第3章 软件项目管理

项目是一个具有工程独立性的工程作业单元，并是一个可将人、财、物组合在一起的工程容器。软件的工程模式开发即以项目为单位进行，并通过项目实施有效管理。

本章要点：

- 软件开发机构、项目小组；
- 项目计划与项目成本估算；
- 软件风险管理、软件文档管理；
- 软件配置管理、软件质量管理。

3.1 开发团队

大规模的软件系统通常是由团队开发的，这些软件是否能够成功开发出来，不仅依赖于开发者的个人才智，还依赖于团队成员之间的良好协作。因此，如何将开发人员有效组织起来，由此建立一个优秀的软件开发团队，是一项非常重要的工作。

3.1.1 软件开发机构

所谓软件开发机构，也就是专门承担软件研发任务的公司、部门或科研院所。显然，它需要有健康的能很好适应软件开发任务的组织机体，如组织结构、制度、文化等将影响这个组织机体的健康度。实际上，一个有健康机体的开发机构，其自身就是一个具有战斗力的最大化的开发团队，而基于其组织结构，其内部又将形成诸多小的开发团队。

无疑，专门从事软件产品研发的软件公司是产业化软件开发的主力军团。如图 3-1 所示为比较常见的软件公司组织结构，其主要职能部门有质量控制部、产品开发部、产品支持部、产品服务部等。

下面是对各职能部门的说明：

(1) 质量控制部：提供软件质量标准，负责各阶段软件成果评审，负责软件开发过程质量控制，以及产品服务质量监督。质量控制部大多设置于组织结构较高层次，以获得对整个项目有效的质量监控。质量控制部大多只有少数固定成员，主要任务是质量管理，而当需要进行质量评审或质量鉴定时，则从产品开发部、产品支持部以及产品服务部邀请技术骨干，或是从有关科研院所邀请技术专家临时担任。

(2) 产品开发部：负责软件开发管理。通常管理多个软件项目小组，每个项目小组有

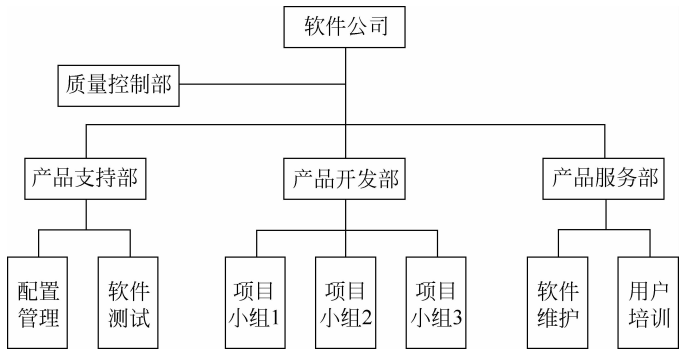

图 3-1 软件开发公司组织结构

不同的软件开发任务。大多数的项目小组是短期的，随承接软件任务而专门设置，并随软件任务的完成而解散。

(3) 技术支持部：负责较长期的产品技术支持，如：配置管理、产品维护等。

(4) 产品服务部：负责较长期的产品后期服务，如：产品推广、用户培训等。

3.1.2 软件项目组

项目组是最小的软件开发团队，通常因项目任务组建。考虑到项目组内成员之间通信与协作的便利，项目组要求小而精，成员大多限制在 8 人以内。

项目组成员角色有：

- 项目组长：负责制定工作计划，负责任务分配与协调，负责项目成果评审。
- 开发人员：按所分配的工作任务从事软件开发，包括软件分析、设计与编码。
- 资料管理员：负责成果归档，进行软件配置。
- 软件测试员：负责软件模块测试与系统集成，进行软件质量控制。

上述角色可以分别由不同的成员担任，但一个人数只是 3 人以内的项目小组，则每个成员往往需要承担多个角色的任务。

很显然，项目组绝不只是项目成员的简单组合，而是必须体现出团队精神，成员之间可良好协作，能将个人目标与项目组团队目标结合在一起，能够为团队的成功而发奋努力。

无疑，项目组长是最核心成员，他是项目小组的领导者，其不一定是技术专家，但必须是管理专家，需要制定项目计划，需要分配与协调项目任务，需要处理成员关系，需要鼓舞成员士气。

