文档更新,则要列出文档更新通知单的编号;如果是数据库修改,则要列出数据库修改申请的标识号。

表 2-5　　软件修改报告单(SCR)

软件修改报告单				登记号(A) 登记日期(B) 评审日期(C)	 年　月　日 年　月　日
项目名(D)		子项目名(E)		代号(F)	
响应哪些 SPR:(G)					
修改类型(X)		修改申请人(Y)		修改人(Z)	
修改:(H)					

修改描述:(I)

批准人:(J)

改动:					
语句类型:(K)	I/O 计算　逻辑　数据处理				
程序名:(L)		老版本号:(M)		新版本号:(N)	
数据库:(O)	DBCR:(P)	文档:(Q)		DUT:(R)	
修改已测试否:(S)	单元	子系统工程	组装	确认	运行
成功否:(S)					
SPR 的问题叙述准确否?(T)	是　否				
附注:(U)					
问题来自:(V)	系统设计规约书　需求规约书　设计说明书　数据库　程序				
资源来自:(W)	人工数:(单位:人日)计算机时间:(单位:小时)				

(7)J 是批准人,经批准人签字、批准后才能进行修改。

(8)K 是语句类型。程序修改中涉及到的语句类型包括:输入/输出语句类、计算语句类、逻辑控制语句类和数据处理语句类(如数据传送、存放语句等)。

(9)L 是程序名,即被修改的程序、文档或数据库的名字。如果只要求软件修改报告单做解释性工作,则是重复软件问题报告单中给出的名字。

(10)M 指当前的版本/修订本标识。

(11)N 指修改后的新版本/修订本标识。

(12)O 指数据库,如果申请数据库修改,这里给出数据库的标识符。

(13)P 是数据库修改申请号 DBCR。

(14)Q 指文档,即如果要求文档修改,这里给出文档的名字。

(15)R 是文档更新通知单编号 DUT。

(16)S 表示修改是否已经测试,指出已对修改做了哪些测试,如单元、子系统、组装、确认和运行测试等,并注明测试成功与否。

(17)T 指出在软件问题报告单中给出问题描述是否准确,并回答是或否。

(18)U 是问题注释,准确地重新叙述要修改的问题。

(19)V 指明问题来自哪里,如系统设计规约书、软件需求规约书、概要设计说明书、详细设计说明书、数据库和源程序等。

(20)W 说明完成修改所需要的资源估计,即所需要的人月数和计算机终端时数。

(21)X 指出所要进行修改的类型,由执行修改的人最后填写。修改类型主要有适应性修改、改进性修改以及计算错误、逻辑错误、输入和输出错误、接口错误、数据库错误、文档错误以及配置错误等的修改。

(22)Y 是提出对软件问题进行修改的人员或单位。

(23)Z 是完成软件问题修改的人员或单位。

2.6　项目组织管理

小型软件项目成功的关键是高素质软件的开发人员。然而大多数软件产品的规模都很大,以致单个的软件开发人员无法在合理的时间内完成软件产品的生产,因此必须把许多软件开发人员组织起来,使他们分工协作共同完成软件开发的工作。因而大型软件项目成功的关键除了高素质的开发人员以外,还必须有高水平的管理。没有高水平的管理,软件开发人员的素质再高,也无法保证软件项目的成功。

为了成功地完成大型软件开发的工作,项目的组成人员必须以一种有意义、有效的方式彼此交互与通信。如何安排项目组成人员是一个管理问题,管理者必须合理地组织项目组,使项目组具有尽可能高的生产率,能够按照预定的进度计划完成所承担的工作。经验表明:影响项目进展和质量的最重要因素是组织管理水平,项目组组织的越好,生产效率就越高,产品质量也越高。本节介绍几种常见的项目组组织形式,管理人员应该了解这些常用的组织形式,根据项目的具体情况决定具体的项目组组织形式。此外不要拘泥于这几种组织形式,在实践中还要不断地探索新的组织形式,完善已有的组织形式,这也是 CMM 最高级对一个组织的要求。

软件工程项目的管理涉及需求分析人员组织管理、规格说明人员组织管理、计划与设计人员的组织管理、编码人员的组织管理等。开发规模较小的软件时,可能由一个人负责需求分析、规格说明、计划和设计工作,而编码则有 2～3 个程序员来完成。这里主要涉及的是程序员的管理。开发大型软件过程的每一阶段都需要大量的开发人员协同工作,而编码阶段是由多个开发人员分担,其中每个程序员独立完成自己负责的模块。因为程序员主要用在编码阶段,所以程序员的组织问题在编码阶段最突出。无论是大型软件项目,还是小型软件项目,程序员的组织管理问题都是组织管理的重点,因此本教材主要研究程序员的管理问题。有两种极端方式可用来组织程序员,一种是民主制程序员组织,另一种是主程序员的组织。

2.6.1 民主制程序员组

民主制程序员组的指导思想是民主决策、民主监督，它要求改变评价程序员价值的标准，使得每个程序员都鼓励该组织中的其他成员找出自己编写的代码中的错误。每个程序员都不认为发现存在的错误是坏事，把找出模块中的一个错误看做是取得了一个胜利。任何人都不应该嘲笑程序员所犯的错误。程序员组作为一个整体，将培养一种平等的团队精神，要树立这样的概念：每个模块都是属于整个程序员组的，而不是属于某个人的。一组无私的程序员构成了一个程序员组。民主制程序员组的结构图如图 2-6 所示（假设该程序员组由 4 个程序员组成）。

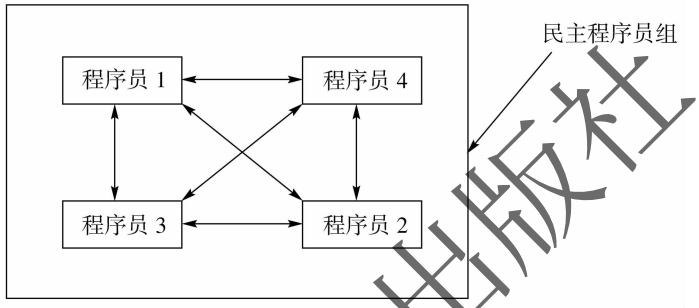

图 2-6 民主制程序员组

民主制程序员组的一个不足之处是：小组成员完全平等，享有充分的民主，通过协商作出技术决策，责任不明确，可能出现表面上人人负责，实际上人人都不负责的局面。再者，小组成员之间的通信是平行的，如果一个小组有 n 个成员，则要占用 n(n−1)/2 个信道。由于这个原因，程序组的人数不能太多，否则将会由于过多的通信而导致效率大大降低。此外，通常不能把一个软件系统分成大量独立的单元，如果程序设计小组人员太多，则每个组员所负责开发的程序单元与系统其他部分的界面将非常复杂，接口出现错误的可能性增加，而且软件测试既困难又费时。

一般说来程序设计小组的规模应该比较小，以 2～8 名成员为宜。如果软件规模很大，用一个小组无法在预定的时间内完成开发任务，则应该采用模块化层层分解的方法，使用多个程序开发小组，每个小组承担工程项目的一部分任务，在一定程度上独立自主地完成各个小组的任务。系统的总体设计应该能够保证各个小组负责开发的各部分之间的接口是经过良好定义的，并且要尽可能简单。

民主制程序员组，通常采用非正式的组织形式，也就是说，虽然名义上有一个组长，但是他和组内其他成员是完全平等的，他们完成同样的任务。这样的小组中，由全体成员讨论决定应该完成的工作，并且根据个人的能力和经验分配适当的任务。

民主制程序员组的优点是：对发现的错误抱着积极的态度，这种积极态度有助于更快速地发现错误，从而生产出高质量的代码；小组成员享有充分的民主，小组具有高度的凝聚力，组内学术气氛浓厚，有利于攻克技术难关。因此，当有难题需要解决时，即当所要开发的软件产品的技术难度较高时，采用民主制程序员组是适宜的。

如果组内大多数成员都是经验丰富技术熟练的程序员，那么非正式的组织形式可能非常成功。在这样的小组内组员享有充分的民主，通过协商，在自愿的基础上作出决定，因此

能够增强团结，提高工作效率。但是，如果组内成员多数据技术水平不高，或是缺乏经验的新手，那么这种非正式的组织方式也可能产生严重的后果：由于没有明确的权威指导开发工程的进行，组员间将缺乏必要的协调，最终可能导致工程失败。

为了使少数经验丰富、技术高超的程序员在软件开发过程中能够发挥更大的作用，程序设计组可以采用下面介绍的主程序员组织形式。

2.6.2　主程序员组

这种组织形式于 20 世纪 70 年代在美国出现。那时 IBM 公司首先开始采用主程序员组的组织形式。当时采用这种组织形式主要出于以下几方面的考虑：

(1)软件开发人员多数缺乏经验。

(2)程序设计过程中有许多事务性的工作，例如有大量的信息存储和更新。

(3)多信道通信量费时间，降低程序员工作的效率。

(4)这种主程序员组织形式有两个关键特性：

①专业化：该组每名成员仅完成那些他们受过专业训练的工作。

②层次性：主程序员指挥组内的每个程序员，并对软件全面负责。

由于以上问题，为了责任分明地做好软件开发工作，发挥少数经验丰富、技术高超的程序员在软件开发过程中的关键作用，通过对其他软件开发人员的专业化训练与专业化分工从而高效率地开发出高质量的软件，所以采用了主程序员组的组织形式。

一个典型的主程序员组如图 2-7 所示。一个小组由主程序员、后备程序员、编程秘书以及 1～3 名程序员组成。在必要的时候，小组还可以有其他领域的专家协助。

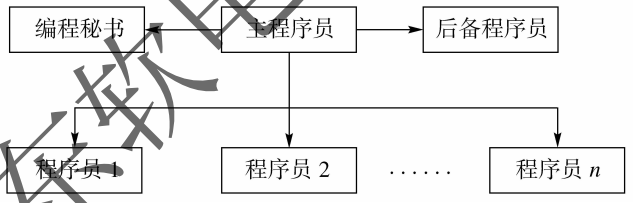

图 2-7　主程序员组的组织形式

主程序员组核心人员的分工如下：

(1)主程序员既是成功的管理人员又是经验丰富、能力强的高级程序员，负责总的软件体系结构设计和关键部分的详细设计，并且负责指导其他程序员完成详细设计和编码工作。程序员之间没有通信渠道，所有接口问题都由主程序员处理。因为主程序员为每行代码的质量负责，所以他还要对其他成员的工作成果进行复查。

