第3章 软件项目管理

项目是一个相对独立的工程单元，是一个可将与项目任务相关联的人、财、物等诸多要素组合在一起的工程容器。软件开发基于项目实施，由此组建项目团队，制订项目计划，评估项目风险。

本章要点：

- 软件研发团队
- 软件项目计划
- 软件项目成本估算
- 软件项目风险
- 软件项目文档、配置与质量管理

3.1 软件研发团队

大型软件系统通常都是基于团队研发的。实际上，一个软件产品是否能够成功研发出来，不仅依赖于软件研发者的个人才智，还依赖于软件研发团队成员之间的良好协作。因此，如何组建软件研发项目团队，并带好这个团队，使其成为一个优秀的研发团队，也就成为一件非常重要的工作。

3.1.1 软件研发机构

所谓软件研发机构，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，它需要有健康的并能很好地适应软件研发任务的组织机体，如组织结构、制度、文化等将影响这个组织机体的健康度。

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、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。

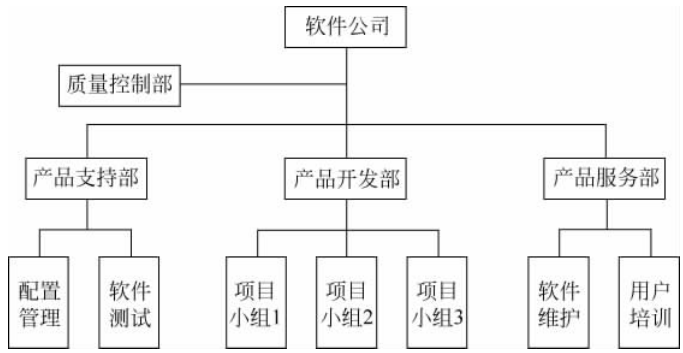

图 3-1 软件开发公司组织结构

(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 软件项目小组

一个有健康机体的软件研发机构，其自身就是一个具有战斗力的研发团队，而为了使软件研发任务能够基于项目有效开展，就还需要基于项目任务组建项目小组。

通常说来，软件项目小组是软件研发机构中被划分到最底层的最小的软件研发团队，其大多因软件项目任务组建。

考虑到软件研发团队成员之间通信与协作的便利，项目小组要求小而精，小组成员大多被限制在 8 人以内。

项目组成员角色包括：

- 项目组长：负责制订工作计划、分配与协调任务、评审项目成果；
- 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，但一个人数不足 3 人的项目小组，则一个团队成员就不得不承担多个角色的任务。

必须注意到的是，项目小组必须要有团队凝聚力，团队成员之间要有良好的协作氛围，要能体现出团队精神，要基于项目任务建立团队共同的追求目标。

毫无疑问的是，项目组长是这个团队最核心的成员，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，需要制订项目计划，需要分配与协调项目任务，需要处理成员关系，需要能够鼓舞成员士气。

通常，可从以下方面对项目小组进行内部合作性评价。

- 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　项目小组管理机制

1. 民主分权制

项目小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，可由其他成员替代。因此，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的要求。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，软件项目容易失败。

2. 主程序员负责制

主程序员负责制是一种以主程序员为核心的团队工作机制。通常看来，这是一种能对项目团队实施集权控制的管理机制，其项目小组成员包括主程序员、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。

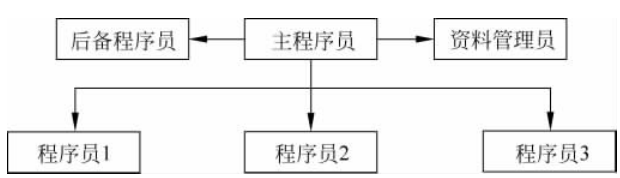

图 3-2　主程序员负责制

很显然，主程序员负责制中，主程序员是处于该组织体系的绝对核心位置，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订与实施，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。

后备程序员被看做是主程序员的得力助手，须协助主程序员工作，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。

主程序员负责制有比民主分权制更严密的组织体系,并有严格的管理制度,团队成员任务分工明确,项目中各项工作基于计划开展。也因此,在大多数情形下,其能够获得比民主分权制更高的团队工作效率与工作质量。实际上,以往在许多人看来,主程序员负责制还是一种能够与结构化工程方法相匹配的项目团队组织机制。也因此,在结构化工程时代,主程序员负责制是最常采用的项目团队组织形式。

然而,主程序员负责制也有以下方面的不足。

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,以致其他成员的创造性受到压抑。

其二,主程序员有过高的工作压力,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,以致对项目团队中其他成员可能是严格要求有余而沟通关心不够,因此难免会影响项目团队中其他成员的工作积极性。

3. 职业项目经理负责制

职业项目经理通常是一些经受过专门训练或考试认证的,由此具有软件项目管理资质的专业管理人才。

由于主程序员负责制有偏重技术而忽视管理之弊端,因此现在许多软件项目就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,其一般只负责项目管理,如制订项目计划、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作外行,而对他有敌对情绪,不愿服从他的安排。为了解决这个问题,一些软件研发机构就采用了给项目经理配备精通技术副手措施,是专门的技术负责人,其承担原主程序员需要承担的技术任务。

图 3-3 所示为职业项目经理负责制的一般组织结构。

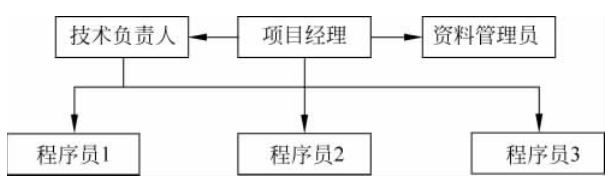

图 3-3 职业项目经理负责制组织结构

4. 项目层级负责制

软件系统规模可大可小。规模较小的软件系统,一个 3 到 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研发队伍。

显然，软件项目参与人数越多，则项目组成员之间的通信与协作越不便利。通常情况下，项目小组人数要求限制在 8 人以内，因此，当项目规模庞大而需要几十人参与时，就有必要将一个大项目分解成许多个小项目，然后建立多个有组织关联的项目小组实施产品研发。

常用的项目分组方法是，将系统按功能分解成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，由此形成的项目团队组织体系即为项目层级负责制。

图 3-4 所示为项目层级负责制的一般组织结构。

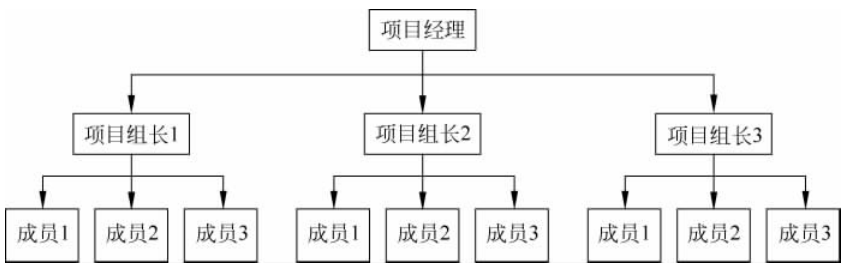

图 3-4　项目层级负责制组织结构

项目层级负责制体现出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，而项目组长又再将承担的小组任务做进一步分解，并分配给每个小组成员。

项目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任务协调。可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，各个项目小组则基于项目整体计划，制订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。

项目任务分派则还须考虑到任务之间较低的依赖度，如分配给各个项目小组的子系统任务有较好的独立性、研发的子系统可独立调试，由此可减轻因高度依赖而带来的任务协调负担。

3.2　软件项目计划

项目是一个工程容器，包含诸多工程要素，项目还是一个相对独立的工程作业单元，可使诸多工程任务基于工程项目得以实施。然而，要使基于项目的工程任务能够有序实施，则必须要有一个项目计划。软件项目要求事先制订计划，其作为项目任务行动指南，可使项目任务在计划约束下有序进行。

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，为确保这些任务按期完成，诸多任务必须合理安排到人，即任务分配。

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小组成员身上。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。图 3-5 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。