通常，可从以下方面对项目组进行合作性评价。

- 成员在技术、经验和个性上是否有良好互补性。
- 成员是否对项目组有良好的团队认同感。
- 成员之间是否有良好的情感沟通。
- 成员在团队中是否有良好的受尊重感。
- 成员对所扮演角色是否有较高的满意度。

3.1.3　项目组管理机制

1. 民主分权制

项目组成员平等地以民主协商形式做出项目决策，诸如项目计划、任务分配、工作制度、设计方案等，都需要集体讨论。

民主形式下也有项目负责人，但并不固定，可随时由其他成员替代。项目组长的核心作用不是很明显，往往只是项目会议的召集者，项目任务的协调者，承接业务的联系人。诸多问题，如：项目计划、任务分配、工作制度、设计方案等，都需要项目组成员集体讨论才能定夺。

民主形式的优点是，成员可较充分表现个人作为，可保持较高的工作热情；成员之间可以轻松交流，并有很好的技术协作。通常认为，民主形式能获得较高的团队凝聚力，可带来较浓厚学术风气，有利于开发者能力的培养，有利于技术攻关。

然而，民主形式成员之间有途径太多的信息通道，如果一个项目组有 n 个成员，则可能的沟通信道有 $n(n-1)/2$ 条。因此，许多决策往往需要花费很长时间才能意见一致，以致影响项目进度，降低工作效率。

民主形式项目组对成员素质一般有较高要求。如果项目组成员都是经验丰富、技术熟练，则所承担软件项目往往比较成功。但如果组内成员技术水平不是很高，则由于管理上缺乏核心领导，技术上缺乏权威指导，则软件项目容易失败。

2. 主程序员负责制

一种对项目组实施集权控制的管理体制，项目组主要成员有：主程序员、后备程序员、资料管理员与一般程序员等，其组织结构如图 3-2 所示。

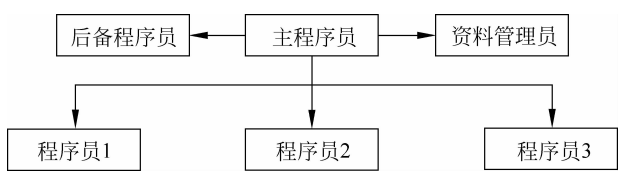

图 3-2　主程序员负责制

很显然，主程序员处于该组织结构的绝对核心位置，因此，需要由资深软件工程师担任，要求经验丰富、技术能力强，将负责整个项目计划的制定与实施，负责软件体系结构与接口设计，负责关键算法设计，并承担对其他成员的工作指导。

后备程序员则是主程序员的得力助手，需协助主程序员工作，并要求在必要时能够接替主程序员的工作。此外，还需要负责制定测试方案、分析测试结果及其他独立于设计过程的工作。

资料管理员则负责软件成果归档与软件资源配置等方面的事务性管理工作。

主程序员负责制有较严密的组织结构。因此，比较起民主分权制，这种管理机制更加规范，项目组成员任务分工更加明确，项目中各项工作的开展更有条理与次序，可带来较高的开发效率与开发质量。

然而，主程序员负责制有以下方面的不足。

其一，对主程序员的依赖度较高，一旦主程序员离开了项目组，则项目有可能瘫痪。另外，由于一切工作以主程序员为中心，项目计划、设计方案等都是由主程序员制定，其他成员只需按部就班地开展工作，缺少发言权，以致创造性受到压抑。

其二，主程序员有过高的工作负担，既要负责项目管理，又要解决技术问题。

其三，主程序员作为技术专家，一般更加关注技术问题，而比较疏忽管理艺术，以致对其他成员可能是严格要求有余而沟通关心不够，因此难免影响其他成员的工作积极性。

3. 职业项目经理负责制

职业项目经理是一些经过专门训练与考试认证，由此具有软件项目管理执业资格的专业软件项目管理人才。

由于主程序员负责制有偏重技术而忽视管理的弊端，因此现在很多软件项目都采用了职业经理负责制，以满足项目负责人首先应该是管理者的要求。

然而，职业项目经理大多并不精通技术，一般只负责项目管理，如：制定项目计划、分配项目任务、进行项目成本预算、按照里程碑进度计划监控项目进程。正因此，职业项目经理与项目组从事技术的其他成员之间容易出现情感隔阂，一些技术人员甚至可能把他看做外行，而对他有敌对情绪，不愿服从他的安排。为了解决这个问题，往往有必要给项目经理配备一个精通技术的副手，如：技术负责人，其承担原主程序员需要承担的技术任务。