(2)后备程序员也应该技术熟练而且富于经验，他协助主程序员工作并且在必要的时候接替主程序员的工作。因此后备程序员必须在各个方面都和主程序员一样优秀，并且对本项目的了解也应该和主程序员一样多。平时，后备程序员的主要工作是设计测试方案、测试用例、分析测试结果及其他独立于设计过程的工作。

(3)编程秘书也就是主程序员的秘书或助手，他必须负责完成与项目有关的全部事务性工作，例如维护项目资料和项目文档，编译、链接、执行程序和测试用例。

这是当初的主程序员组的思想，现在的情况已经大为不同。现在各个程序员都已经有

了自己的终端或工作站，他们在自己的终端或工作站上完成代码的输入、编辑、编译、链接和测试等工作，无需由编程秘书统一做这些工作，编程秘书很快就退出了软件工程领域。

1972年完成的纽约时报信息库管理系统的项目中，由于使用结构程序设计技术和主程序员组的形式，从而获得了巨大的成功。83 000行程序只用11人/年就全部完成；验收测试中只发现21个错误；系统在第一年运行中只暴露出25个错误，而且仅有一个错误造成系统失效。

主程序员组的组织形式是一种比较理想化的组织形式，但是在实际中很难组成这种典型的主程序员组的软件开发队伍，典型的主程序员组在许多方面是不切实际的。

首先，如前所述，主程序员应该是高级程序员和成功的管理者的结合体，承担这项工作需要同时具备这两方面的才能。但是，在现实社会中很难找到这样的人才，通常，既缺乏成功的管理者，也缺乏技术熟练的程序员。

其次，后备程序员更难找到。人们总是期望后备程序员像主程序员一样出色，但是他们必须坐替补席上，拿着较低的工资等待随时接替主程序员的工作。任何一个优秀的高级程序员或高级管理人员都不愿意接受这样的工作。

实际工作中需要一种更合理、更现实的组织程序员组的方法，这种方法应该能充分结合民主制程序员组和主程序员组的优点，并能用于实现更大规模的软件产品。

2.6.3 现代程序员组

民主制程序员组的最大优点是小组成员都对发现程序错误持积极、主动的态度。由于它固有的一些缺点，使得它不适合大型软件项目中的程序员组织，所以产生了主程序员组的组织形式。但是，使用主程序员组的组织方式时，主程序员对每行代码的质量负责，因此必须参与所有代码的审查工作。由于主程序员同时又是负责对小组成员进行评价的管理员，他参与代码审查工作就会把所发现的错误与小组成员的工作业绩联系起来，从而造成小组成员不愿意发现错误的心理。

摆脱上述矛盾的方法是取消主程序员的大部分行政管理工作。前面已经介绍过很难找到一个既是高度熟练的程序员又是成功的管理员的人，取消主程序员的行政管理工作，不仅摆脱了上述矛盾也使得寻找主程序员的人选不再那么困难。于是，实际的主程序员由两个人来担任：一个技术负责人，负责小组技术活动；一个行政负责人，负责所有非技术的管理决策。这样的组织结构如图2-8所示。

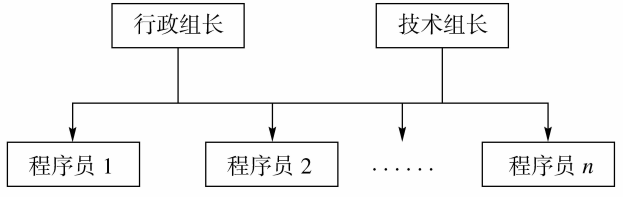

图 2-8 现代程序员组

图2-8所示的组织结构并没有违反雇员不应该向多个管理者汇报工作的基本管理原则。负责人的负责范围定义得很清楚：技术组长只对技术工作负责，因此他不处理诸如预算和法律之类的问题，也不对组员业绩进行评价；另一方面，行政组长全权负责非技术事务，因

此，他无权对产品的交付日期做出许诺，这类承诺只能由技术组长来做。

技术组长自然要参与全部代码的审查工作，因为他对全部代码的质量负责。相反，不允许行政组长参加代码审查工作，因为他的职责是对程序员的业绩进行评价，行政组长的责任是在常规调度会议上了解小组中每个程序员的技术能力。

在项目开始前，明确划分技术组长和行政组长的管理权限是很重要的。但是，有时也会出现职责不清的矛盾，例如，考虑年度休假问题，行政组长有权批准某个程序员年度休假的申请，因为这是一个非技术问题；但是技术组长可能马上会否决这个申请，因为已经接近预定的产品完工期限，人手非常紧张。解决这类问题的办法是求助于更高层次的管理人员，对于行政组长和技术组长都认为是属于自己职责范围的事务，制订一个处理方案。

由于程序员组的组成人员不宜过多，当软件项目规模较大时，应该把程序员分成若干个小组，采用如图 2-9 所示的组织结构。该图描绘的是技术管理的组织结构，非技术管理的组织结构与此类似。由图可以看出，产品的实现作为一个整体是在项目经理的指导下进行的，程序员向他们的组长汇报工作，而组长向项目经理汇报工作。当产品规模更大时，可以增加是间管理层次。

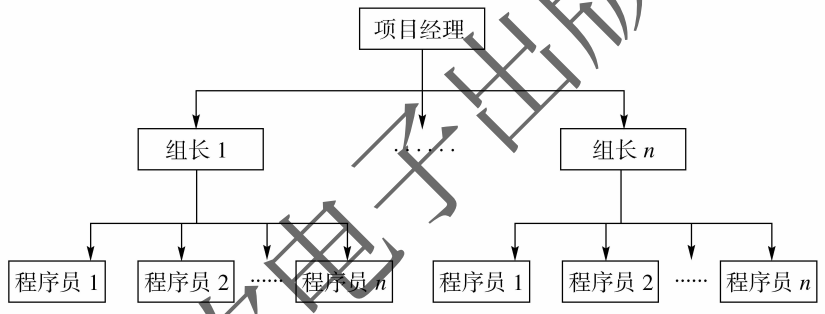

图 2-9　大型项目的技术管理组织结构

把民主程序员组和主程序员组的优点结合起来的另一种方法是在合适的地方采用分散决定的方法，如图 2-10 所示。这样做有利于形成畅通的通信渠道，以便充分发挥每个程序员的积极性和主动性，集思广益攻克技术难关。

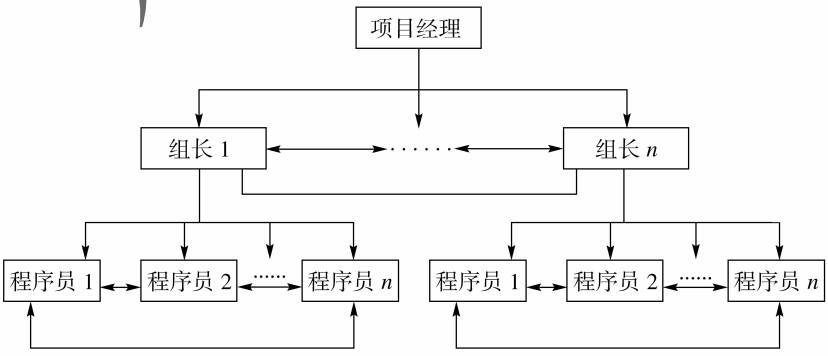

图 2-10　包含分散决策的组织方式

这种组织方式对于适合采用民主方法的那类问题非常有效。尽管这种组织方式适当地发扬了民主，但是上下级之间的箭头仍然是向下的，也就是说，是在集中指导下发扬民主。显然，如果程序员可以指挥项目经理，则只会引起混乱。

2.6.4 软件项目组

如前所述，程序员组的组织方式主要用于实现阶段，当然也适用于软件生命周期的其他阶段。

1.三种组织方式

Mantei 提出了三种通用的项目组织方式：

(1)民主分权式。这种软件工程小组没有固定的负责人，任务协调人是临时指定的，随后将由新的协调人取代。用全体组员协商一致的方法对问题及解决问题的方法作出决策。小组成员间的通信是平行的。

(2)控制分权式。这种软件工程小组有一个固定的负责人，协调特定任务的完成并指导负责子任务的下级领导人的工作。解决问题仍然是一项群体活动，但是，通过小组负责人在子组之间划分任务来实现解决方案。子组和个人之间的通信是平行的，但是也有沿着控制层的上下级之间的通信。

(3)控制集权式。小组负责人管理顶层问题的解决过程并负责组内协调。负责人和小组成员之间的通信是上下级式的。

选择软件工程小组的结构时，应该考虑下述 7 个项目因素：

(1)待解决问题的困难程序。

(2)要开发的程序的规模。

(3)小组成员在一起工作的时间。

(4)问题能够被模块化的程序。

(5)待开发系统的质量和可靠性的要求。

(6)交付日期的严格程度。

(7)项目的社交(通信)程度。

集权式结构能够更快地完成任务，它最适合处理简单问题。与分权式的小组比起来，能够产生更多、更好的解决方案，这种小组在解决复杂问题时成功的可能性更大。因此，控制分权式或者控制集权式小组结构能够成功地解决简单的问题，而民主分权式结构则适合于解决难度较大的问题。

小组的性能与必须进行的通信量成反比，所以开发规模很大的项目最好采用控制分权式或者控制集权式小组结构的小组。

小组生命周期长短影响小组的士气。经验表明，民主分权式结构能够导致较高的士气和较高的工作满意度，因此适合于生命周期长的小组。

民主分权式结构最适合于解决模块化程度较低的问题，因为解决这类问题需要更大的通信量。如果能达到较高的模块化程度，则控制分权式或者控制集权式小组结构更为适宜。

人们曾经发现，控制分权式或者控制集权小组结构产生的缺陷比民主分权式结构小组少，但这些数据在很大程度上取决于小组采用的质量保证的问题。

完成同一个项目，分权式结构通常需要比集权式结构更多的时间；不过当需要高通信量时，分权式结构是最适宜的。

历史上最早的软件项目组是控制集权式结构，当时人们把这样的软件项目组称为主程序员组。如表 2-6 所示概括了项目特性对项目组织方式的影响。

表 2-6　　项目特性对项目结构的影响

项目特性 \ 小组类型		民主分权式	控制分权式	控制集权式
困难程度	高	√		
	低		√	√
规模	大		√	√
	小	√		
小组生命周期	长		√	√
	短	√		
模块化程度	高		√	√
	低	√		
可靠性	高	√	√	
	低			√
交付日期	紧			√
	松	√	√	
通信	高	√		
	低		√	√