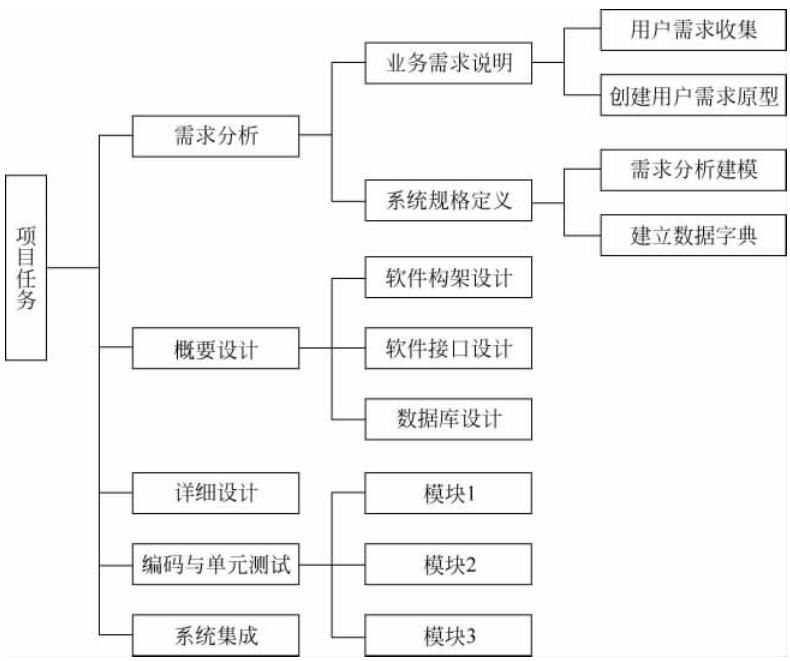

图 3-5 项目任务树

3.2.2 进度计划

项目计划的核心内容是项目进度计划。应该基于里程碑制订项目进度计划,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。而这个里程碑就是项目进行过程中的关键任务完成节点,它有具代表性的成果作为里程碑标志,如软件需求规格说明书、软件系统设计说明书,其用于表示已完成了该项任务,如产生了软件需求规格说明书,即表明已经完成了需求分析任务。

在制订项目进度计划时,需要对与里程碑直接关联的关键任务做重点部署,其对项目进程有至关重要的影响。

项目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。

项目进度计划制订中还须顾及到的是,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，在制订项目进度计划时，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。

通常情况下，一个基于里程碑的进度计划将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。

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(Gantt)

甘特图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，并能方便基于里程碑的项目管理。

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。

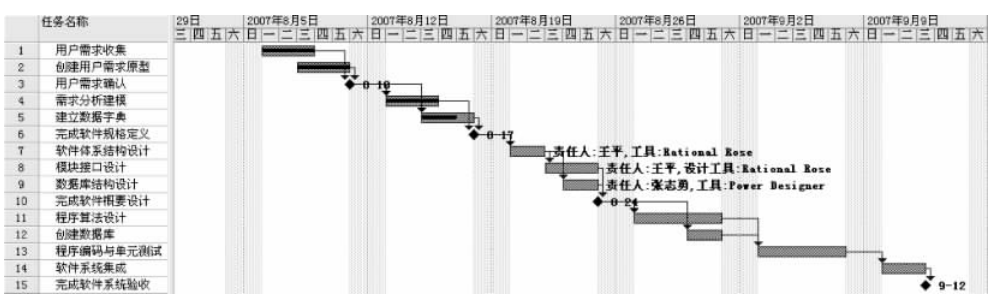

图 3-6　使用甘特图描述项目进度

2. 任务网络图(PERT)

任务网络图则基于项目任务创建，每个任务为一个任务节点，任务节点之间的网络连线可表示任务执行流程。

图 3-7 所示即为任务网络图，其使用矩形表示任务，网络连线为执行路线，箭头所指则为执行方向。

任务网络图直观性好，有利于描述并行任务。一些项目任务需要并行开展，还可能需要根据任务依赖关系、成员协作性等，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。

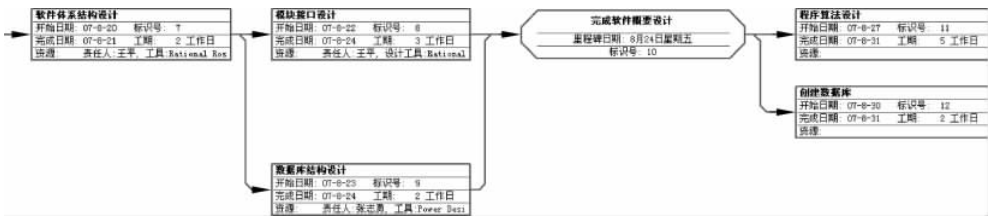

图 3-7 使用任务网络图描述项目进度

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,可看作关键路径。任务网络图可直观表现关键路径。例如图 3-7 中的模块接口设计与数据库设计这两项并行任务,由于模块接口设计耗时更多,因此为关键路径,被放在主线位置。

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,因此能够得到了更加明确的表达。

3.2.3 项目计划书

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
     - 3.3.1 程序(说明需移交用户的程序的名称、编程语言、存储形式等)
     - 3.3.2 文档(说明需移交给用户的文档的名称及内容要点)
   - 3.4 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)
   - 3.5 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)
   - 3.6 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订的其他专题计划,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 软件项目成本估算

软件开发涉及成本。通常情况下,在软件项目初期,如项目立项论证时,或制订项目开发计划时,需要对软件开发项目做成本估算,然后以估算的成本为依据,对项目以后的具体实施进行必要的成本控制。

3.3.1 软件成本估算策略

软件项目有来自多方面的成本,如人力成本、场地费、差旅费、设备费、资料费等,并且这些成本因素都有很大的不确定性。因此,对于一个大型软件项目,为使估算尽量可靠,往往要用到多种估算方法。下面是一些常用的估算方法。

- 成本建模:根据软件项目特征,用数学模型来预测项目成本。一般采用历史成本信息来建立估算模型,并通过这个模型预测研发工作量和项目成本。
- 专家判定:聘请一个或多个领域专家和软件开发技术人员,由他们分别对项目成本进行估计,并最后达成一致而获得最终的成本。
- 类比评估:根据以前类似项目的实际成本作为当前项目的估算依据。
- 自顶向下估算:依据软件功能以及实现该功能的子功能组成形式逐层分配成本。
- 自下而上估算:首先估计出每个单元的成本,然后累加得到最终的成本。

诸多估算方法都有它们的优势和不足。因此,在软件项目成本估算中,除了要从多种不同途径进行估算,还需要比较估算结果,如果采用不同方法估算的结果大相径庭,就说明没有收集到足够的成本信息,应该继续设法获取更多的成本信息,重新进行成本估算,直到几种方法估算的结果基本一致为止。

3.3.2 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,软件成本也就越高。

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的估计基础上。

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。

还需要考虑到的是,虽然可依据上面的方法进行软件代码行估算,但计算量必然很大。因此,有必要建立一个代码行估算程序,以减轻估算工作劳动强度。

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;
将软件功能项加入到功能表;
Do While 功能表中有没搜索到的功能项
    从功能表选择功能项 i;
    将功能项 i 分解为诸多子功能项;
    将子功能项加入到子功能表;
    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到子功能表;
            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,则根据软件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,而软件项目组成员平均每人每月能够开发 1 000 行代码,假如每人每月平均工资是 4 000 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. 代码行成本估算局限性

精确的代码行估算是建立在程序精细的功能分解基础上的。然而,一个很大规模的软件系统,要在项目早期即做出很精细的功能分解则并非易事,因此估算结论必然有很大的不确定性。

实际上,软件成本不能仅凭代码行数决定。来自设计者的质疑是,如仅凭代码行数决定成本,则因算法设计优良而只需较少代码即可实现的程序,就会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。显然,这是不合理的。

3.3.3 功能点成本估算

1979 年,Albrecht 最早提出了基于功能点的成本估算方法。这是一种建立在软件应用层上的、能够更好地适应项目初期的软件成本估算方法。

应该说,前面的代码行估算是一种很耗时间的成本估算方法。功能点估算则显得更具经济性,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出功能点数,就能由此获取到软件成本,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

复杂度 / 功能元素	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如下。

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如下。

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

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,可作出的判断包括:“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?	需要有良好的交互相应	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	
8	主数据文件联机更新?		需要	
9	数据输入、输出、查询、存储复杂度?	较低复杂度	一般复杂度	较高复杂度
10	数据内部处理复杂度?	较低复杂度	一般复杂度	较高复杂度
11	代码复用率?		需要考虑代码复用,以利于系统扩充改造	
12	系统须考虑平台转换?		需要	
13	系统须考虑多次安装?	可多次重复安装	可多次安装,并有多种安装配置模式	
14	系统须具有灵活性与易用性?		涉及多种数据输入与显示方式	多输入与显示方式,并有很好的交互提示

例如,前面软件问题中的功能数是 46。假如该软件系统中的基础数据需要备份(一般关联),需要进行远程身份验证(微弱关联),有一般的联机数据显示与更新需要,并需要有满