如图 3-3 所示为职业项目经理负责制的一般组织结构。

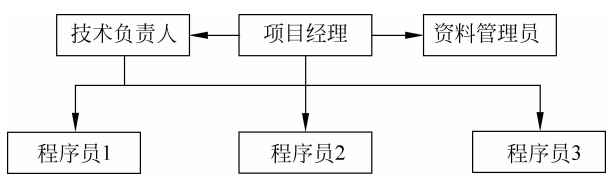

图 3-3　职业项目经理负责制组织结构

4. 层级负责制

软件系统规模可大可小。规模较小的软件系统，一个 3～5 人的项目小组即可承接下来，并可在规定期限内开发出来。但大规模软件系统，若要在规定期限内开发出来，则不得不组织起几十人的开发队伍。

显然，项目参与人数越多，则成员之间的通信与协作越不便利。通常情况下，项目小组人数应限制在 8 人以内，因此，当项目规模庞大而需要几十人参与时，有必要将一个大项目分解成许多个小项目，然后组织多个项目小组实施开发。

常用的项目分组方法是，将系统按功能分解为若干个具有一定独立性的子系统，然后再按子系统进行项目任务分组，每个项目小组只需要完成分配的子系统的开发任务，由此形成的人员管理体制即为层级负责制。

如图 3-4 所示为层级负责制的一般组织结构。

层级负责制体现出了上级对下级基于任务分解的指挥控制，项目经理分解项目任务并分配给各项目小组，项目组长则将所承担的小组任务做进一步分解并分配给各小组成员。

层级负责制是由多个项目小组共同完成一个项目任务，因此各项目小组之间必然涉及

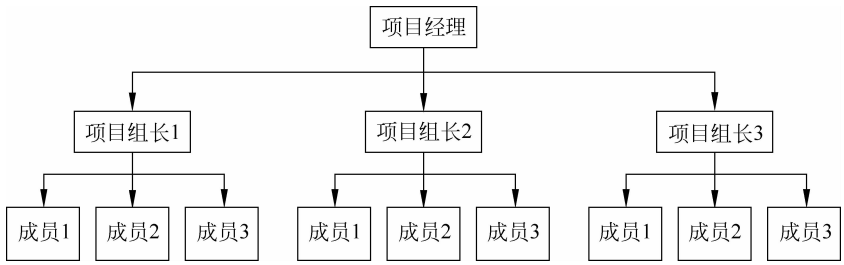

图 3-4　层级负责制组织结构

任务协调。

应该基于里程碑进程进行小组任务协调。例如，整个项目有一个统一的整体里程碑进度计划，各项目小组则基于项目整体计划，制定出自己承担的子项目的里程碑进度计划，以保证局部子项目计划与项目整体计划的协调。

有必要使各个项目小组之间有较低的相互依赖，由此可减轻因依赖而带来的协调工作负担。因此，分配给各项目小组开发的子系统，应尽量能够独立运行，以使得各项目小组任务有较好的独立性。

3.2　项目计划

一个缺失计划的软件项目往往会出现这样或那样的混乱。因此，为使软件开发中的各项工作能够有秩序地进行，项目管理者必须事先制定出项目开发计划。

3.2.1　任务分配

软件开发面临各种各样的任务，如：需求分析、结构设计、程序编码、文档管理，等等，为确保这些任务按期完成，诸多任务必须合理安排到人，这即是任务分配。

显然，一些大的任务单位需要分解细化，以方便项目任务可合理地适度地分配到各项目成员身上。然而，如何进行项目任务分解呢？

首先要进行的是基于软件生命周期的阶段划分。整个软件开发可划分为需求分析、概要设计、详细设计、编码与单元测试、系统集成等诸多阶段任务。

接着是对这些阶段任务做进一步的任务细分，以使分配到任务承担者身上的各项任务，能够内容更具体、目标更清晰、责任更明确。例如：软件结构设计，即可进一步分解为软件构架设计、软件接口设计、数据库设计。