2.软件工程小组的组织范型

软件工程小组有 4 种组织范型：

(1)封闭式范型。按照传统的权力层次来组织项目组。当开发与过去已经做过的产品相似的软件时，这种项目组可以工作得很好；但是，在这种封闭范型下难以进行创新性的工作。

(2)随机式范型。松散地组织项目组，小组工作依靠小组成员发挥个人的主动性。当需要创新性或技术上的突破时，用随机式范型组织起来的项目组能够工作得很好；但是，当需要“有次序地执行”才能完成任务时，这样的项目组就可能陷入困境。

(3)开放式范型。这种范型试图以一种既具有封闭式范型的控制性，又包含随机式范型的创新性的方式来组织项目组。通过大量协商和基于一致意见做出的决策，项目组成员相互协作完成任务。用开放式范型组织起来的项目组适合解决复杂问题，但是可能没有其他类型小组的效率高。

(4)同步式范型。按照对问题的自然划分，组织项目组成员各自解决一些子问题，他们之间很少有主动的通信要求。

2.6.5　IT 组织管理

前面从微观方面介绍了软件开发程序员组的组织形式，下面从宏观方面讨论 IT 组织(或企业)如何组织才有利于软件项目的实施。

无论是项目型公司还是产品型公司，从事软件开发的组织或公司应该有一定的软件开发组织结构。一个合理的软件开发组织结构是确保软件开发质量的最基本保证，各个组织各司其职，可以确保软件开发按拟订的质量控制规则与软件开发计划进行，有利于软件公司软件质量与成本的控制。

1.软件开发组织机构设置

一般而言，对于产品型软件公司，其公司内部均会有一个类似于产品管理小组这样的组织和一个专门负责产品发展的产品经理部门；而项目型公司则相对简单一些，主要是针对项目进行定制开发，一般对项目的发展方向不做控制。但从项目开发演变为可推广产品的另当别论。一般来讲，如图 2-11 所示是一个典型的软件公司软件开发的组织机构设置。

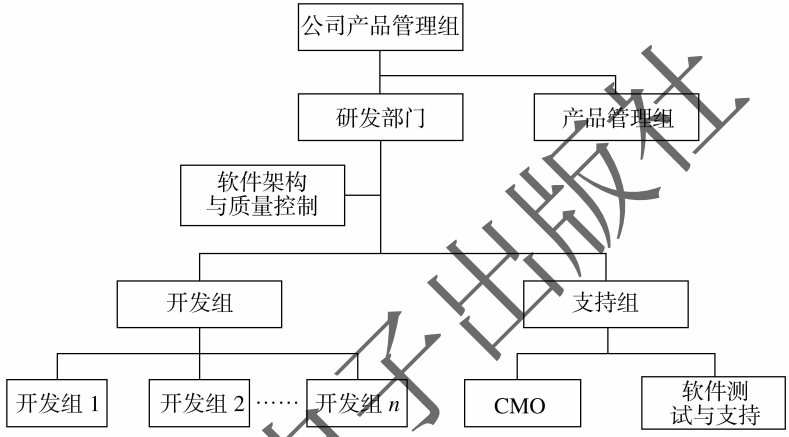

图 2-11 软件公司的组织机构设置

2.组织机构的职责分工

在上述的组织机构中，各职能组织有各自明确的责权范围，完成各自的本职工作。各组织相互协调完成相应的软件开发与维护工作。

(1)公司产品管理组。对于产品型软件公司而言，软件产品是其生存与发展的基础。公司对新产品立项、现有产品的发展方向及有关产品发展的重大决定均需由公司产品管理组来决定。公司产品管理组一般由公司的执行总裁、技术总监、市场总监、产品经理、研发经理及其他必要人员组成。

(2)产品管理部门。产品管理部门是介于研发部与市场部之间的一个桥梁部门。产品管理部门的主要职责是负责产品发展策略的制订与执行。这里的执行包括软件开发前期的市场及需求调研，完成可行性分析报告，制订产品规格。它参与软件开发项目组，并完成相关工作。

(3)研发部门。研发部是软件开发的主体，主要任务是完成软件或项目的开发工作。其工作内容通过各职能组实现，主要包括：功能规范、开发活动、支持工作、项目计划、定义项目里程碑和软件定版等。

(4)软件架构与质量控制。是软件开发的质量控制机构，主要职责是负责软件开发过程的质量控制，及时发现问题、解决问题，确保进入下一阶段的设计符合设计规范要求，实现软件开发全程监控。软件架构与质量控制为非常设机构，主要由研发经理、产品经理、资深系统分析员、测试经理等人员组成。根据项目进展需要，由研发经理召集进行项目阶段评审。

(5)软件开发组。主要由各种角色的开发人员构成,完成开发任务。

(6)CMO(Configuration Management Officer,软件配置管理员)。对于一个具有一定规模的软件公司都会有一个软件配置管理机构,对于小型公司一般由项目经理代管。CMO的主要职责是进行软件开发过程中的软件配置管理,以及软件定版后的维护管理。在软件开发过程中,由于多个开发人员协同工作,需要对其工作协同管理,确保协同工作的顺利进行。同时,由专人进行配置管理,使得大部分开发人员不会得到全部源代码,也有利于软件公司的安全保密工作。在软件定版后,由于软件的 Bugs、功能的完善及各种原因导致的对软件的修改,版本控制就显得极为重要,软件配置管理可以确保得到不同时间的软件版本。

(7)软件测试组。软件测试是软件工程的重要组成部分。软件测试组承担的工作主要是确认测试和系统测试。模块测试与集成测试由软件开发人员完成。对于项目软件开发,软件测试组需要根据需求规格说明书中规定的内容,对可运行的软件进行功能及性能等方面的测试,确保交付的应用系统是用户可信赖的系统。

在以上的软件开发组织机构中,不论公司规模的大小,以上的各个职能应该是健全的。明确的责任分工有利于软件开发的顺利进行与质量控制;同时,也必将有利于公司的成本控制,降低软件开发风险。

3.软件开发项目组的角色

一般来讲,一个软件开发项目组是由多个不同角色的人员构成,每种角色在软件开发中起不同的作用,各个不同角色的人员协同工作,共同完成软件开发工作。

典型的软件开发项目组由下列角色构成,如图 2-12 所示。

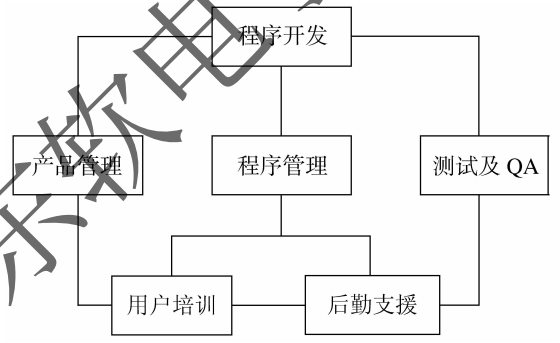

图 2-12　软件开发项目组的主要角色

在软件开发项目组中一般有 6 种角色,分别是:产品管理、程序开发、程序管理、测试及质量保证、用户培训和后勤支援。

在大型软件开发项目中,可以将每个角色赋予不同的个人。对小型项目,一个人可以肩负多个不同的角色。每种角色的人员在项目中起着同等重要的作用。每种角色都有其特定的任务及技能要求。

(1)产品管理。产品管理负责建立及更新项目的商业模型,在确定及设置项目目标方面起关键作用。产品经理应确保项目成员清楚理解项目的商业目的,并根据商业需求的优先级确定功能规范。

(2)程序管理。程序管理负责确定软件特色及功能规范,根据软件开发标准协调日常开发工作,确保及时交付开发任务。协助产品经理完成项目需求文档,并根据需求文档起草软

件功能规范;同时负责与系统分析、规范及框架结构有关的各种活动。管理与协调同外部标准与系统的互操作性,控制项目进度。程序经理是项目组成员间沟通与协调的核心。

(3)程序开发。开发队伍负责交付符合功能规范的软件系统。开发队伍应积极参与功能规范的制订,在建立项目原型时开发人员与程序经理可以同步进行并提供技术可行性。在功能规范确立后,开发人员必须与程序经理就如何解决重大疑难问题达到一致。

(4)测试与质量保证。测试与质量保证是保证系统符合功能规范的保证。为保证“零误码”,测试/QA 人员应积极参与开发过程,确保开发、交付符合功能规范的软件系统。测试/QA 人员负责准备测试计划、测试用例、自动测试程序、执行测试工作、管理并跟踪 Bug。测试工作与开发工作是独立并行的。

(5)用户培训。培训人员负责设计编写离线及在线培训文档,包括演示材料。培训人员应参与用户界面和系统的设计与构造,并参与安装程序与安装过程的设计,参与系统的可用性测试及设计改进,并确保系统的变化及时反映到文档中去。

(6)后勤支援。后勤支援包括确保项目顺利进行的各方面工作。

对于一个项目组,建立一个良好的团队氛围是非常重要的,每个角色在项目中都是不可缺少的,项目的成功是团队成员共同努力的结果。鼓励成员的积极进取、高效参与的团队精神,提高成员的责任感,避免造成团队或项目的成功依赖于少数个人的贡献。

2.7 项目质量管理

根据国际标准组织(ISO)的定义,质量是依靠特定的或暗指的能力满足特定需要的产品或服务的全部功能和特征。这个定义说明了质量是产品的内在特征,描绘了产品的质量观点。质量不是单独以产品为中心的,而是与客户和产品都有联系的。其中客户是出资者或受影响的部分人,而产品包括利益和服务。进一步讲,质量的概念会随着时间和环境价值的改变而改变,价值会使人们弄清什么是好的、什么是不好的。因此,软件的质量作为产品或服务需要的功能/特征,也必须定义于客户和组织间的内容。