足一般安装功能的简单安装程序，以使软件可多次重复安装使用，则该软件的功能点数的计算如下。

功能点数 = 46 × [0.6 + 0.01 × (3 + 1 + 3 + 3)] = 33.2

4. 计算人力成本

如果要通过功能点数估算软件人力成本，则需要知道软件基于功能点数的生产率高低，即项目组成员人月均创建的功能点数与人均月工资数。

基于功能点的软件人力成本计算式如下。

软件人力成本 = (软件功能点数/人月均功能点创建数) × 人均月工资

例如，一个有 3 个成员的软件项目组，其每月能开发 90 个功能点，而该项目组人均月工资是 4 000 元，则开发出 33.2 功能点软件的人力成本如下。

软件人力成本 = (33.2/(90/3)) × 4 000 = 4 293(元)

实际成本估算时，还将涉及到程序实现语言的差异。一般地说，程序实现语言越是偏向高层应用，则实现一个功能点所需代码行数就越少，成本自然也就越低。

表 3-3 所列是一些常用程序设计语言的代码行与功能点的比值，其来自经验统计，因此并不精确，但可用来做成本评估参考。

表 3-3　常用程序设计语言的功能点与代码行的对应关系

程序设计语言	LOC/FP(平均值)
汇编语言	320
C 语言	128
VC++	64
Visual Basic	32
SQL	12

3.3.4　软件过程成本估算

软件还可以按照生产过程估算成本。软件的一般生产过程是：需求分析、概要设计、详细设计、编码实现。

生产过程中的每个阶段是一个任务单元，其主要人力成本因素包括时间长短、参与人数、参与者工资待遇等。需要注意的是，不同阶段会有不同的参与者。

通常情况下，前期阶段以高层人员为主，如系统分析员、构架设计师，他们有相对较高的工资待遇，但人员数量较少，时间也一般较短。后期阶段则以低层人员为主，他们有相对较低的工资支出，但人员数量较多，时间也一般较长。

下面是对这种成本估算方法的举例。

需要开发一个“企业资源管理系统”，并考虑按系统分析、结构设计、算法设计、程序编码、系统集成等几个步骤安排项目进程，并进行人员配置。

项目进度与人员配置如表 3-4 所示。

表 3-4 进度安排与人员配置

任务名称	参入人员结构(人)	完成周期(月)
系统分析	项目经理: 1 系统分析师: 1 构架设计师: 1 文档管理员: 1	1
结构设计	项目经理: 1 构架设计师: 1 高级程序员: 3 文档管理员: 1	1
算法设计	项目经理: 1 高级程序员: 3 程序员: 6 文档管理员: 1	1
程序编码	项目经理: 1 高级程序员: 3 程序员: 6 文档管理员: 1	2
系统集成	项目经理: 1 构架设计师: 1 高级程序员: 3 文档管理员: 1	1

除了分阶段进行项目成员配置外,还需要考虑各阶段项目成员的工资标准。表 3-5 所示为该项目成员工资标准。

表 3-5 工资标准

人员类别	月平均工资(元)
项目经理	6 000
系统分析师	6 000
构架设计师	6 000
高级程序员	4 000
程序员	3 000
文档管理员	2 000

各阶段任务的人力成本是这个阶段所有成员工资之和,而整个的软件人力成本则为各阶段人力成本之和。因此,可按以下表达式计算人力成本。

$$\text{人力成本} = \sum_{\text{阶段}}\left(\sum_{\text{职位}}(\text{职位人数} \times \text{职位工资标准})\right)$$

表 3-6 为开发“企业资源综合管理系统”时的各阶段成本,以及由各阶段成本累加而产生出来的总的人力成本。

表 3-6 人力成本

任务名称	阶段成本(元)
系统分析	20 000
体系结构设计	26 000
详细算法设计	38 000
编码	76 000
系统集成	26 000
人力总成本	186 000

3.4 软件项目风险

软件开发涉及诸多不确定性,如用户需求的不确定性、技术策略的不确定性等。这些不确定因素的存在,使得软件开发有了这样那样的风险,例如,人们或许在使用一种新技术,并可能为新技术的功能强大惊叹,然而在获得强大功能的同时人们也在冒技术风险,因为人们对新的技术缺乏良好的工程经验,因此有更大的出错概率。

值得注意的是,风险所影响的是项目的未来结果,人们只能判断其今后的发生概率,而并不能够百分之百地确定其影响。

3.4.1 风险类别

软件项目涉及多方面风险,如计划风险、管理风险、需求风险、技术风险、人员风险、产品风险、用户风险、商业风险,等等。

(1) 计划风险:由计划的不确定性所致,如列入计划预算被压缩,已列入计划的专用设备、技术资料不能及时到位,产品的计划交付日期被提前,等等。

(2) 管理风险:由管理的不确定性所致,如管理层意见可能不统一,管理制度可能不完善,管理行为可能不协调,等等。

(3) 需求风险:由需求的不确定性所致,如需求内容可能遗漏,规格说明可能有歧义,用户有需求变更,等等。

(4) 技术风险:由技术的不确定性所致,如技术可能不够先进,技术可能不够成熟,可能缺乏有效的技术支持,等等。

(5) 人员风险:由人员的不确定性所致,如项目成员可能缺乏较高职业素质,技术人员可能没有很好地掌握关键技术,技术骨干可能中途离职,新成员可能缺乏有效培训,等等。

(6) 产品风险:由产品的不确定性所致,如实际产品规模可能远远大于预计产品规模,产品可能不能获得预期的市场竞争力,等等。

(7) 用户风险:由用户的不确定性所致,如项目有可能难从用户那里获得预期的支持与理解。

(8) 市场风险:由市场的不确定性所致,如研制的软件产品可能不能获得预期的市场竞争力。

3.4.2 风险识别

风险调查是识别项目风险的有效途径。可以依据风险类别进行风险调查,以对项目风险有较全面的把握。下面是基于风险类别的相关调查提问。

1. 针对计划风险的提问

(1) 制订项目计划时是否对现实环境有较全面的考虑?
(2) 项目计划是否留有余地,并能有效应对项目计划的变更?

2. 针对管理风险的提问

(1) 管理团队是否由多方面专家组成,并有良好的集体协作?
(2) 开发机构是否有能很好适应软件生产的组织结构?
(3) 开发机构是否建立了较完善的项目管理规则?

3. 针对需求风险的提问

(1) 软件需求是否已被用户与开发者完全理解?
(2) 软件需求是否已经过用户与开发者双方共同确认?
(3) 需求规格定义是否已将用户需求完全包含进来?
(4) 用户对软件提出的要求是否现实?
(5) 软件需求是否稳定,是否会发生大的变更?

4. 针对技术风险的提问

(1) 项目所采用的技术是否能确保系统 5 年时间的有效应用?
(2) 项目所采用的技术是否已有成功的项目范例供参考?
(3) 项目组对所采用的技术是否有过成功应用的经验?

5. 针对人员风险的提问

(1) 项目组人员配备是否能够满足该系统的开发?
(2) 开发人员对项目将采用的技术是否有较好的把握?
(3) 项目组成员对工作是否有较高的满意度?

6. 针对产品风险的提问

(1) 待开发的系统的作用范围是否已有较清晰的边界定义?
(2) 项目组是否有承担如此大规模系统开发的经验?
(3) 项目组是否有与该系统相类似的其他系统开发的经验?

7. 针对用户风险的提问

(1) 用户机构高层管理者对项目是否能够给予积极支持?
(2) 软件最终操作者对有待创建的系统是否热情期盼?

8. 针对商业风险的提问

(1) 待开发的软件是否已有较全面的市场调研?
(2) 目前软件市场是否已有了同类型的软件产品?
(3) 是否有比同类产品更强的市场竞争力?

3.4.3　风险评估

风险评估需要考虑的是:风险有多大的发生概率?风险可带来多大的损害?