任务树是一种最常使用的任务分解工具。如图 3-5 所示，即是基于任务树的任务逐层分解，其中的任务树根节点是对任务的整体概括，而树中根节点以下的各级子节点，则实现了对任务的逐级分解。

3.2.2　进度计划

项目开发计划的核心是进度计划。应该基于里程碑制定项目进度计划，以使项目中的任务及资源能按照里程碑进行合理的流程部署。

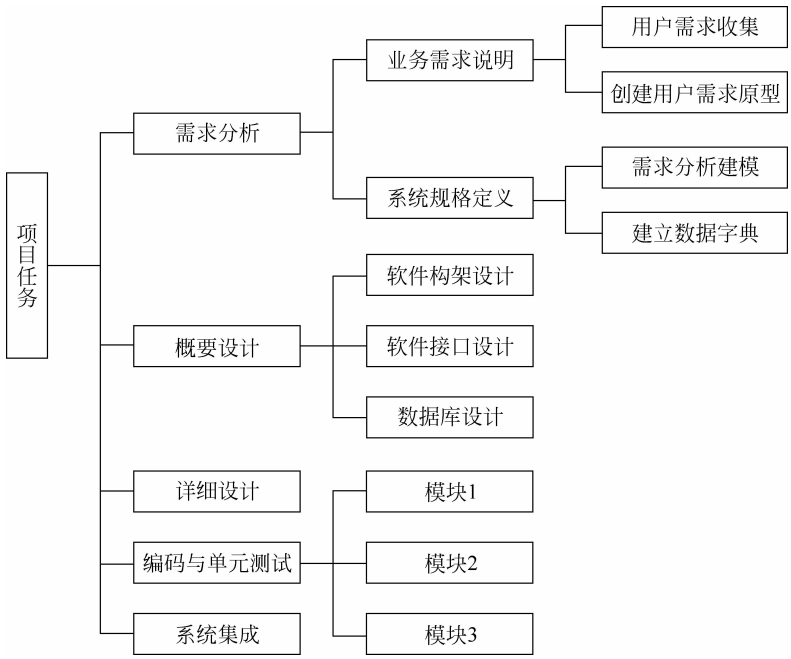

图 3-5 项目任务树

通常情况下，一个基于里程碑的进度计划，能够建立起对项目进程的量化监控。实际上，如果项目中的每一项任务都有了比较确定的工作量比例，并还能通过关键成果对该任务是否完成做出判断，则项目进展就有了能够量化的完成比例说明。

在制定项目进度计划时，需要重点考虑关键任务，其对项目进程有至关重要的影响。

项目进程中的里程碑标志性成果有需求规格说明书、系统设计说明书，关键任务与这些里程碑标志性成果直接相关。此外，一些直接影响产品质量的任务，一些涉及不确定性技术因素的任务，一些特别耗时的任务，也是关键任务。

进度计划是对任务合理的时序安排。首先是避免任务重叠，以减少人力及资源浪费，并提高开发效率。应特别注意关键任务的延期对整个项目进程的影响。一些无时间关联的任务则可并行开展，以加快任务进程。

进度计划制定中还须想到的是，项目初期对问题的考虑往往有不确定性。实际上，项目进行中也难免会有不确定性事件发生，例如，项目组中的某个成员可能因生病而请假，或是因某种原因离职；项目中的某台设备可能出了故障；项目遇到了事先没有估计到的技术性困惑，等等。因此，在制定项目进度计划时，需要考虑到一定的工作弹性，需要有一定的时间宽松，以利于从容应对不确定性因素，以使得当项目的实际进程与先期进度计划有较大偏差时，可根据项目实际进程对进度计划进行调整。

有许多工具可用来帮助建立项目进度计划，如：甘特图、任务网络图。下面介绍这两种进度计划辅助工具。

1. 甘特图

甘特图(Gantt)是最常用的项目进度图，其建立在日历表上，可直观地反映项目进展情况。

甘特图中的长条图形表示项目中的每项任务。长条形的起点为任务的开始，长条形的终点为任务的结束，任务进展可通过长条图形内不同颜色的细线表示，并可使用一些特殊图标（如：菱形）表示任务里程碑。