2.7.1 软件质量概述

质量管理是企业管理的核心内容之一。由于软件产业本身的特殊性,即主要靠脑力劳动,而非靠设备和材料等的传统工业化生产,因此,软件企业开展质量管理工作就变得十分困难 。软件产品的开发涉及到方方面面的人员,历经多个生产环节,产生大量的中间软件产品,各个环节都可能带来产品质量问题;同时,由于软件产品是逻辑体,不具备实体的可见性,因而难以度量,质量也难以把握。因此如何有效地管理软件产品的质量一直是软件企业面临的挑战。归纳起来,软件质量管理大体分为 3 种:事后检验、全面质量管理和权威认证。

1.事后检验

事后检验的方式是在产品生产的最后环节进行质量检查,合格的产品准许出厂,不合格的产品作为次品处理。这种质量管理方式对于制造批量大、制造成本较低的产品是一种较好的质量管理方式,但这是传统产业生产的最初的质量管理方式,是低级的质量管理方式。

虽然，在传统产业这仍然是质量控制的最后一个环节，但已不是质量管理的主流方法，也不适应软件产品的质量管理要求，因为这种产品的生产没有批量可言。

2.全面质量管理

ISO9000 质量管理体系就是全面质量管理体系的一个范例。它要求从影响软件产品质量的各个方面加强对软件质量的全面管理。ISO9000 中列出的影响软件质量的因素包括了管理职责、质量体系、合同评审等 20 个方面。实现对这些影响因素的全面管理是全面质量管理，当然多数组织不能做到完全符合 ISO9000 的规定，只要做到其中的绝大多数方面，就可以认为实现了全面质量管理。

3.权威认证

认证的概念来自于这样的事实：如果一个组织具有合格的技术人员，且这个组织的管理水平很高，比如完全实现了全面质量管理，那么这样的组织具备了一定的生产合格产品的能力，应该能够生产出合格的软件产品。所以要考察一个组织的产品质量，可以首先看该组织通过了哪一个级别的质量体系认证（是 ISO9000 还是 CMM2 还是 CMM5）。认证已经成为一个组织资质的证明，也成为买方选择合格供应方的首要考虑。

软件质量管理的目的是建立对项目的软件产品质量的定量理解，和实现特定的质量目标。软件质量管理着重于确定软件产品的质量目标，制订达到这些目标的计划，并监控及调整软件计划、软件工作产品、活动及质量目标以满足顾客及最终用户对高质量产品的需要及期望。软件质量管理的实践基于集成软件管理、软件产品工程、定量过程管理的实践，集成软件管理、软件产品工程建立和实施项目的明确定义的软件过程区域，定量过程管理建立了对项目明确定义的软件过程达到期望目标的结果能力的定量理解。有以下要点：

(1)对项目的软件质量活动做出计划。

(2)对软件产品质量的可测量的目标及其优先级进行定义。

(3)确定实现软件产品质量目标的实现过程是可量化的和可管理的。

(4)为管理软件产品的质量提供适当的资源和资金。

(5)对实施和支持软件质量管理的人员进行所要求的培训。

(6)对软件开发项目组和其他与软件项目有关的人员进行软件质量管理方面的培训。

(7)按照已文档化的规程制订和维护软件项目的质量计划。

(8)项目的软件质量管理活动要以项目的软件质量计划为基础。

(9)在整个软件生命周期，要确定、监控和更新软件产品的质量目标。

(10)在事件驱动的基础上，对软件产品的质量进行测量、分析，并将分析结果与产品的定量目标相比较。

(11)对软件产品的定量质量目标进行合理分工，分派给向项目交付软件产品的承包商。

(12)对软件产品进行测试，并将测试结果用于软件质量管理活动的状态。

(13)高级管理者定期参与评审软件质量管理的活动。

(14)软件项目负责人定期参与评审软件质量管理的活动。

(15)软件质量保证评审小组负责评审软件的质量管理活动和工作产品，并填写相关报告。

软件质量过程要注意 4 点：

(1)从一开始就要保证不出错，至少应该努力使错误尽量不在编写代码时发生。为了做

到这一点包括采用适当的软件工程标准和过程，建立独立的质量保证标准和过程；根据过去的经验和教训制订正确的方法；像软件工具和合同软件一样的高质量输入。

(2)确保尽早发现错误并纠正。错误隐藏得越久，修正错误花的代价就越大。因此，质量控制必须在开发生命周期中的每一个阶段都要重视，如需求分析、设计、文档和代码。这些都属于所有的回顾方法，如检查、预先排除与技术回顾等。

(3)消除引起错误的引导因素，还没有找到错误的诱因就纠正错误是不恰当的。通过排除错误的诱因你就达到了改良过程的目的。

(4)质量管理的基本原则实际上就是贯彻全面质量管理的原则。具体地说就是坚持下面的质量管理原则：

控制所有过程的质量；过程控制的出发点是预防不合格；质量管理的中心任务是建立并实施文件化的质量体系；持续的质量改进；有效的质量体系应满足顾客和组织内部双方的需要和利益；定期评价质量体系；搞好质量管理关键在于领导。

2.7.2 软件质量因素

一个软件的质量如何，可以用一套质量指标来衡量。这些指标包括：

(1)正确性。系统满足规格说明和用户目标的程度，即在预定环境下能正确地完成预期功能的程序。它要求软件没有错误，能够满足用户的目标。

(2)健壮性。在硬件发生故障、输入的数据无效或操作错误等意外环境下，系统能做出适当响应的程度。

(3)效率。为了完成预定的功能，系统需要的计算资源的多少。这可以用系统需要占用的计算机硬件资源，或者所需要的时间来表示。

(4)完整性(安全性)。对未经授权的人使用软件或数据的企图，系统能够进行控制(禁止)的程度，以及为某些目的能够保护数据，使系统免受偶然的/有益的破坏、改动或遗失的能力。

(5)可用性。系统在完成预定应该完成的功能时令人满意的程度，即用户学习、使用软件及程序准备输入和解释输出所需要的工作量的多少。

(6)风险。按预定的成本和进度把系统开发出来，并且为用户所满意的概率。

(7)可理解性。理解和使用该系统的容易程度。

(8)可维修性。为了满足用户的新需求或者由于环境发生了变化，或者发生了新的错误，诊断和改正在运行现场发现的错误所需要的工作量的多少。

(9)灵活性(适应性)。修改或改进正在运行的系统需要的工作量的多少。

(10)可测试性。软件容易测试的程度，及测试软件以确保其能够执行预定功能所需要的工作量的多少。

(11)可移植性。把程序从一种硬件配置和(或)软件系统环境转移到另一种配置和环境时，需要的工作量的多少。有一种定量度量的方法是：使用原来程序设计和调试的成本除移植时需用的费用。

(12)可再用性。在其他应用中该程序可以被再次使用的程度(或范围)。

(13)互运行性。把该系统和另一个系统结合起来所需要的工作量的多少。

2.8　项目风险管理

软件风险是软件开发过程某个时间点以后的关于软件的不确定性因素对于软件开发过程的影响。风险会造成的损失可能是经济上的，也可能是时间上的，或者是无形的其他损失等。如果项目风险变成现实，就有可能会影响项目的进度，增加项目的成本，甚至使软件项目不能实现。如果软件开发项目不关心风险管理，结果就会遭受极大的损失。如果对项目进行良好的项目风险管理，就可以降低软件项目的风险，大幅度增加项目实现目标的可能性。因此任何一个软件开发项目都应将风险管理作为软件项目管理的重要内容。软件风险管理的目的在于标识、定位和消除各种风险因素，在其来临之前阻止或最大限度减少风险的发生，从而避免不必要的损失，以使项目成功操作或使软件重写的几率降低。

在进行软件项目风险管理时，要标识出潜在的风险，评估它们出现的概率及产生的影响，并按重要性加以排序，然后建立一个规划来管理风险。风险管理的主要目标是预防风险，所以必须建立一个意外事件计划，使其在必要时能以可控的和有效的方式做出反应。风险管理目标的实现包含 3 个要素：首先，必须在项目计划书中写下如何进行风险管理；第二，项目预算必须包含解决风险所需的经费，如果没有经费，就无法达到风险管理的目标；第三，评估风险时，风险的影响也必须纳入项目规划中。

2.8.1　风险的分类

根据风险内容，可以将风险分为项目风险与外来风险。项目风险是项目自身具有的风险，包括需求风险（表现为需求的变化或者原来需求分析不准而带来的风险）、项目技术风险（表现为采用了不成熟的技术而使软件开发不能顺利进行下去）、管理风险（公司管理人员是否成熟等）、预算风险（预算是否准确等）。外来风险包括外来技术风险（由于更新的技术的出现，而使得采用的技术过时等）、商业风险（开发出的产品由于不被市场接受而无法销售出去等）、战略风险（公司的经营战略发生了变化）等。

在这些风险中，有些风险是可以预见到的，如员工离职；而有些不是不可预见的。可以预见的风险不会造成根本性的损失，不可预见的风险有时会造成系统彻底失败。

2.8.2　风险的识别

风险识别是系统化地识别可预测的项目风险，在可能时避免这些风险，且当必要时控制这些风险。风险识别的有效方法是建立风险项目检查表，对所有可能的风险因素进行检查。主要涉及以下几方面的检查。

（1）产品规模风险。与软件的总体规模相关的风险，即对于软件的总体规模预测是否准确。如果预测规模小于实际软件规模，肯定会导致费用上升，开发时间增加。

（2）需求风险。是否与用户进行了充分的交流，是否了解用户使用软件所处理的问题域，是否充分理解用户的需求，书面形式的需求分析是否得到用户的认可。

（3）过程定义风险。与软件过程定义相关的风险。

(4)开发环境风险。与开发工具的可用性及质量相关的风险。

(5)技术风险。采用的技术对于解决项目所涉及的问题是否是最适当的技术,技术是否成熟,是否会被淘汰等。技术风险威胁到软件开发的质量及交付的时间;如果技术风险变成现实,则开发工作可能变得很困难或根本不可能。

(6)人员数目及经验带来的风险。与参与工作的软件工程师的总体技术水平及项目经验相关的风险。在进行具体的软件项目风险识别时,可以根据实际情况对风险分类。但简单的分类并不是总行得通的,某些风险根本无法预测。比如美国空军软件项目风险管理(Software Risk Abatement)手册中的风险识别方法,要求项目管理者根据项目实际情况标识影响软件风险因素的风险驱动因子,这些因素包括以下几个方面:性能风险,即产品能够满足需求和符合使用目的的不确定程度;成本风险,即项目预算能够被维持的不确定的程度;支持风险,即软件易于纠错、适应及增加的不确定的程度;进度风险,即项目进度能够被维护且产品能按时交付的不确定的程度。