1. 评估风险概率

评估风险概率,即是确定风险事件发生的可能性有多大。

大多数情况下,人们只能凭据已有的项目管理经验对风险概率作出初步估计,然后再根据当前项目的实际情况,对这个经验数据加以修正。

例如,项目组成员流动风险。假设某个软件项目组有 10 个成员,但其中有一位女设计师最近结婚,并有生子休产假计划;另还有一位年轻设计师准备考研,有离职求学的可能。假如一般成员中途离职的概率是 0.1,但有生子计划的女设计师与有考研计划的年轻设计师是特例,因此需要将他们的离职概率修正为 0.6。

根据以上描述,可以对该项目组成员离职概率进行以下估算。

离职概率 = (8 × 0.1 + 2 × 0.7)/10 = 0.22

2. 评估风险影响力

假如某个风险事件的发生概率很低,则它尽管有发生的可能,但其影响力并不大。

然而风险影响力不能仅考虑风险概率,而是还需考虑风险事件一旦发生的危害程度。例如,飞机控制程序,尽管其出错的概率极低,但一旦发生则可能导致机毁人亡,因此必要高度关注。

一种来自于美国空军的项目风险评价体系是,将风险影响范围考虑为质量、成本、支持、进度这 4 个方面,而风险事件发生后的危害程度则考虑为灾难、严重、轻微、可忽略 4 个级别。表 3-7 是对这 4 个级别的详细说明。

表 3-7　风险评价体系

影响范围 / 危害程度	性　　能	支持	成本	进度
灾难	性能指标显著下降	不能修改、扩充	资金短缺	无法按期完成
严重	性能指标有一定程度的下降	较难修改、扩充	资金不足	可能延期
轻微	性能指标有较小程度的下降	较易修改、扩充	资金有保障	可按期完成
可忽略	性能指标没有下降	便于修改、扩充	可低于预算	将提前完成

风险影响力是风险概率与风险危害程度的乘积,即

风险影响力 = 风险概率 × 危害程度

例如：某软件系统被设计为由 100 个可复用组件构造，但其中的 30 个组件有特殊需求，并估计有 60%的可能需要专门定制，而定制一个组件的费用比使用已有组件的费用高 1 000 元，并需额外增加 3 个人日的工作时间。

则根据以上描述，可以计算出该软件系统组件复用时对成本、进度的风险影响力如下：

成本影响力 = 0.6 × (30 × 1 000) = 18 000 元

进度影响力 = 0.6 × (30 × 3) = 54 人日

3.4.4 风险防范

可以从风险规避、风险监控与风险应急这三个方面进行风险防范。

1. 风险规避

风险规避所考虑的是如何制定有效的风险预防措施以缓解风险的发生，包括降低风险发生概率、限制风险影响范围、减小风险危害程度。

例如，项目组成员流动，这是许多软件项目都存在的风险，可能带来的危害是使项目工作出现混乱，使项目技术外流，并可能使项目进度延期、项目成本提高、产品质量降低。因此，有必须采取合理措施以减少项目成员的流动，降低其危害性。

针对项目组成员流动风险，可以考虑的规避措施如下。

(1) 改善工作环境，建立公平合理的业绩评价体系，以提高成员对于当前工作的满意度与成就感。

(2) 增强情感交流，了解成员在工作、生活上遇到的困难，并能给予适当帮助，以使成员能够更加重视当前工作。

(3) 加强配置管理，以使项目工作及成果可追踪，防止项目技术被个人垄断。

(4) 重视新人培养，针对关键性技术工作还应建立有效的后备人员制度，以使得当某关键技术人员出现流动时，其工作能够快速方便地转移给后备人员。

2. 风险监控

风险监控是对风险规避的补充，以达到对项目风险更好的预防。

下面是针对项目组成员流动的风险监控。

(1) 项目组成员之间的任务合作与交流。

(2) 项目组成员的工作态度。

(3) 项目组成员对于工资、奖金的满意度。

(4) 项目文档的规范化程度与可读性。

(5) 项目配置管理工作质量。

3. 风险应急

风险应急所指的是，当风险最终成为现实后应该采取的对策。

通常情况下，有效的风险预防可降低风险影响力。

然而也有一些风险，无论之前的预防措施如何周密，只要其成为现实就必然带来很大的影响。例如，前面举例中的组件复用风险，尽管可通过改善软件规格、严格按标准设计应用

接口等措施提高组件复用率，但只要某个组件不能复用，则就必然带来的定制费用。

显然，对于这类风险，不得不有应急措施。

4. 风险计划

为实现对项目风险的有效管理，还必须制订风险计划。风险规避、风险监控与风险应急等应列入风险计划。

在制订项目计划同时，即可制订出风险计划。然而，如果项目风险很多很复杂，则有必要单独编制，以提高风险计划的清晰度。

大多数的风险是低概率的，并且有很低的影响力。实际上，风险计划一般只列入 20% 的高概率或高破坏性风险，以减轻风险计划编制负担，并使主要风险能得到高度重视。

可以参照表 3-8 样式制订风险计划。

表 3-8 风险计划表

风险名称	组件复用风险	类　别	技术风险
概率	60%	影响程度	严重
风险说明	因有特殊需求，可能有 30 个组件不能复用而需要专门定制，并可能使项目成本、进度受到影响 成本影响力＝0.6×(30×1 000)＝18 000 元 进度影响力＝0.6×(30×3)＝54 人日		
发生阶段	需求阶段、设计阶段		
预防措施	尽量基于标准组件库建立软件规格，并严格按标准设计应用接口		
监控措施	严格按组件复用标准进行需求与设计评审		
应急措施	按照该风险成本影响力，应预留 18 000 元作为风险应急资金 按照该风险进度影响力，应在项目进度计划中安排 54 人日的任务机动		

3.5 项目文档管理

软件文档是工程模式软件开发的成果体现。然而，软件文档要能产生很好的工程效应，则还必须要有管理规范的支持。

3.5.1 文档概念

软件文档是软件开发、维护与使用中需要依赖的资源，具有永久性，并可以由人或机器阅读。软件文档可体现出以下方面的用途：

(1) 软件开发的阶段性工作成果与里程碑标志。

(2) 提高软件开发过程的能见度。

(3) 提高软件开发效率。

(4) 可方便软件的使用与维护。

(5) 可方便潜在用户了解或选购软件。

3.5.2 文档分类

1. 按照软件文档形式分类

- 正式文档：必须建立的各种技术资料、管理资料，一般要有文档编码号，并需要进行专门的归档管理；
- 非正式文档：根据需要随时创建的并且无须归档的模型或工作表格。

2. 按照软件文档使用范围分类

- 技术文档：软件开发人员的技术性工作成果，如需求规格说明书、数据设计说明书、概要设计说明书、详细设计说明书；
- 管理文档：软件开发人员的工作计划或工作报告，如项目开发计划书、软件测试计划书、开发进度月报、项目总结报告；
- 用户文档：软件开发人员为用户准备的软件操作使用说明，如用户手册、操作手册、维护说明。

3. 按照国家标准分类

按照计算机软件产品开发文件编制指南的国家标准(GB 8567—88)的要求，在一项计算机软件的开发过程中，一般应产生以下若干种有关软件的文档资料。

(1) 可行性研究报告。
(2) 项目开发计划。
(3) 需求规格说明书。
(4) 测试计划。
(5) 概要设计说明书。
(6) 数据库设计说明书。
(7) 详细设计说明书。
(8) 模块开发卷宗。
(9) 用户操作手册。
(10) 系统维护手册。
(11) 测试分析报告。
(12) 系统试运行计划。
(13) 开发进度月报。
(14) 项目开发总结报告。

3.5.3 软件文档与软件生命周期之间的关系

可将软件生命周期划分为以下 6 个阶段：

(1) 项目论证阶段。
(2) 需求分析阶段。
(3) 软件设计阶段。

（4）软件实现阶段。

（5）软件测试阶段。

（6）软件运行与维护阶段。

不同阶段将会有不同的软件文档输出，具体对应关系如表 3-9 所示。

表 3-9　软件开发不同阶段需要产生的文档

阶段 / 文档	立项论证阶段	需求分析阶段	设计阶段	实现阶段	测试阶段
可行性研究报告	—				
项目开发计划	—	—			
需求规格说明书		—			
测试计划		—	—		
概要设计说明书			—		
数据库设计说明书			—		
详细设计说明书			—		
模块开发卷宗				—	—
用户操作手册	—	—	—	—	
系统维护手册			—	—	—
测试分析报告					—
开发进度月报	—	—	—	—	—
系统试运行计划					—
项目开发总结					—

3.5.4　文档的使用者

软件项目的管理者，软件的创建者、维护者，以及软件最终用户，都将需要使用软件文档。然而，他们分别承担着不同的职责，并因职责的不同而需要使用到不同的文档。

1. 管理人员

（1）可行性研究报告。

（2）项目开发计划。

（3）模块开发卷宗。

（4）开发进度月报。

（5）项目开发总结报告。

2. 开发人员

（1）可行性研究报告。

（2）项目开发计划。

（3）需求规格说明书。

（4）详细设计说明书。

（5）数据库设计说明书。

（6）测试计划。

(7) 测试分析报告。

3. 维护人员

(1) 设计说明书。
(2) 测试分析报告。
(3) 模块开发卷宗。

4. 最终用户

(1) 用户手册。
(2) 操作手册。
(3) 系统维护手册。

3.5.5 文档编码

为使文档能获得规范管理,所有正式的软件文档都有必要加编码号(用做文档的唯一标识),并需要有相应的文档编码规则与其配套。

文档编码要求能够体现出文档的基本特征,如开发机构标识、文件种类标识、项目标识、软件标识、创建年月、顺序号和版本号等。下面是一些常用的文档编码规则。

(1) 开发机构标识:由开发机构英文名称单词缩写或汉字名称拼音首字母等字符组成。如新时代软件技术有限公司,其标识即可为 XSD。

(2) 文件种类标识:由 1 到 2 个字符组成,用于说明文档归类。如 R—需求分析类;D—设计说明类;C—源代码类;M—用户文档类。

(3) 项目标识:由项目英文名称单词缩写或汉字名称拼音首字母等字符组成。如学校工作流平台研发,其标识即可为 XXGZLPTYF。

(4) 软件标识:由项目中需要开发的某软件的英文名称单词缩写或汉字名称拼音首字母等字符组成。如教务工作流系统,其标识即可为 JWGZLXT。

(5) 创建年月号:由 4 个数字组成,前两个数字年份,后两个数字为月份。如 0208,即表示是在 2002 年 8 月创建。

(6) 分类顺序号:由 4 位数字组成,用于表示所在文件类中的文件连续计数。

(7) 版本号:由数字组成,表示文件的更新次数。

例如,新时代软件技术有限公司针对学校工作流平台研发项目中的教务工作流系统,于 2007 年 6 月 18 日编写了“需求规格说明书”的第一版。则根据上述编码规则,可以得出该文档的编码号是:XSD-R-XXGZLPTYF-JWGZLXT-070618-0003-1。