甘特图中任务之间的依赖关系则通过链接的箭头线表示，用于表明只有当前一项任务完成之后，下一项任务才能启动。大多数情况下，依赖关系基于里程碑建立。

甘特图能较全面地反映项目开展情况，并能方便进行基于里程碑的项目管理。

如图 3-6 所示是一个有关软件项目的甘特图。这是一个与图 3-5 中的任务树相关联的甘特图，可安排任务序列，并可表示任务进展、任务依赖关系、任务里程碑等。

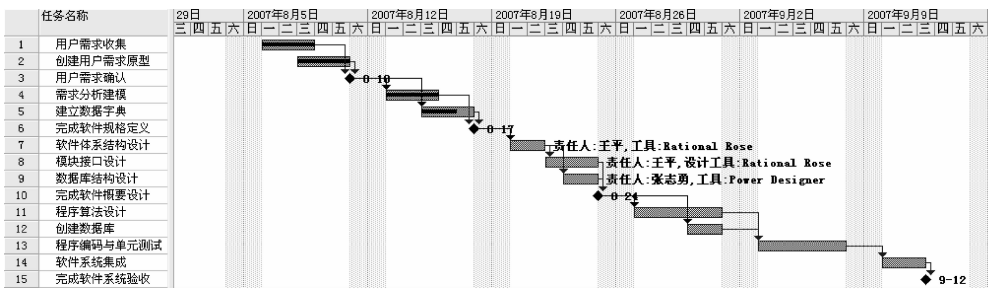

图 3-6　使用甘特图描述项目进度

2. 任务网络图

任务网络图（PERT）则基于项目任务流程创建，使用矩形表示任务，使用箭头表示任务前进方向。如图 3-7 所示即为任务网络图。

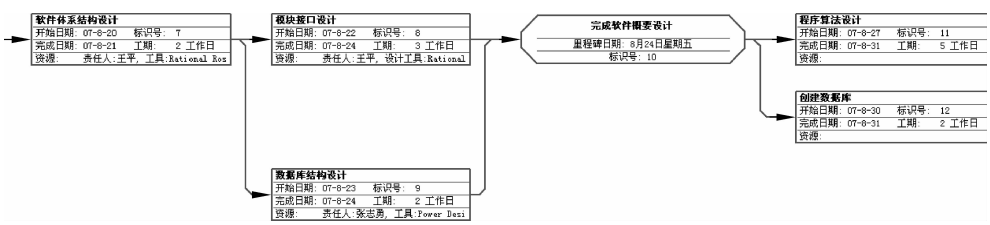

图 3-7　使用任务网络图描述项目进度

许多项目任务需要并行开展，并需要根据任务依赖关系、成员技术特点与协作性等，进行任务分配与调度。对此即可通过任务网络图给予直观描述。

当涉及并行任务时，耗时最长的路线将决定项目进度，须得到最多关注，可看做关键路径。任务网络图可非常直观地表现关键路径，例如图 3-7 中的模块接口设计与数据库结构设计这两项并行任务，由于模块接口设计耗时更多，因此被表示为关键路径。