2.8.3 风险的评估

风险评估对识别出的风险进行进一步的确认分析。假设这一风险将会出现,评估它会对整个项目带来什么样的不利影响,如何将此风险的影响降低到最小,同时确定主要风险出现的个数及时间。进行风险评估时,最重要的是量化不确定性的程度和每个风险可能造成损失的程度。为了实现这点,必须考虑风险的不同类型。识别风险的一个方法是建立风险清单(表 2-7),清单上列举出在软件开发的不同阶段可能遇到的风险,最重要的是要对清单的内容随时进行维护,更新风险清单,并向所有的成员公开。应鼓励项目组的每个成员勇于发现潜在的风险并提出警告。风险清单给项目管理提供了一种简单的风险预测技术,它实际上是一个三元组$[R_iP_iL_i]$,其中R_i是第i种风险,P_i是风险R_i出现的概率,L_i是假设$P_i=1$时的损失。风险分析表如表 2-7 所示。这种损失可以用增加多少费用、增加多少开发时间或者只是某些定量的影响程度指标来表示,则P_iL_i可以刻画这种风险对于软件开发过程的潜在影响,而风险管理的目标就在于尽量减小P_i的值。

表 2-7 风险分析表

风　险	风险出现的概率 P_i	风险的影响 L_i	风险排序
风险 1	0.6	6	3.6
风险 2	0.6	5	3.0
…	…	…	…
风险 n	0.01	5	0.05

风险清单中,风险的概率值可以由项目组成员个别估算,然后加权平均,得到一个有代表性的值;也可以通过先做个别估算而后求出一个有代表性的值来完成。对风险产生的影响可以用影响评估因素进行分析。一旦完成了风险清单的内容,就要根据P_iL_i值进行排序,值大的风险放在上方,依此类推。项目管理者对排序进行研究,并划分重要和次重要的风险,对次重要的风险再进行一次评估并排序。对重要的风险要进行管理。从管理的角度来考虑,风险的影响及概率是起着不同作用的,一个具有高影响且发生概率很低的风险因素

不应该花太多的管理时间，而高影响且发生率高的风险以及低影响且高概率的风险，应该首先列入管理考虑之中。

2.8.4　风险的驾驭和监控

风险的驾驭与监控是指利用某些技术或方法，比如原型化、软件自动化、软件心理学、可靠性方法及软件项目管理的方法、保险方法等避开或者转移风险，使风险对项目所造成的影响(损失)尽可能地减小；如无法避免则应该使它降低到一个可以接受的水平。风险的驾驭，现在还没有成熟的方法或技术来指导，主要靠管理者的经验，根据不同的情况来实施。风险驾驭的原则如下。

(1)首先抓主要风险。所谓主要风险就是风险分析表中排在最前面的 P_iL_i 值最大的风险因素。通过对该风险的分析，找出避免或转移风险的办法，使该风险的 P_iL_i 值尽可能减小，并计算出该风险的新的 P_iL_i 值。将避免与转移风险的方案形成风险驾驭文档，然后对经过这样处理过的风险分析表重新进行排序。

(2)对新的风险分析表重复第(1)步的方法，又得到一个新的风险分析表。这样多次重复，直到风险分析表中的所有项的 P_{ii} 值都在可以接受的范围内，停止进行。

(3)在项目开始前与项目进行中，时刻注意可能出现的风险，按照风险驾驭文档的方法避免或转移风险。当出现新的风险时，要及时对风险分析表进行调整，并形成新的风险驾驭文档。

Boehm 归纳了 6 步风险管理法则，其中有两步关键法则，每步法则有 3 个子步骤。Boehm 建议采用适当的技术来实现每个关键步骤和子步骤。

第一步是评估，包括：

(1)风险确认。确认影响软件成功的详细的项目风险因素。

(2)风险分析。检查每个风险因素的发生概率和降低其发生概率的可能性。

(3)给确认和分析的风险因素确定级别，即风险考虑的先后顺序。

一旦项目风险因素的先后顺序排列出来了，第二步就是风险管理。这一步中，要对这些风险因素进行控制，包括：

(1)风险管理计划。制订每个风险因素如何定位，这些风险因素的管理如何与整个项目计划融为一体。

(2)在每个实现活动或工作中的风险解决方案中，消除或解决风险因素的特殊活动。

(3)风险监视。跟踪解决风险活动的风险过程的趋势。

2.9　项目沟通管理

项目沟通管理，就是为了确保项目信息的合理收集和传输，以及最终处理所需实施的一系列过程。包括为了确保项目信息及时适当的产生、收集、传播、保存和最终配置所必须的

过程。项目沟通管理为成功所必须的因素——人、想法和信息之间提供了一个关键连接。涉及项目的任何人都应准备以项目“语言”发送和接收信息，并且必须理解他们以个人身份参与的沟通会怎样影响整个项目，沟通就是信息交流。组织之间的沟通是指组织之间的信息传递。对于项目来说，要科学的组织、指挥、协调和控制项目的实施过程，就必须进行项目的信息沟通。好的信息沟通对项目的发展和人际关系的改善都有促进作用。

项目沟通管理具有复杂和系统的特征。著名组织管理学家巴纳德认为“沟通是把一个组织中的成员联系在一起，以实现共同目标的手段”。没有沟通，就没有管理。沟通不良几乎是每个企业都存在的老毛病，企业的机构越是复杂，其沟通越是困难。往往基层的许多建设性意见还未反馈至高层决策者，便已被层层扼杀，而高层决策的传达，常常也无法以原貌展现在所有人员之前。

2.10 项目集成管理

项目集成管理，是指为确保项目各项工作能够有机地协调和配合所展开的综合性和全局性的项目管理工作和过程。它包括项目集成计划的制定、项目集成计划的实施、项目变动的总体控制等。

2.11 工具实践——Project 的使用方法

2.11.1 实践背景

MSPorject 是微软公司出品的一款用来辅助项目管理的常用工具软件。人们使用 Project 进行项目的计划制定、控制和跟踪项目的进展、制定详细的进度安排等。

2.11.2 实践目的

为了让学生能够了解 Project 的使用环境，明确 Project 的初步使用方法，通过项目计划的制定过程，让学生能够使用工具实现一个项目的初步进度计划。

通过学习项目进度计划中的内容，我们可以掌握项目进度计划的基本组成部分。它包括项目的总体目标，任务的完成需要对目标任务进行分解，以便能够形成小的容易完成的任务项，这些任务之间存在一定的约束关系。任务项终归需要有人去完成，那么应该由哪些人去完成，需要花多长的时间就是后续需要制定的内容。进度计划的内容基本上已经形成，但是任务分配的是否合理，还需要从资源分配、任务分配等方面进行检验。

2.11.3　实践步骤

1.任务 1:安装 Project 工具环境

任务描述:

准备系统应用环境:包括操作系统 Windows 2000 以上版本,应用系统 Microsoft Office Project 2010。根据安装向导安装软件,并初始化工具界面。

实践步骤:

(1)安装 Microsoft Office Project 2010 工具软件

Project 2010 的安装比较简单,只要根据安装向导对安装路径进行选择之后,就可以连续点击“下一步”进行默认安装。安装成功后,在开始菜单中查找程序菜单中的“Microsoft Office”找到“Microsoft Office Project 2010”菜单项进行启动,将看到如图 2-13 所示的界面,选取“新建”→“空白项目”,创建一个新的 Project 文档。

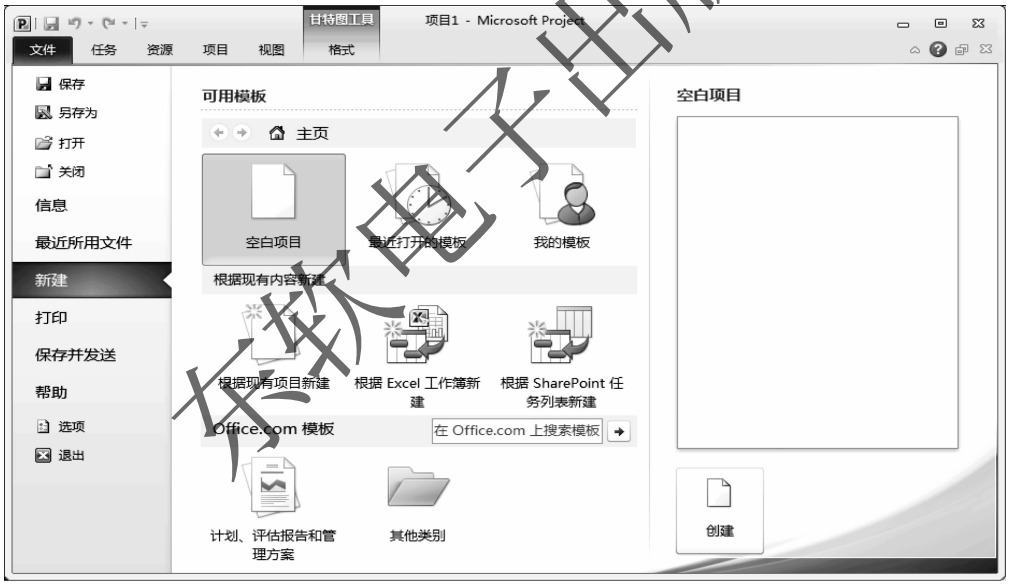

图 2-13　新建文档界面

(2)进入项目计划编辑环境

双击“空白文档”项创建项目计划文件,进入项目计划编辑环境。此时的界面如图 2-14 所示。

在这个界面中可以看到最上端的是工具栏、左边的任务分解列表和右边的甘特图区域。

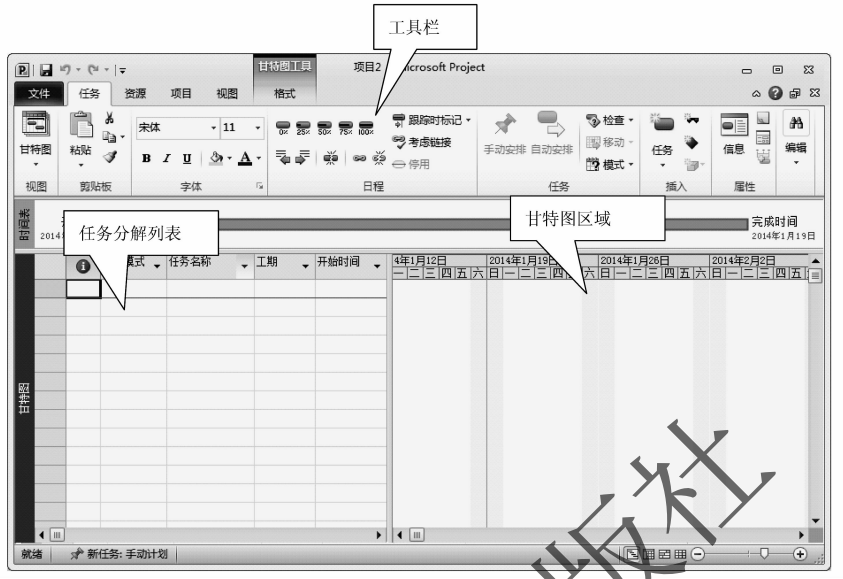

图 2-14　初始界面

2.任务 2:熟悉 Project 功能

任务描述:

通过加载一个已有项目计划文件迅速构建一个项目计划列表,然后对常用工具进行介绍,让大家对 Project 功能有个大致的了解。

实践步骤:

(1)打开已有文件形成一个项目计划列表

在配套光盘工具实验目录下打开文件"软件开发.mpp",出现如图 2-15 所示 Project 主界面。在默认情况下,当前属于甘特图的视图。

	任务名称	工期	开始时间	完成时间	前置任务	资源名称
1	**项目范围规划**	**3.5 个工作日**	**2004年1月1日**	**2004年1月6日**		
2	确定项目范围	4 工时	2004年1月1日	2004年1月1日		管理人员
3	获得项目所需资金	1 个工作日	2004年1月1日	2004年1月2日	2	管理人员
4	定义预备资源	1 个工作日	2004年1月2日	2004年1月5日	3	项目经理
5	获得核心资源	1 个工作日	2004年1月5日	2004年1月6日	4	项目经理
6	项目范围规划完成	0 个工作日	2004年1月6日	2004年1月6日	5	
7	**分析/软件需求**	**14 个工作日**	**2004年1月6日**	**2004年1月26日**		
8	行为需求分析	5 个工作日	2004年1月6日	2004年1月13日	6	分析人员
9	起草初步的软件规范	3 个工作日	2004年1月13日	2004年1月16日	8	分析人员
10	制定初步预算	2 个工作日	2004年1月16日	2004年1月20日	9	项目经理
11	工作组共同审阅软件规范/预	4 工时	2004年1月20日	2004年1月20日	10	项目经理, 分析人
12	根据反馈修改软件规范	1 个工作日	2004年1月21日	2004年1月21日	11	分析人员
13	确定交付期限	1 个工作日	2004年1月22日	2004年1月22日	12	项目经理
14	获得开展后续工作的批准 (假	4 工时	2004年1月23日	2004年1月23日	13	管理人员, 项目经
15	获得所需资源	1 个工作日	2004年1月23日	2004年1月26日	14	项目经理
16	分析工作完成	0 个工作日	2004年1月26日	2004年1月26日	15	
17	**设计**	**14.5 个工作日**	**2004年1月26日**	**2004年2月13日**		
18	审阅初步的软件规范	2 个工作日	2004年1月26日	2004年1月28日	16	分析人员
19	制定功能规范	5 个工作日	2004年1月28日	2004年2月4日	18	分析人员

图 2-15　Project 主界面

在打开的软件开发文件中，我们可以看到左侧的任务分解列表中出现了有关软件开发的任务分解项；右边的甘特图区域中自动出现了相应的条状图形。

(2)任务分解列表的使用

如图 2-16 所示，在任务分解列表中，按照标题栏上从左至右的顺序依次介绍各项含义。

	任务名称	工期	开始时间	完成时间	前置任务	资源名称
1	**项目范围规划**	**3.5 个工作日**	**2004年1月1日**	**2004年1月6日**		
2	确定项目范围	4 工时	2004年1月1日	2004年1月1日		管理人员
3	获得项目所需资金	1 个工作日	2004年1月1日	2004年1月2日	2	管理人员
4	定义预备资源	1 个工作日	2004年1月2日	2004年1月5日	3	项目经理
5	获得核心资源	1 个工作日	2004年1月5日	2004年1月6日	4	项目经理
6	项目范围规划完成	0 个工作日	2004年1月6日	2004年1月6日	5	
7	**分析/软件需求**	**14 个工作日**	**2004年1月6日**	**2004年1月26日**		
8	行为需求分析	5 个工作日	2004年1月6日	2004年1月13日	6	分析人员
9	起草初步的软件规范	3 个工作日	2004年1月13日	2004年1月16日	8	分析人员
10	制定初步预算	2 个工作日	2004年1月16日	2004年1月20日	9	项目经理
11	工作组共同审阅软件规范/预	4 工时	2004年1月20日	2004年1月20日	10	项目经理, 分析人员
12	根据反馈修改软件规范	1 个工作日	2004年1月21日	2004年1月21日	11	分析人员

图 2-16　任务分解列表

①序号

任务分解列表中最左侧的数字表示的是任务序号。在任务分解表中填写的每一个任务都被系统分配一个数字。这个数字将会随着任务的上下移动而自动更改。同时，根据工作分解结构 WBS(Work Breakdown Structrue)，为每项任务在项目大纲结构中设置 WBS 代码，来唯一标识每项任务在大纲中的层次位置，如图 2-17 所示。

图 2-17　WBS 码在任务信息中的位置

在图 2-17 任务信息对话框中，WBS 码的位置里填写的编号是“1.2”。其中第一个“1”表示当前任务“获得项目所需资金”是属于第一个阶段性描述任务“项目范围规划”中的子任务，小数点后面的“2”表示 “获得项目所需资金”为当前阶段性描述任务的第二个子任务。

②信息备注

当某个任务项有信息需要提示时，  符号列则会有相应的符号加以标注。

例如，如果需要对相应任务填写一些补充信息时，双击当前任务项，在弹出的对话框中

填写相关的内容后，对应行就会出现一个备注标志。如图 2-18 所示摘要信息中添加备注，在保存备注信息后，在其左侧的信息提示栏就会出现备注的符号。

图 2-18　摘要信息中添加备注

③任务名称

对于分解任务的概括性描述。其中，任务名称前面带有"减号"或"加号"标志的，则代表该任务为阶段性描述任务(一系列任务的总称)。"加号"表示该任务项包含有子任务项，单击后可以进行展开；"减号"表示此任务也为阶段性描述任务，在其下可以看到该任务的子任务项，此时"加号"变成"减号"。点击"减号"后，该任务的子项将隐藏起来，"减号"变成"加号"。

④工期

完成当前任务需要花费的时间。工期的单位可以有如表 2-8 中所示描述的几种形式。点击任务项中的工期栏，可以通过上下按钮调整工期的长短。

表 2-8　　工期单位的几种形式

缩写	含义	缩写	含义
m	分钟工时(24 小时下的分钟)	em	分钟
h	工时	eh	小时
d	工作日	ed	天
w	周工时	ew	周
mo	月工时	emo	月

⑤开始时间

显示当前任务的开始年月日。点击该项时将出现一个日历项，用户可以方便地对日期进行可视化设置。

⑥完成时间

系统将自动根据工期和开始时间计算完成时间。系统也可以通过对完成时间的确定与设定的开始时间来自动更改工期。

⑦前置任务

前置任务限定了与当前任务之间存在的约束关系。在前置任务类型中分为 5 种。分别为“完成-开始(FS)”,表示前置任务完成后当前任务才能够开始;“开始-开始(SS)”表示前置任务开始后,当前任务就可以开始进行了;“完成-完成(FF)”表示前置任务完成时,当前任务也必须完成;“开始-完成(SF)”表示前置任务开始时当前任务必须完成。如图 2-19 所示给出了 Project 中任务类型列表。

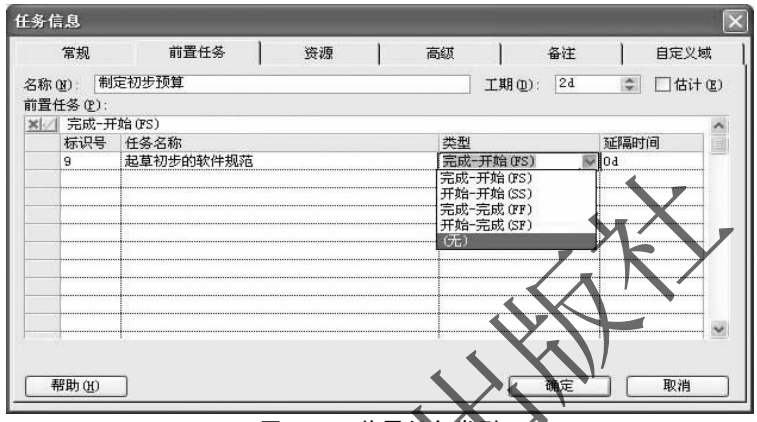

图 2-19　前置任务类型

⑧资源名称

当前任务完成所需要的资源项,这里主要指的是人力资源,表明当前的任务由谁来完成。

⑨甘特图(图 2-20)