需要注意的是,文档编码规范可能与开发机构历史有关,以致不同的开发机构可能会有不同的文档编码规范,但无论采取什么编码规范,目的都是为了更好地管理文档。

3.5.6 文档格式

通常情况下,正式的软件文档需要包含封面、目录、版本更新说明、文档内容等成分,如图 3-8 所示。

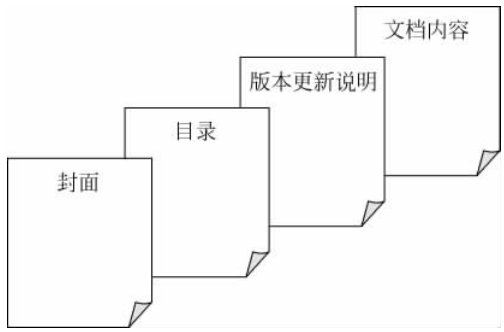

图 3-8　软件文档组成

1. 封面

文档封面需要体现的内容包括软件项目名称、文档名称、文档编码、保密级别、版本号、完成日期、作者姓名、软件机构名称等。

2. 目录

目录可采用手工编辑或是通过文档编辑软件自动生成。

3. 版本更新说明

当文档内容有更改时，如产生了文档的第二版、第三版时，需要加进版本更新说明，内容包括新版本号、旧版本号、更新日期、更新理由、责任人等。

4. 文档内容

文档内容主要成分如下。

(1) 标题

文档须按层次关系建立标题，以方便目录的编制。标题一般不使用标点符号，并按照科学编码法，从 1、1.1、1.1.1 开始编号。

通常情况下，一级标题使用四号加粗宋体，二级标题使用五号加粗宋体，三级标题使用五号普通宋体。例如，图 3-9 所示多级标题样式。

2. 可行性研究的前提

2.1　基本要求

2.1.1　基本功能要求

图 3-9　文档中的多级标题

(2) 正文

通常情况下，正文中的中文使用五号普通宋体，英文使用 Times New Roman 字体。

(3) 图表

图片不能跨页。表格可以跨页，但表格的第 2 页必须要有表头。图表需要编排顺序号，

一般按一级标题编排,例如,图 1-1、表 1-1。

(4) 引用文献

文档中需要引用文献时,需要使用方框号"[ ]"引用,例如,参看文献[6,7]。

(5) 术语

科技术语应采用国家标准(或行业标准)规定的(或通用的)术语或名称。对于新名词或特殊名词,需要在适当位置加以说明或注解。对于英文缩写词,应在文中第一次出现时用括号给出英文全文。注意文中名词术语的统一性。

(6) 参考文献

参考文献应按在文档中引用的先后次序排列。

• 著作文献编排格式如下:

序号 作者 书名 出版单位 出版年份 引用部分起止页

• 翻译文献编排格式如下:

序号 作者 书名 翻译者 出版单位 出版年份 引用部分起止页

(7) 附录

需要收录于文档,但又不适合写进正文中的内容,如附加资料、公式推导等。若有多个附录,可以采用序号,如附录 A、附录 B 加以区别。

(8) 索引

为了便于检索文中内容,可编制索引置于文中之后。索引以文档中的专业词语为关键字进行检索,指出其相关内容所在页码。中文词句索引按拼音排序,英文词句索引按英文字母排序。

(9) 页眉和页脚

一般页眉处标明一级标题名称,页脚处标明页码、日期,以保证单独的一页文档也能显示其正确属性。

3.6 项目配置管理

3.6.1 软件配置概念

所谓软件配置,也就是基于软件生产轨迹进行过程控制与产品追踪,其贯穿整个软件生命周期,可使软件开发中产生的各种成果具有一致性。

软件开发将会产生许多方面的成果,如软件需求规格说明书、软件设计说明书、软件测试计划、软件源程序清单,等等。通常,我们要求这些成果具有一致性。然而,软件开发时所涉及问题的复杂性与不确定性则可能给开发工作带来混乱,并可能因此带来成果内容的不一致。例如,软件需求变更,也许我们根据用户的变更请求修改了源程序,但需求规格说明与软件设计说明却遗漏了这种变更,于是源程序与需求说明、设计说明之间,就有了不一致的内容。

实际上,软件还必须考虑今后的维护,然而如果需要维护的软件的规格、设计与程序清单是不一致的,则今后的维护工作也必然会出现混乱。

软件配置管理可用来克服软件开发与维护时的混乱。

软件配置的主要任务包括软件配置规划、软件变更控制、软件版本控制。显然，这是一项非常专业的工作，对此许多软件开发机构设置了专门的软件配置部门，以提高配置管理工作的专业化程度，并使得软件开发中取得的成果能够集中管理。

软件项目组则通常配备有专门的配置管理员承担软件配置任务，其作为项目负责人的重要助手，进行软件开发任务的分配与软件成果的归档。

3.6.2 配置规划

1. 设置配置基线

软件开发过程被划分为几个阶段，每个阶段都将产生成果。配置基线是基于阶段任务定义的，并体现为某个阶段的所有成果的集合。

当需要对软件进行配置管理时，各个阶段中的每一项成果都要求作为配置项进行标识，基线则是阶段中所有软件配置项的集合。

设置配置基线是实现软件配置管理的基础，可使软件生产获得有效的监控与追踪。实际上，在基于里程碑的工程过程中，配置基线是一个清晰的里程碑标记，可使项目负责人清楚地知道是否到达了里程碑目标。

配置基线一般按照软件过程阶段进行设置，几个主要的软件阶段是需求分析阶段、软件设计阶段、软件实现阶段，与它们对应配置基线是需求基线、设计基线与产品基线。

(1) 需求基线中的配置项包括需求规格说明书、系统验收计划等。

(2) 设计基线中的配置项包括概要设计说明书、详细设计说明书、数据库设计说明书、系统集成计划、单元测试计划等。

(3) 产品基线中的配置项包括源程序文件、数据文件、数据库脚本、测试数据、可运行系统、使用说明书等。

2. 标识配置项

标识配置项即是给软件配置中的每个配置项一个唯一标记，以使它们能够被清楚地识别与追踪。

根据软件配置项的内容特征，其分为基本配置项和复合配置项两大类。例如：

某一份测试计划，某一组测试数据，某一个源程序文件，某一个数据文件，某一个数据表等，由于具有单一元素特征，因此可看作基本配置项。

某一个测试方案，某一个组件包或某一个子系统，则由许多个元素组成，因此被看作是复合配置项。

无论是基本配置项，或是复合配置项，都需要进行配置标识，以使其有唯一标记。配置项还一般需要通过一组信息加以细节说明，如名称、描述、资源、实现途径等。为使诸多配置项能够得到有效管理，通常还有必要建立分层目录组织它们，以便于表现复合配置项中的更小的软件配置项。

3. 建立配置库

软件配置管理需要建立 3 个配置库，即开发库、基线库与产品库。

(1) 开发库(Development Library)

开发库是一个面向开发人员的成果库,里面的成果一般是临时的,大多是有待进一步完善的半成品。

每个开发人员都应该有属于自己的开发库,以防止出现成果混乱。

当开发人员进入开发环境时,可以从开发库提取(Check out)开发资源进行系统创建;而当开发人员退出开发环境时,则可以把新建资源,或经过更新的资源,保存(Check in)到开发库中。

可通过开发库动态追踪开发人员的工作轨迹,或还原其以前的工作状态。实际上,由于有了开发库,开发过程中的软件变更会变得相对便于管理一些。

(2) 基线库(Baseline Library)