任务网络图可给项目任务资源配置带来便利。在任务网络图中，任务承担人、任务所需设备、软件工具等是与任务项结合在一起的，因此能够得到更加明确的表达。

3.2.3　项目开发计划书

项目计划书是项目计划的具体体现，可作为软件开发的工作指南。

项目计划书涉及内容有：项目任务分解、资源配置、成本估算、进度安排等。下面是项目计划书的基本格式。

1. 引言
    1.1 编写目的(说明项目计划书的编写目的及阅读对象)
    1.2 项目背景(说明项目来源、开发机构和主管机构)
    1.3 定义(说明计划书中专门术语的定义与缩写词的原意)
    1.4 参考资料(说明计划书中引用的资料、标准或规范的来源)
2. 项目概述
    2.1 工作内容(说明项目中各项工作的主要内容)
    2.2 条件与限制(说明影响软件开发的各项约束条款，如：项目实施应有条件、已有条件、尚缺条件，用户及项目承包者需承担的责任，项目完工期限等)
    2.3 产品
        2.3.1 程序(说明需移交用户的程序的名称、编程语言、存储形式等)
        2.3.2 文档(说明需移交给用户的文档的名称及内容要点)
    2.4 运行环境(包括硬件设备、操作系统、支撑软件及其协同工作应用程序等)
    2.5 服务内容(说明需向用户提供的各项服务，如：人员培训、安装、保修、维护与运行支持等)
    2.6 验收标准(说明用户验收软件的标准与依据)
3. 实施计划
    3.1 任务(分解项目任务，确定任务负责人)
    3.2 进度(使用图表安排任务进度，涉及任务的开始时间、完成时间、先后顺序和所需资源等)
    3.3 预算(说明项目所需各项开支，如：人力费用、场地费用、差旅费用、设备资料费用)
    3.4 关键问题(说明将影响项目成败的关键问题、技术难点和风险)
4. 人员组织及分工(说明本项目人员组织结构，参入者基本情况等)
5. 交付期限(说明项目完工并交付使用的预期期限)
6. 专题计划要点(简要说明项目实施过程中需制定的其他专题计划，如：人员培训计划、测试计划、质量保证计划、配置管理计划、用户培训计划和系统安装计划等)

3.3 项目成本估算

项目必然涉及成本。通常情况下，在项目初期，如：项目立项论证时，或制定项目开发计划时，需要对项目成本有一个合理的初步估算，然后以该成本估算为依据，对项目以后的具体实施进行必要的成本控制。

软件项目涉及多方面的成本，如：工资开支、场地费、差旅费、设备费、资料费，并且这些费用都有不确定性。因此，对软件项目的成本估算往往有较大的模糊性。尽管如此，项目成本仍要估算，并需要以此为依据进行成本控制。

软件项目的最大成本是人员工资成本。一些常用成本估算方法，如：程序代码行成本估算、软件功能点成本估算、软件过程成本估算，所考虑的主要是人力成本。下面介绍这几种成本估算方法。

3.3.1　程序代码行成本估算

代码行成本估算是一种传统的基于软件规模的成本估算方法。显而易见的是，软件越复杂，软件规模越大，则构造软件的程序代码行数将越多，软件成本将越高。

1. 估算代码行数

程序代码行成本估算建立在对程序代码行数有较准确估计的基础上。基于功能的程序分解是一种比较有效的代码行估算策略，其方法是对程序系统进行逐级的功能分解，直到分解出足够简单的，或是已有代码行估算依据的程序功能元素为止。

很显然，估算一个软件的代码行数是一件非常耗时并非常复杂的工作。实际上，仅依靠人力手工计算，要比较准确地估计出程序的代码行数并非易事。因此，有必要建立一个代码行估算程序，以减轻这项工作的复杂度与劳动强度。

下面是软件代码行估算的一般过程的算法说明。

```
确定软件功能范围;
分解软件功能为诸多功能项;
将软件功能项加入功能表;
Do While 功能表中有没搜索到的功能项
    从功能表中选择功能项 i;
    将功能项 i 分解为诸多子功能项;
    将子功能项加入子功能表;
    Do While 子功能表中有没搜索到的子功能项
        从子功能表中选择子功能项 j;
        If 子功能项 j 在代码行估算历史数据库中有类似功能元素 p 对应
        Then
            通过功能元素 p 获取子功能项 j 的代码行数;
            将子功能项 j 的代码行数累加到总代码行数中;
        Else
            If 子功能项 j 已简单到能够直接估算代码行数
            Then
                估算子功能项 j 的代码行数;
                保存子功能项 j 到代码行估算历史数据库中;
                将子功能项 j 的代码行数累加到总代码行数中;
            Else
                将子功能项 j 继续分解为更小的子功能项;
                将这些更小的子功能项加入子功能表;
            End If
        End If
    End Do
End Do
```

2. 计算人力成本

上面介绍了如何估算程序代码行数。实际上，如果已经估算出了程序的总代码行数，则根据项目组成员的人月均程序代码生产率和人月均工资，即可非常简单地计算出程序的人

力成本。下面是程序代码行成本估算的计算式：

人力成本＝（总代码行数 / 人月均代码行创建数）×人月均工资

例如，某程序总代码行数估算结果是 20 000 行，而项目组成员平均每人每月能够开发 1000 行代码，假如每人每月平均工资是 4000 元，则由上面的计算式可计算出该程序人力成本是 80 000 元。