在完成了任务分解列表之后,系统将自动形成甘特图。在图 2-20 中,我们可以很明显地观察到任务的先后关系和工期的长短以及任务人员的分派。

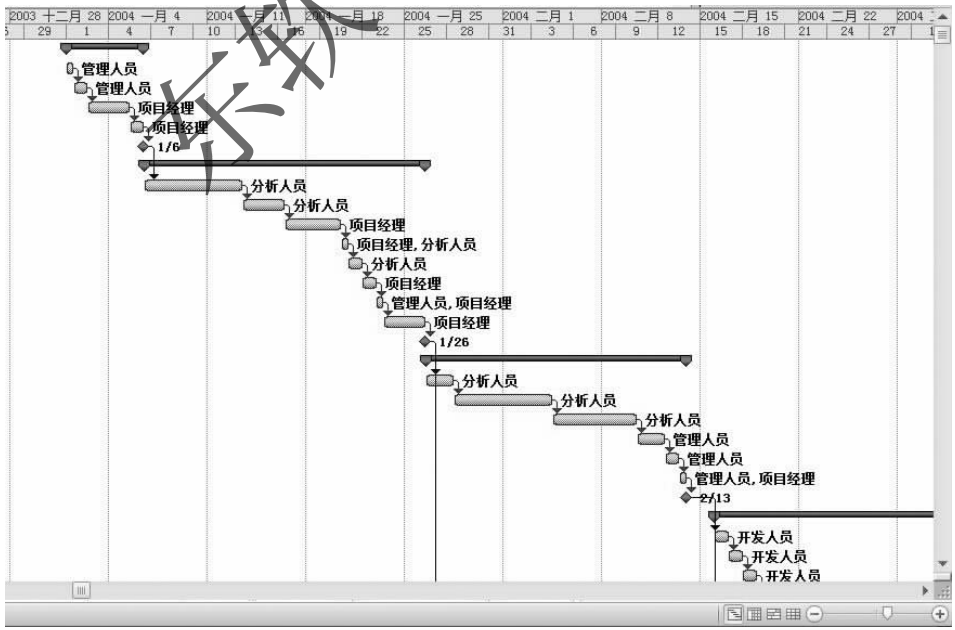

图 2-20　与任务分解列表相对应的甘特图

当项目计划涉及的时间较长时，观看完整的甘特图内容会比较小，此时我们可以通过界面最下方放大或缩小工具切换甘特图中的时间坐标的精度来改变视界。

(3)跟踪甘特图

点击如图 2-21 所示工具栏中“任务”项→“甘特图”下方的下拉列表，选择“跟踪甘特图”，出现跟踪甘特图(图 2-22)。它是以甘特图形成的结果为基础进行显示的。跟踪甘特图区别于甘特图的地方是它用红色表示未完成的任务，用蓝色表示完成了的任务。跟踪甘特图中可以对项目完成的百分比进行设置，便于项目经理对整个项目进度的监控。

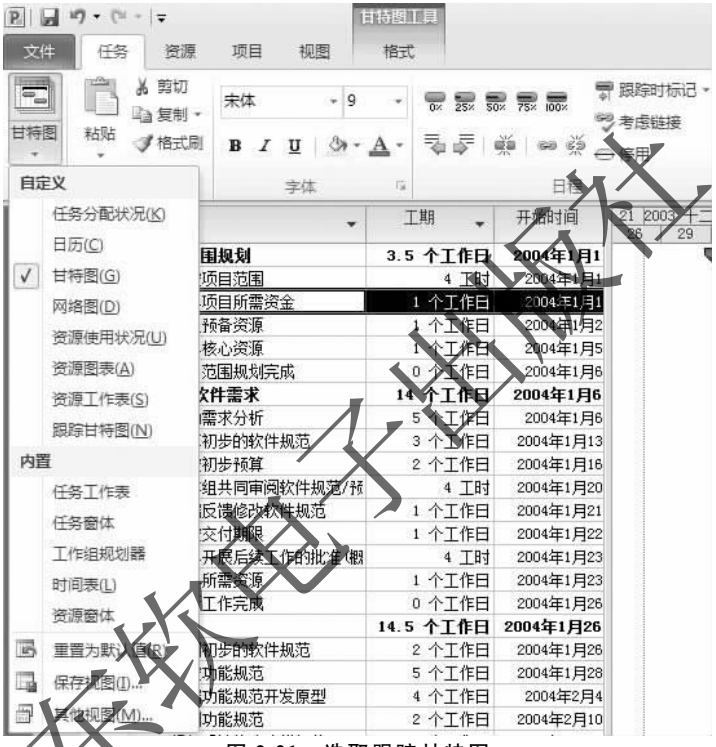

图 2-21　选取跟踪甘特图

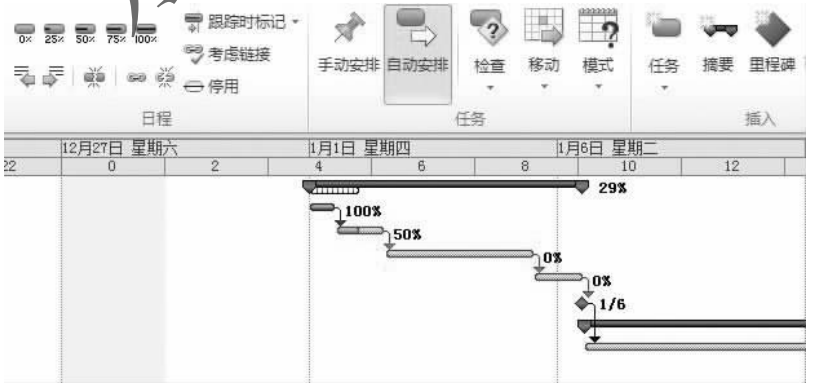

图 2-22　跟踪甘特图对项目的监控

(4)资源使用状况

在任务项中“甘特图”下方视图列表中选取“资源使用状况”，看到如图 2-23 所示资源使

用状况表。它表明列表左侧的资源(这里主要是人力资源的角色描述),在每天综合工作量的分配并以资源为单位列出与资源相关的任务,以便明确任务的分工。一般为 8 小时工作制,如果某天的工时超过 8 小时,则会用红色突出显示。

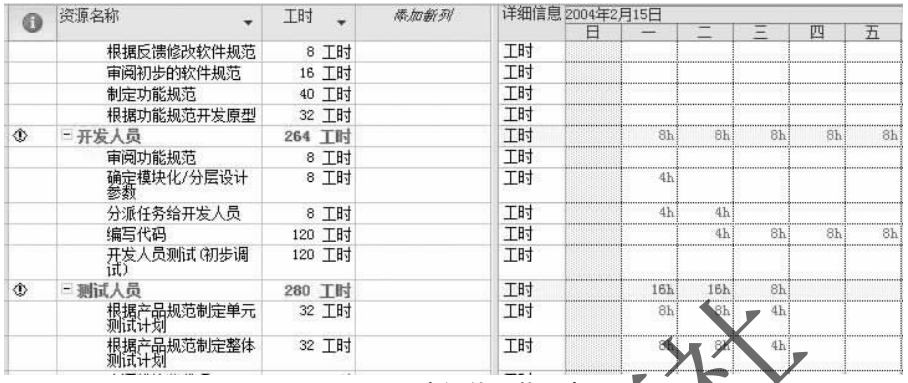

图 2-23　资源使用状况表

(5)日历

日历视图是以日程表的形式展示工作内容的一种方式,如图 2-24 所示。

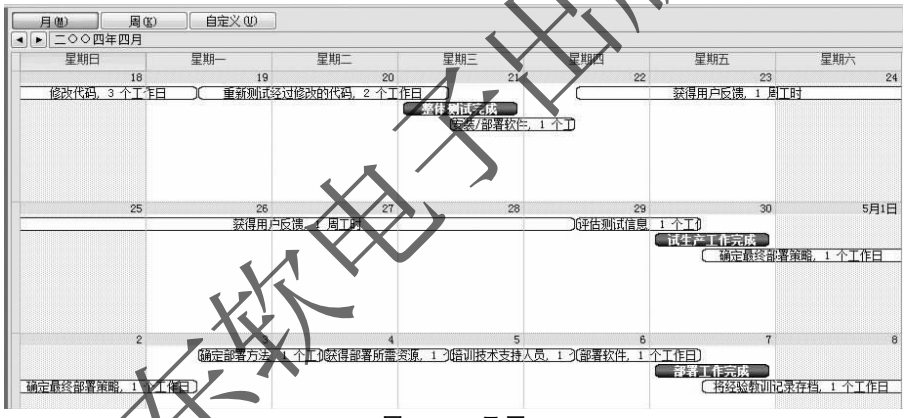

图 2-24　日历

(6)网络图

系统将根据甘特图任务列表中的内容转换成活动网络图,并自动用红色标识出关键路径上的关键任务。

(7)资源工作表

列出目前可以使用的资源的名称,同时对资源的单位成本进行设置,便于日后进行成本估算。

(8)资源图表

资源图表是以资源为单位,独立地进行统计。可以较为明显地看出是否存在资源过度分配的状况。通过移动图标左边的活动条,可以看到每个资源在每天的任务分配量。如果超过额定工作量,图表将会以百分比的形式将超额的部分以红色突出显示。资源图表与资源使用状况表可以参照使用。

(9)其他视图

如图 2-25 所示。

图 2-25 其他视图

其他视图对话框中显示了所有可以使用的视图,方便在各种视图中进行转换。

3.任务 3:自己创建一个项目进度计划

任务描述:

不使用模板,根据自己项目背景创建一个项目进度计划的过程。

实践步骤:

(1)新建项目计划文档

参照任务 1 的步骤 1 创建一个空白项目计划文档。

(2)设定项目整体信息

选择“项目标签”,点击“项目信息项”,设定项目的起始时间。如图 2-26 所示。

图 2-26 设定项目的起始时间

(3)输入任务信息,创建项目开发过程的框架

写出主要的任务框架名称。直接在 Project 中的任务列表的任务名称项中填写任务的名称,后面的工期、开始时间、完成时间、前置任务和资源名称可以先采用默认值,得到如图 2-27 所示任务列表。

	任务模式	任务名称	工期	开始时间	完成时间	前置任务
1		确定项目范围	1 个工作日			
2		制定项目计划	1 个工作日			
3		获取需求	5 个工作日			
4		需求建模	5 个工作日			
5		撰写文档	2 个工作日			
6		概要设计	4 个工作日			
7		详细设计	4 个工作日			
8		编写代码	10 个工作日			
9		单元测试	5 个工作日			
10		集成测试	3 个工作日			
11		确认测试	3 个工作日			
12		性能测试	3 个工作日			
13		部署	2 个工作日			
14		验收	1 个工作日			
15		总结	1 个工作日			

图 2-27　创建项目框架

(4)确定任务的隶属关系,分解框架任务,形成各自的子任务

①通过任务的升级、降级整理各任务之间的隶属关系。

②方法:用鼠标拖拽;通过工具栏上的“+”,“-”符号;右键菜单。

(5)设置工期,可以通过手动输入的方式更改工期的单位

(6)确定前后的链接关系(依赖关系)

①在任务列表中,选中有依赖关系的列,采用工具栏中的链接工具  进行设置;