基线库是一个面向项目组的成果库,用来保存被确认的基线成果。通常情况下,如果开发库中的软件半成品,经过评审而确认达到了基线标准,就可从开发库移入基线库。

基线库的操作方式类似开发库,但这是一个受到严格变更控制的配置库,里面的配置项处于半冻结状态,只有项目组配置管理员或项目负责人具有操作权限,并需要经过严格评审才能发生变更。这也就是说讲,除非有十分充足的变更理由,否则基线库中内容是不允许改变的。

(3) 产品库(Product Library)

产品库是一个面向软件开发机构的成果库,用来保存最终产品。产品库的管理权一般属于软件机构中的配置管理部,只有该部门的工作人员才具有操作权限。

当软件开发任务全部完成之后,最终产品中的一切成果,如程序、数据、文档,都需要由基线库移入产品库。产品库要求处于完全冻结状态,里面的配置项原则上不允许变更。通常情况下,只有当软件需要进行错误修正或进行版本进化时,才允许进行变更。

3.6.3 软件变更控制

软件开发时可能会有变更发生,然而需要配置管理进行严格控制。如果不能有效地控制好变更,则势必会造成项目混乱,可能会给软件带来严重错误。

当软件成果还只是驻留在开发库时,开发者对软件中的错误修正可看作开发库授权者的自主行为,无须专门的变更审查。但是,当软件成果被移入基线库后,任何有关软件成果的变更,则都已不能是无约束的自主行为,而是必须受到专门的变更审查。

基线是用来体现阶段任务的。实际上,随着软件成果由开发库向基线库的一次完整移交的完成,则开发中的某个阶段的任务也即随之全部完成。我们所要求的是,项目能够按照基线实现成果的追踪,进行成果一致性控制。例如,我们在实现软件时发现了一个设计错误,然而对这个错误的修正则不能只考虑程序,而必须遵守变更规则,并依照变更规则将与这个修正有关联的所有成果(如规格说明、设计说明)都考虑进来,以确保成果的一致性。

软件变更控制一般会涉及以下两个方面的问题:其一,设置变更控制点;其二,确定变更步骤。其中的变更步骤是一种管理制度,主要有以下几个步骤。

(1) 提交书面的变更请求,详细说明变更的理由、变更方案、变更的影响范围等。

(2) 变更控制机构对变更请求作出较全面的评价,如变更的合理性、方案的技术价值与副作用、对其他配置项及整个系统的影响等。

(3) 评价结果以变更报告形式提交给变更控制负责人确认。

(4) 变更控制负责人执行变更。

上述步骤中,对变更的评价是最重要的一个环节。例如,来自用户的需求变更。如果经评价确认变更代价较小,并且不会影响到系统其他部分,则通常批准这个变更。但如果变更的代价比较高,或影响到了系统整体结构,则必须认真进行利弊分析,以确定是否同意进行这种变更。

3.6.4　软件版本控制

软件开发过程中会产生出许多版本。每一个版本可看成是有关程序、文档、数据的一次完整收集。

软件的多种版本的用途首先体现于软件的适应性上。通常,不同的操作系统或不同的用户类别,对软件会有不同的要求,因此需要有不同的软件版本适应。

软件版本还可用来控制软件进化。开发机构往往通过发布新的软件版本而使软件逐步完善——从测试版本到正式版本,再到升级版本。

当然,软件版本也需要进行配置管理,否则也可能会给软件带来混乱。通常,软件版本可通过版本号进行标识,并可利用版本控制工具进行管理。为对软件版本进行有效控制,其版本号要求按照一定的规则逐步升级,图 3-10 即反映了软件版本的有规则的升级,图中的每一个结点即代表着一个版本。

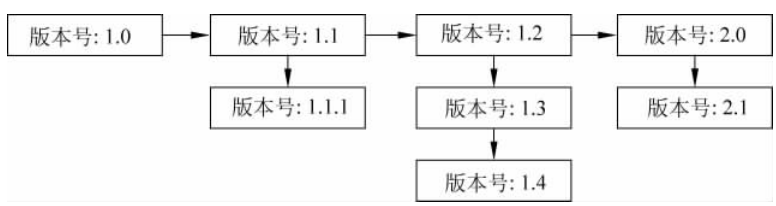

图 3-10　软件版本演变

3.7　项目质量管理

所谓软件质量,也就是对软件品质的优劣评价。很显然,软件开发者应该开发出高质量的软件产品,以更好地满足用户应用。然而,高质量的软件产品并不容易获得。

来自经验的结论是:严格有效的软件质量管理,可带来高质量的软件产品。

软件质量管理涉及问题包括质量标准、质量计划与质量控制。下面即从这 3 个方面介绍软件质量管理。

3.7.1　质量标准

1. 质量标准分类

软件质量标准是有关软件质量的纲领性规定,是建立有效的质量保证体系的基础,是评价软件质量好坏的基本依据。

根据制定质量标准机构的不同,可分为国际标准、国家标准、地区标准、行业标准,分别由国际机构、国家行政部门、地区行政部门或行业组织制定。例如,ISO9000 质量管理与质量保证标准,是由国际标准化组织(ISO)制定,为国际标准;而 GB/T10300 质量管理与质量保证标准,则由国家技术监督局参照 ISO9000 制定,为国家标准。

有人认为国际质量标准一定高于国家质量标准,国家质量标准又一定高于地方质量标准,而实际情形可能刚好相反。其情况往往是,国家质量标准基于国际质量标准建立,并还根据自己国家特定需要,加进了一些特别限制,因此,国家质量标准往往要高于国际质量标准。实际上,一些较大的软件开发机构也可根据自身特点,并参照国际标准、国家标准,制定出有自己企业特征的质量标准。显然,这样的质量标准有更高的质量要求。

软件质量标准涉及产品标准与过程标准两个方面的内容。

(1) 产品标准:用于定义被开发的软件成果需要遵守的标准,如文档标准、编码标准、接口标准。

(2) 过程标准:用于定义软件开发时必须遵循的工作流程与活动规则。

需要注意的是,软件产品必然需要基于一定的过程形成,因此,软件的产品标准将依赖于过程标准。通常,产品标准用于规定过程结果,而过程标准则用于规定获取产品的步骤与活动。

2. ISO9000 质量标准

国际标准化组织(ISO)于 1987 年发布了 ISO9000《质量管理和质量保证》系列标准。这是一套具有较强的指导性和实用性的针对产品生产过程做出质量保证的国际标准,在国际标准化组织的强力推动下,目前已有 50 多个国家和地区采用了这套标准,并制定了对应于该系列标准的国家标准。如欧洲共同体的 EN2900 标准,美国的 ANSI/ASQOQ90 标准,英国的 BS5750 标准,我国的 GB/T19000 系列标准。

ISO9000 系列标准共分为 5 个组成部分,即 ISO9000、ISO9001、ISO9002、ISO9003、ISO9004。

ISO9000 是该系列标准的选用指南,并为 ISO9001、ISO9002、ISO9003、ISO9004 的应用建立了准则。它主要阐述了质量概念、质量体系环境特点、质量体系国际标准分类,并对合同环境中质量体系国际标准的应用做出了说明。

ISO9001 是有关开发、设计、生产、安装和服务的质量保证标准。

ISO9002 是有关生产和安装的质量保证标准。

ISO9003 是有关最终检验和试验的质量保证标准。

ISO9004 是质量管理和质量体系要素的指南,是非合同环境中用于指导企业管理的标准,可为企业内部质量管理提供操作指南。

3.7.2 质量计划

为了使得所采用的质量标准能够在项目中得到有效贯彻,需要编制质量计划。

质量计划中需要说明将采用什么质量标准,需要明确规定什么样的软件产品才是符合质量要求的软件产品,并需要考虑将通过什么途径、措施、步骤使软件产品达到规定的质量要求。

通常情况下，可从以下方面考虑质量计划的编制。

(1) 过程质量，涉及内容包括制造过程质量规范、维护服务质量规范。

(2) 产品质量，涉及内容包括产品质量标准、产品质量属性(如系统最大出错概率、系统交互最大响应时限、系统最小内存要求)。

(3) 质量风险，涉及内容包括潜在的质量风险、风险应对措施。

3.7.3 质量保证

1. SQA 小组

为了保证软件质量，许多软件开发机构设立有 SQA 小组。这是一个专门负责软件质量保证任务的工作小组，其中的 SQA 是英文 Software Quality Assurance 的缩写。