3. 方法局限性

实际上，程序成本不能单凭代码行数决定。来自设计者的质疑是，因算法设计优良而只需较少代码即可实现的程序，似乎比因欠缺良好设计而必须依靠编写大量代码以实现功能的程序有更低的成本。

代码行估算的局限性还体现在需要对程序做很详细的功能分解上。然而，一个很大规模的程序系统，要在项目早期即做出很详细的功能分解并非易事，因此估算结论必然有很大的不确定性。

3.3.2 软件功能点成本估算

功能点成本估算是一种建立在应用层上的能够更好适应项目早期的软件成本估算方法，由 Albrecht 首先提出。

应该说，前面的代码行估算是一种并不经济的成本估算，需要太多的人力成本代价。功能点估算则更加考虑成本估算自身的经济性，它不需要对软件进行逐级的深度功能分解，而只要对软件按区域估算功能点数，因此有相对较少的人力成本消耗。

1. 功能元素

软件功能点成本估算需要考察的对象是功能元素，其一般做法是将软件划分为用户输入、用户输出、用户查询、内部存储、外部接口等几个功能区域，然后基于这些区域分类考虑功能元素。

（1）用户输入：一般对应于一个相对完整的输入功能，如：报名登记、学籍注册。

（2）用户输出：一般对应于一个相对完整的输出功能，如：报表输出、屏幕输出、提示消息。

（3）用户查询：一般对应于一个相对完整的查询功能，并通常与一个联机操作相关联。例如：输入查询条件、系统根据查询条件进行查询计算、产生查询结果。这样一个完整查询过程可看成是一个查询元素。

（4）内部存储：一般对应于一个与功能实现相关联并与外界无直接关系的内部数据存储，如：数据文件、数据表。

（5）外部接口：一般对应于一个相对完整的与外界其他系统的交流，如：数据导入处理，数据导出处理。

通常，软件的每个功能元素可通过加权因子反映其复杂程度。表 3-1 列出了不同区域中的功能元素的加权因子的取值范围。

表 3-1　不同功能元素的加权因子的取值范围

功能元素 \ 复杂度	低	中等	高
用户输入	3	4	6
用户输出	4	5	7
用户查询	3	4	6
内部存储	7	10	15
外部接口	5	7	10

2. 功能数

软件中诸多功能元素的加权因子之和即为该软件的功能数。

例如，某软件有一个数据录入窗（高复杂度），一个用户注册窗（中等复杂度），一个用户登录窗（低复杂度），一个数据汇总报表打印输出（中等复杂度），一个数据查询窗（中等复杂度），一个数据存储表（中等复杂度），一个用户表（低复杂度），一个数据文件导出通道（中等复杂度）。则该软件功能数的计算是：

$$\text{功能数}=6+4+3+5+4+10+7+7=46$$

3. 功能点数

通过软件功能数即可计算出软件的功能点数。

软件功能点数的计算式是：

$$\text{功能点数} = \text{总的功能数} \times \left(0.6+0.01\times\sum F_i\right)$$

上面表达式中的 F_i 是软件复杂度调整值。通过回答表 3-2 中的问题，可确定软件复杂度调整值。其有 14 个关联程度问题需要回答，可做出的判断有："没有关联——0"、"微弱关联——1"、"轻度关联——2"、"一般关联——3"、"较大关联——4"、"很大关联——5"。

表 3-2　软件复杂度调整值

序号	问　　题	关联程度（F_i）		
		微弱关联——1	一般关联——3	很大关联——5
1	数据备份与恢复？	核心数据需要备份	基础数据需要备份	全部数据需要备份，并需记录工作日志，以便于恢复系统
2	数据通信？	需要远程数据验证	需要远程数据存储	多用户远程数据通信协作
3	数据分布式处理？		事务数据本地处理，基础数据服务器提供	
4	高性能要求？	需要有良好的交互响应	需要快速导入大批量数据	需要对环境数据进行实时监控
5	高负荷操作环境？	涉及大批量数据汇总计算	须同步处理多项事务	需要多服务器支持，以应对多用户服务阻塞
6	联机输入输出数据？		需要	
7	多窗口切换？		需要	