②在任务列表中,编辑前置任务栏中的数据自行设置;

例如,前置任务栏中填写“3”或“1,3”,表明当前任务依赖于任务 3(或任务 1 和任务 3)的完成。

③通过在甘特图上进行拖拽设置。

确定链接关系后可以选中某一任务,然后点击右键选择“任务信息”项,对编辑好的内容进行修改。同时,可以通过“前置任务”的“延隔时间”列向一个依赖关系录入超前或滞后时间。

延隔时间为负数:

设置超前时间使得具有依赖关系的任务之间存在一定时间上的重叠。

延隔时间为正数:

前置任务完成后还要等待一段时间才开始下一项任务。

(7)设定里程碑(图 2-28)

①设定工期为 0,则自动表示为里程碑;

②设定任意大于 0 的工期,但需要在任务信息对话框中的高级选项中选定表示为里程碑选项。

图 2-28　里程碑的设定

(8)设定周期性任务(可选,图 2-29)

①在工具栏任务标签中选择任务的任务周期。

图 2-29　插入周期性任务

②填写周期性任务信息对话框中相应的选项,如图 2-30 所示。

图 2-30　周期性任务信息

(9)分配资源

①建立资源列表:点击资源工作表,填写资源名称;

②在甘特图中的任务列表里进行资源分配;

③通过“工具”→“分配资源”对话框进行分配；

④查看资源分配的统计性信息。

尽可能使得没有人员的超负荷工作(红色表示任务超负荷)。

(10)调整计划

根据项目的实际情况，进行上述各项的协调，以期达到合理的人员、时间的安排。

2.11.4　实践要求

(1)课前需要根据小组各自的选题进行讨论，确定粗略的项目规划。

(2)根据讨论的结果，将项目规划用 Project 工具表示出来，在课堂上给老师进行验收。

注意　此次项目的给定期限为 12 周，请根据各自的情况针对于软件生命周期的各阶段给出适当的期限。

2.12　项目实践——构建项目小组

2.12.1　实践目的

本学期要求学生随着教学的进度完成一个贯穿整门课程的自拟项目。“构建项目小组”实践环节是整个项目的启动环节。通过组建课程项目小组，确定项目名称，明确本学期要完成的任务目标，培养学生团队协作能力。在进行小组成员的职责分配时，锻炼学生的沟通、组织和协调的能力，培养学生在团队间协商、适当妥协的能力，同时也对工程实践过程中的角色职责有所了解。

2.12.2　知识点

1.项目小组的角色组成及其承担的责任

为了充分调动学生项目参与的积极性，现拟定如下 8 种角色。其中第 7 和第 8 种角色要求小组成员全部参与，其他 6 种角色由小组成员分别承担。角色进行分派以后，并不意味着担任某种角色的成员只完成这部分工作，而是在实践过程中协助项目组长主持工作，起到分阶段主要负责人的作用，而在其他阶段则需要配合阶段负责人完成相应工作。

(1)项目组长

整个小组的领导者，负责小组的协调、管理及任务的分配，在整个项目进行过程中组织会议，控制进度，协调各角色之间的关系，负责组织收集与提交项目文档。向老师负责。

(2)需求分析师

要求熟悉业务，保证整个项目的需求一致性，构建系统需求模型。在需求分析阶段负主要责任。向项目组长负责。

(3)系统架构师

负责分辨出影响系统架构的功能性与非功能性因素，协助需求分析师确定系统的核心业务，主要负责系统体系结构的合理构建。向项目组长负责。

(4)数据设计师

负责业务数据的搜集与整理,负责系统数据模型的构建,包括构建数据库、确定数据结构等内容。协助需求分析师和系统架构师进行系统的分析与构建。

(5)质量保证人员

在整个项目进行过程中进行质量的监督与控制。需要参与到各个阶段的决策过程当中,主要是持有一种质疑的态度从质量保证的角度对软件开发阶段性成果提出可能存在的问题、弊端,并给出相应的质量修改建议。

(6)界面设计人员

确定界面要素的合理性,负责构建系统的界面原型,辅助需求分析师进行需求的确认。同时界面设计人员还需要运用用户界面设计原则,对界面进行美观性、友好性等方面的设计,完成系统的最终界面设计。

(7)编程人员

负责用代码实现项目小组构建的系统。要求为全体小组成员。向项目经理负责。

(8)文档作者

负责编写项目文档。要求为全体小组成员。向项目经理负责。

2.任务分配列表(表 2-9)的使用说明

表 2-9　　任务分配列表

任务 / 角色	组织整个会议	绘制顶层图	逐层分解系统	确定需求优先级	建立数据模型	构建数据字典	其他信息的整理
项目组长	△	√	√				√
需求分析师	√	△	△	√	√	√	△
系统架构师		√	√	△	√	√	
数据设计师			√		△	△	
质量保证人员			√	√			√
界面设计人员		√					
编程人员							
文档作者		√	√	√	√	√	√

符号说明:

△ 代表主要负责人,是任务的发起者和领导者,负责构思整个任务的完成策略。

√ 代表当前任务的主要参与者,配合并协助主要负责人完成任务,是主要负责人的重要合作成员。

2.12.3　实施步骤

1.任务 1:组建项目组

(1)确定小组人数及成员

按照自愿的原则进行自由组合。人数不宜多,根据软件工程基本原理中对人员的要求,3～5人为一个合理的范围。

(2)小组成员的职责

参照知识点中提到的角色,为每一名小组成员分配角色,确保所有的角色都被分配给相应的小组成员。一名小组成员可以担任三种以上的角色。每一名成员至少要担任三种角色(也就是说每名小组成员必须担任一种除文档作者和编程人员以外的角色)。

①对于三人小组的职责分配参考方案:

1 人担任项目经理、需求分析师;

1 人担任数据分析师、界面设计人员;

1 人担任系统架构师、质量保证人员。

②对于四人小组的职责分配参考方案:

1 人担任项目经理、质量保证人员;

1 人担任需求分析师、界面设计人员;

1 人担任系统架构师;

1 人担任数据设计师。

③对于五人小组的职责分配参考方案:

1 人担任项目经理;

1 人担任需求分析师;

1 人担任系统架构师;

1 人担任数据设计师;

1 人担任质量保证人员、界面设计人员。

上述方案仅供同学进行参考,也可以提出不同的角色分配方案。

2.任务 2:确定项目组系统题目

(1)小组选题

请参考本教材附录 B 中提供的项目参考题目,选择项目组成员相对比较熟悉的项目背景。也可以自拟题目。

(2)确定项目背景

需要提交项目的简介。简介内容请参照附录 B。

2.12.4　实践要求

(1)项目组长根据《项目小组成员名单规范》提交小组的组成名单和项目简介。

(2)项目组长在组内明确角色分配方案,以备后续项目实践使用。

小　结

软件项目管理作为一种科学的管理手段,就是为了使软件项目能够按照预定的成本、进度、质量顺利完成,而对成本、人员、进度、质量、风险等进行分析和管理的一系列活动。决定一个软件项目实施成功与否,软件项目管理无疑起着举足轻重的作用,软件项目管理已经是公认的软件开发企业的核心竞争力之一。本章着重介绍了软件项目计划的概念及制定过

程、软件组织构成、质量管理、配置管理、风险管理等内容。

习　题

一、选择题

1. 项目经理在进行项目管理的过程中用时最多的是_____。

A. 计划　　B. 控制

C. 沟通　　D. 团队建设

2. 项目团队组建工作一般属于_____。

A. 概念阶段　　B. 开发阶段

C. 实施阶段　　D. 收尾阶段

3. 项目快要完成时客户想对工作范围作一大的变更，项目经理应该_____。

A. 进行变更

B. 将变更造成的影响通知客户

C. 拒绝变更

D. 向管理当局抱怨

4. 项目范围_____。

A. 只是在项目开始时才加以考虑

B. 在合同或其他项目授权文件被批准后通常就不成为问题

C. 应该从项目概念形成阶段到收尾阶段一直加以管理与控制

D. 主要是项目执行期间变更控制程序处理的一个问题

5. 项目工期紧张时你会集中精力于_____。

A. 尽可能多的工作

B. 非关键工作

C. 加速关键线路上工作的执行

D. 通过降低成本加速执行

6. 下列哪一项是质量控制的输出_____。

A. 统计抽样　　B. 质量管理计划

C. 工作结果　　D. 过程调整

7. 下面四个选项中哪一项与风险影响分析最相关_____。

A. 风险管理　　B. 风险评估

C. 风险识别　　D. 风险减轻

8. 小组成员完全平等，享有充分的民主，通过协商做出技术决策，这种组织程序员组的方法称为_____。

A. 主程序员组　　B. 民主制程序员组

C. 现代程序员组　　D. 传统程序员组

9. 软件质量必须在_____加以保证。

A. 设计与实现过程　　B. 开发之前

C. 开发之后　　D. 开发期间

10. 为了保证软件质量，在开发过程的各个阶段进行_____是一个重要手段。

A. 验收测试　　B. 用户培训

C. 软件评审　　D. 文件修改

二、简答题

1. 请分析一下导言中软件项目实施过程中罗列的任务属于哪个方面的管理任务。

2. 某软件项目需 40 名开发人员。有两种人员组织方案：40 人归为一组，或者将 40 人分为 8 组。试比较两种方案的优劣并说明理由。

3. 比较 CMM 与 ISO9000 两者的异同。

4. 请描述 CMM 分级结构及主要特征。

5. 风险管理的目标是什么？工程项目的哪些特征使之有风险？

三、综合练习

1. 假定要开发一个图书馆管理系统，你是该项目的软件系统负责人。请为该项目的软件开发制定切实可行的规划。

提示：

(1)可根据软件的生命周期进行计划的制定，整个工期设定为 100%，则各阶段所花费的时间可按百分比给出。

(2)人员的分配可按照角色给出。角色包括以下几种：项目经理、系统分析员、软件架构师、程序员、测试人员、集成人员和客户等。

2. 假设你被指派为一个软件公司的项目负责人，你的任务是开发一个技术上具有创新性的产品，该产品把虚拟现实硬件和最先进的软件结合在一起。由于家庭娱乐市场的竞争非常激烈，这项工作的压力很大。你将选择哪种项目组结构？为什么？你打算采用哪种(些)软件过程模型？为什么？