SQA 小组要求由技术专家、领域专家、用户代表等诸多方面成员组成，他们将共同承担软件质量保证。

SQA 小组一般从以下方面进行质量保证。

(1) 制订软件质量保证计划。

(2) 按照质量计划中的过程质量标准对软件过程进行质量评审。

(3) 按照质量计划中的产品质量标准对软件产品进行质量评审。

(4) 记录软件过程与软件产品中存在的质量缺陷。

(5) 编写软件质量报告，并向负责软件质量的上级汇报软件质量。

2. 质量评审

质量评审是一种很传统的，但却非常有效的质量控制手段，能够实现对软件质量的全面的监督、检查与控制。实际上，尽管已经有了许多用于质量评价的软件工具，但这些工具大多只起辅助作用，一般只能针对某个局部问题进行质量评估，而对软件质量的整体评价，则仍依赖于质量评审。

质量评审有质量走查、质量审查两种形式。

(1) 质量走查

质量走查是一种非正式的有较大随意性的质量评审，可在软件开发的任何时候进行。也因此，质量走查往往被看作是项目小组内部的不定期自审。

质量走查一般是由项目负责人召集项目相关人员进行，内容涉及项目进展、文档规范化程度、源程序清晰度等。

(2) 质量审查

质量审查则是一种比较正式的质量评审，要求与里程碑过程管理结合，以方便基于里程碑进行质量控制。

质量审查一般由 SQA 小组负责实施。为使质量审查能够在较短时间里完成，SQA 小组可要求项目小组在进行正式的质量审查前，先进行走查式自审。

质量审查可按以下步骤进行：

① 综述。由需要评审的阶段成果的责任人向评审小组对该成果做概要说明，并在综述会议结束后把该成果详细报告分发给评审人员。

② 准备。评审人员仔细阅读需要评审的阶段成果的详细报告,然后列出在准备性审查中发现的错误,并按照错误发生频率对错误进行分级,以利于最常发生错误的区域能获取到更多的注意力。

③ 审查。评审小组按照成果的产生的路线、步骤,仔细检查成果的每个细节,并尽力找出成果中的所有错误。

④ 返工。由成果责任人负责改正在评审报告中列出的所有错误。

⑤ 跟踪。检查每个错误的返工修正,并确保没有引入新的错误。

3.7.4 质量指标

1. 软件可靠性

软件可靠性是很重要的软件质量指标。软件可靠性的定义是:在给定时间段内,程序按照规格要求无故障运行的概率。

显然,软件可靠性与软件运行时间长短有密切关系。一般地说,软件运行时间越长,则软件发生故障的概率也就越大,其可靠性也就越低。

软件可靠性是可进行测量的统计指标。例如,对某程序进行 100 次 24 小时无间断运行质量测评,若其中有 3 次运行中途发生了故障,则由该检测可得出该程序 24 小时无间断运行的可靠性是 97%。

2. 软件可用性

软件质量还可通过可用性进行评估。

软件可用性的定义是:在给定时间点上,程序按照规格要求无故障运行的概率。

软件可用性涉及平均无故障时间与平均故障修复时间两个可测量参数,并一般使用稳态可用性进行评价计算,其计算式如下:

稳态可用性 = 平均无故障时间/(平均无故障时间 + 平均故障修复时间) × 100 %

显然,软件平均无故障时间越长,而软件平均故障修复时间越短,则软件可用性也就越高。例如,对某程序进行质量测评,其在运行 100 小时后出现故障,经 1 小时修复后继续运行,并在运行 120 小时后又出现故障,并又经过 2 小时修复再次进入正常运行。

$$\text{平均无故障时间} = (100 + 120)/2 = 110 \text{ 小时}$$

$$\text{平均故障修复时间} = (1 + 2)/2 = 1.5 \text{ 小时}$$

$$\text{稳态可用性} = 110/(110 + 1.5) \times 100\% = 98.7\%$$

3.8 软件企业能力成熟度模型(CMM)

1987 年美国卡内基梅隆大学软件工程研究所推出了软件能力成熟度模型(Capability Maturity Model,CMM)。CMM 对软件组织在定义、实施、度量、控制和改善其软件过程的实践中各个发展阶段进行了说明,其成为了软件企业生产过程成熟度评估标准,并成为了改进软件企业软件工程过程与提高软件企业生产率及质量的依据。

3.8.1 能力成熟度等级

CMM将软件过程的成熟度分为5个等级，即初始级、可重复级、已定义级、已管理级、优化级，下面是这5个等级的特征说明。

(1) 初始级(initial)。工作无序，项目进行过程中常放弃当初的计划。管理无章法，缺乏健全的管理制度。开发项目成效不稳定，项目成功主要依靠项目负责人的经验和能力，但一旦项目负责人变更，则工作秩序面目全非。

(2) 可重复级(Repeatable)。管理制度化，建立了基本的管理制度和规程，管理工作有章可循。初步实现标准化，开发工作比较好地按标准实施。有了配置管理，变更依法进行，做到了基线化，稳定可跟踪，新项目的计划和管理基于过去的实践经验，具有了重复以前成功项目的环境和条件。

(3) 已定义级(Defined)。开发过程，包括技术工作和管理工作，均已实现标准化、文档化。建立了完善的培训制度和专家评审制度，全部技术活动和管理活动均可控制，对项目进行中的过程、岗位和职责均有共同的理解。

(4) 已管理级(Managed)。产品和过程已建立了定量的质量目标。开发活动中的生产率和质量是可量度的。已建立过程数据库。已实现项目产品和过程的控制。可预测过程和产品质量趋势，如预测偏差，实现及时纠正。

(5) 优化级(Optimizing)。可集中精力改进过程，采用新技术、新方法。拥有防止出现缺陷、识别薄弱环节以及加以改进的手段。可取得过程有效性的统计数据，并可以此为依据进行分析，从而得出最佳方法。

3.8.2 软件过程进化

CMM认为，软件工程过程是可进化的，任何软件企业只要能够持续努力去建立有效的软件工程过程，不断地进行软件过程管理的实践与改进，则其软件工程过程就可不断地走向成熟、趋于完善。

CMM为软件企业的过程能力的提高提供了一个基于成熟度等级的阶梯式改进框架，其中初始级是混沌的软件过程，可重复级是经过训练的软件过程，定义级是标准一致的软件过程，管理级是可预测的软件过程，优化级是能持续改善的软件过程。

任何软件组织所实施的软件过程，都可能在某一方面比较成熟，在另一方面不够成熟，但总体上必然属于这5个层次中的某一个层次。而在某个层次内部，也有成熟程度的区别。在CMM框架的不同层次中，需要解决带有不同层次特征的软件过程问题。因此，一个软件开发组织首先需要了解自己正处于哪一个层次，然后才能够对症下药地针对该层次的特殊要求解决相关问题，这样才能收到事半功倍的软件过程改善效果。

任何软件开发组织在致力于软件过程改善时，只能由所处的层次向紧邻的上一层次进化。而且在由某一成熟层次向上一更成熟层次进化时，在原有层次中的那些已经具备的能力还必须得到保持与发扬。

CMM过程改进框架是一个行动指南，它指明了一个软件组织在软件开发方面需要管理哪些主要工作，这些工作之间的关系，以及以怎样的先后次序一步一步地做好这些工作，

由此而使得软件组织逐步地从无定规的混沌过程向训练有素的成熟过程演进的途径。

3.8.3 个人软件过程(PSP)

CMM 提供了软件过程改进框架,使软件过程改进有了依据。但 CMM 没能提供过程改进的行为机制。为了弥补这个欠缺,卡内基梅隆大学软件工程研究所于 1995 年推出了个人软件过程(Personal Software Process,PSP)。

CMM 面向整个软件组织,侧重的是软件企业中有关软件过程的宏观管理。PSP 面向软件开发人员,侧重的软件企业中有关软件过程的微观优化。因此,CMM、PSP 二者互相支持,互相补充。也因此,为提升个人能力而进行的基于 PSP 的软件工程师培训,是提升软件企业的软件能力成熟度的重要内容之一。

PSP 提供了针对软件工程师个体的软件过程原则,被用来控制、管理和改进软件工程师的个人工程行为,以带来软件工程师个人工程行为的持续改进。

PSP 不受具体软件技术(如程序设计语言、软件工具或软件设计方法)局限,其原则能够应用到几乎任何的软件工程任务之中。

作为个人软件过程工程框架,PSP 涉及软件工程师个体软件工程活动的各个方面,其内容包括:

① 帮助软件工程师制订准确的个人过程计划;

② 确定软件工程师为改善产品质量要采取的步骤;

③ 建立度量个体软件过程改善的基准;

④ 确定过程的改变对软件工程师能力的影响。

3.8.4 团队软件过程(TSP)

CMM 应用还涉及到团队行为。也因此,卡内基梅隆大学软件工程研究所在 PSP 基础上,又于 1998 年推出了团队软件过程(Team Software Process,TSP),以对软件开发团队提供行为指导,以帮助软件开发团队改善其质量和生产率,使其更好地满足成本及进度目标。

1. TSP 自主团队

TSP 应用的首要问题是建立能够自我管理、自我完善的自主团队。基于 TSP 组建自主团队的原则如下:

(1) 遵循确定的、可重复的与迅速反馈的过程,以使团队培训达到最有成效。

(2) 团队目标、团队工作环境、技术指导人、行政领导人,这 4 方面的因素的综合决定了团队品质,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) 及时总结经验教训,当受训者在项目中面临到实际问题并寻求有效的解决方案时,就会更深刻地体会到 TSP 的价值。

(4) 借鉴前人经验进行团队过程改进,包括工程经验、科学原则、培训方法等,都可借鉴应用。

2. TSP 过程规则

可基于以下规则设定 TSP 过程如下：

(1) 循序渐进，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，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，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，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，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，对每一个项目的 TSP 规定明确的过程规范。

(7) 指令明确，对实施 TSP 中可能遇到的问题提供解决问题的指南。

3. TSP 项目特征

TSP 项目由一系列阶段组成，各阶段均由计划会议发起。

(1) 在首次计划中，TSP 组将制订项目整体规划和下阶段详细计划。TSP 组员在详细计划的指导下跟踪计划中各种活动的执行情况。

(2) 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地得到更新。通常无法制订超过 3 到 4 个月的详细计划。所以，TSP 根据项目情况，每 3 到 4 个月为一阶段，并在各阶段进行重建。

(3) 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，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。

(4) 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。

(5) 开发策略用于确保 TSP 的规则得到自始至终的维护。TSP 由诸多构成循环的阶段组成，并遵循交互性原则，以便每一阶段和循环都能在上一循环所获信息的基础上得以重新规划。

4. TSP 实施原则

在实施团队软件过程 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，应该遵循以下 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，开发小组成员如果发现过程不合适，应主动、及时地进行改进，以保证始终用高质量的过程来生产高质量的软件，任何消极埋怨或坐视等待的态度都是不对的。

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

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

小结

1. 软件研发团队

需要组建优秀的软件研发团队,以生产出高质量的软件产品。

软件研发机构应该有健康的可适应软件研发任务的组织机体。项目小组则是最小的因项目任务组建的研发团队,要求小而精,成员大多限制在 8 人以内。主要成员包括项目负责人、开发人员、资料管理员、软件测试员。

项目小组有多种管理机制,如民主分权制、主程序员负责制、职业项目经理负责制、层级负责制。

2. 软件项目计划

为使软件开发各项工作有秩序地进行,项目管理者必须事先制订项目开发计划。

- 任务分配:进行任务分解,然后合理地、适度地给每个成员分配项目任务;
- 进度安排:对项目任务及其资源按时序进行合理部署。可基于里程碑制订项目进度计划,一些关键性成果,如需求规格说明书,可作为项目进度里程碑标志。

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。项目计划书则是项目计划的具体体现,可作为软件开发的工作指南。

3. 软件项目成本估算

软件项目有来自多方面的成本,如工资开支、场地费、差旅费、设备费、资料费,但项目最主要成本是人员工资成本。软件成本估算,主要就是是对人力成本的估算。

常用的人力成本估算方法有: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。

4. 软件项目风险

软件开发涉及诸多不确定性,如用户需求的不确定性、技术策略的不确定性等。这些不确定因素的存在,使得软件开发有了这样那样的风险,如计划风险、管理风险、需求风险、技术风险、人员风险、产品风险、用户风险、商业风险等。

值得注意的是,风险所影响的是项目的未来结果,我们只能判断其今后的发生概率,而并不能够百分之百地确定其影响。因此,需要对风险实施有效的管理,以降低风险事件的发生概率。风险管理主要任务如下。

- 风险识别:调查是识别项目风险的有效途径。可以依据风险类别进行风险调查,以对项目风险有较全面的把握。
- 风险评估:风险有多大的发生概率?风险有多大的影响力?

• 风险防范：可以从风险规避、风险监控与风险应急这 3 个方面进行风险防范。

5. 软件文档管理

软件文档是工程模式软件开发的成果体现。然而，软件文档要能产生很好的工程效应，则还必须要有管理规范的支持。

开发时必须建立的技术资料、管理资料为正式文档，通常需要专门归档。而根据需要随时创建的并且无须归档的模型或工作表格则为非正式文档。

按照文档使用范围，则又可分为技术文档、管理文档与用户文档。

6. 软件配置管理

所谓软件配置，也就是基于软件生产轨迹进行过程控制与产品追踪，其贯穿于整个软件生命周期，因此可使软件开发中产生的各种成果具有一致性。

软件配置的主要任务包括软件配置规划、软件变更控制、软件版本控制。

7. 软件质量管理

所谓软件质量，也就是对软件品质的优劣评价。

软件开发者应该开发出高质量的软件产品，以更好地满足用户应用。来自经验的结论是：严格有效的软件质量管理，可带来高质量的软件产品。

软件质量管理涉及问题包括质量标准、质量计划与质量控制。

8. 软件企业能力成熟度模型(CMM)

能力成熟度等级：CMM 将软件过程的成熟度分为 5 个等级，即初始级、可重复级、已定义级、已管理级、优化级。

(1) 软件过程进化。软件工程过程是可进化的，任何软件企业只要能够持续努力去建立有效的软件工程过程，不断地进行软件过程管理的实践与改进，则其软件工程过程就可不断地走向成熟、趋于完善。

(2) 个人软件过程(PSP)。提供了针对软件工程师个体的软件过程原则，被用来控制、管理和改进软件工程师的个人工程行为，以带来软件工程师个人工程行为的持续改进。

(3) 团队软件过程(TSP)。对软件开发团队提供行为指导，以帮助软件开发团队改善其质量和生产率，使其更好地满足成本及进度目标。

习题

1. 软件研发机构内一般都设有质量控制部，并一般将其置于产品开发部、服务部之上。对此，读者有什么看法？

2. 通常认为，项目负责人不一定是技术专家，但必须是管理专家。对此，读者有什么看法？

3. 工作热情将直接影响工作效率。假如读者需要组建一个项目小组，以承担某项软件开发任务。那么，读者将如何管理这个项目小组，以使其有较高的工作热情？

4. 试比较民主分权制与主程序员负责制的优劣。如果由读者邀集几个要好同学一起承接某个软件项目,读者将采用哪种管理机制? 为什么?

5. 现需要开发一个学生管理系统,涉及以下功能,包括学籍管理、成绩管理、考绩管理、评优管理,并由一个 5 人小组承担这项开发任务,限期两个月内完成开发。对此,请读者使用任务树、甘特图、网络图等,对该项目作出较合理的任务及进度安排。

6. 需要使用 C 语言开发一个矩阵运算程序,涉及矩阵的加、减、乘等各种运算,估计有 30 个程序函数,每个函数平均约 80 行代码。如该程序安排 3 人合作创建,其每人每天平均能够完成 100 行代码,如每人的月平均工资是 3 000 元,每月按 20 个工作日计算。则完成该程序需要多长工期? 需要多少人力成本?

7. 需要使用 VC++开发一个设备监控程序,对它的初步估计是:一个监控参数设置窗(一般复杂度)、一个设备监控输出窗(高复杂度)、两个数据查询(中等复杂度)、一个外部设备接口(中等复杂度)。并有以下问题关联度判断。

序号	问　　题	关联程度(F_i)
1	备份与恢复?	0
2	数据通信?	5
3	分布式处理?	0
4	高性能要求?	5
5	高负荷操作环境?	3
6	联机输入输出数据?	4
7	多窗口切换?	3
8	主数据文件联机更新?	3
9	数据输入、输出、查询、存储复杂度?	3
10	数据内部处理复杂度?	3
11	代码要求高复用?	3
12	系统须考虑平台转换?	0
13	系统须考虑多次安装?	4
14	系统须考虑灵活性与易用性?	4

假如开发该软件的人员月平均工资是 3 125 元,每月 20 个工作日,且每人每天平均能够完成 100 行 C++程序代码,而一个功能点大约需要编 64 行 VC++代码。则开发该软件需要多少人力成本?

8. 软件项目为什么存在风险? 有哪些方面的风险? 如何管理项目风险?

9. 软件项目中主要有哪些技术文档? 有哪些管理文档? 有哪些用户文档?

10. 为什么软件开发需要有配置管理,其主要有哪些方面的管理?

11. 为什么配置管理中需要有开发库、基线库与产品库这 3 个配置库?

12. 什么是质量标准? 国际质量标准是否一定高于国家质量标准? 软件企业是否可制定自己的质量标准?