第 23 章 - 项目规划

- <编号>
- 第1讲

涵盖的主题

- <编号>
- 软件定价
- 计划驱动开发
- 项目调度
- 敏捷规划
- 估计技术

项目计划

- <编号>
- 项目规划涉及将工作分解为多个部分并将其分配给项目团队成员，预测可能出现的问题并准备针对这些问题的暂定解决方案。
- 在项目开始时创建的项目计划用于向项目团队和客户传达工作将如何完成，并帮助评估项目进度。

规划阶段

- <编号>
- 在提案阶段，当您竞标开发或提供软件系统的合同时。
- 在项目启动阶段，当您必须计划谁将负责该项目、如何将项目分解为增量、如何在整个公司中分配资源等时。
- 在整个项目期间，根据所获得的经验和监控工作进度的信息，定期修改您的计划。

提案规划

- <编号>
- 规划可能是必要的，只有概要的软件需求。
- 在此阶段进行规划的目的是向客户提供将用于为系统设定价格的信息。

23.1 软件定价

- <编号>
- 进行估算以发现开发人员生产软件系统的成本。
    - 您将硬件、软件、差旅、培训和工作成本考虑在内。
- 开发成本和向客户收取的价格之间没有简单的关系。
- 更广泛的组织、经济、政治和商业考虑会影响收取的价格。

影响软件定价的因素

- <编号>

因素	描述
市场机会	开发组织可能会报低价，因为它希望进入软件市场的新领域。接受一个项目的低利润可能会让组织有机会在以后获得更大的利润。获得的经验也可能有助于它开发新产品。
成本估算的不确定性	如果一个组织不确定其成本估算，它可能会通过超出其正常利润的意外事件提高其价格。
合同条款	客户可能愿意允许开发人员保留源代码的所有权并在其他项目中重用它。收取的价格可能低于将软件源代码移交给客户的价格。

影响软件定价的因素

- <编号>

因素	描述
需求波动	如果要求可能会发生变化，组织可能会降低价格以赢得合同。合同授予后，对要求的更改可能会收取高额费用。
财务健康	经济困难的开发商可能会降低价格以获得合同。获得低于正常利润或收支平衡比停业要好。在经济困难时期，现金流比利润更重要。

23.2 计划驱动发展

- <编号>
- 计划驱动或基于计划的开发是一种软件工程方法，其中详细规划了开发过程。
    - 计划驱动开发基于工程项目管理技术，是管理大型软件开发项目的“传统”方式。
- 创建一个项目计划，记录要完成的工作、谁来做、开发计划和工作产品。
- 管理人员使用该计划来支持项目决策并作为衡量进度的一种方式。

计划驱动的开发——利弊

- <编号>
- 支持计划驱动方法的论点是，早期计划允许密切考虑组织问题（人员的可用性、其他项目等），并且在项目开始之前发现潜在的问题和依赖关系，而不是一旦项目进行。
- 反对计划驱动开发的主要论点是，由于开发和使用软件的环境发生了变化，因此必须修改许多早期的决策。

23.2.1 项目计划

- <编号>
- 在计划驱动的开发项目中，项目计划列出了项目可用的资源、工作分解和执行工作的时间表。
- 计划部分
    - 介绍
    - 项目组织
    - 风险分析
    - 硬件和软件资源要求
    - 工作分解
    - 项目进度
    - 监测和报告机制

项目计划补充

- <编号>

计划	描述
质量计划	描述将在项目中使用的质量程序和标准。
验证计划	描述用于系统验证的方法、资源和时间表。
配置管理计划	描述要使用的配置管理程序和结构。
维修计划	预测维护要求、成本和工作量。
员工发展计划	描述如何培养项目团队成员的技能和经验。

23.2.2 规划过程

- <编号>
- 项目规划是一个迭代过程，当您在项目启动阶段创建初始项目计划时开始。
- 计划变更在所难免。
    - 随着项目期间有关系统和项目团队的更多信息可用，您应该定期修改计划以反映需求、进度和风险变化。
    - 业务目标的变化也会导致项目计划的变化。随着业务目标的变化，这可能会影响所有项目，然后可能需要重新规划。

项目规划过程

- <编号>
- **UML 活动图 - 一个典型的工作流**

23.3 项目调度

- <编号>

- 项目调度是决定如何将项目中的工作组织为单独的任务，以及何时以及如何执行这些任务的过程。
- 您估计完成每项任务所需的日历时间、所需的工作量以及谁将处理已确定的任务。
- 您还必须估计完成每项任务所需的资源，例如服务器上所需的磁盘空间、专用硬件（例如模拟器）所需的时间以及差旅预算。

项目调度活动

- <编号>

- 将项目拆分为任务，并估算完成每项任务所需的时间和资源。
- 同时组织任务以优化使用劳动力。
- 最小化任务依赖性，以避免因一项任务等待另一项任务完成而导致延迟。
- 取决于项目经理的直觉和经验。

里程碑和可交付成果

- <编号>

- 里程碑是计划中的点，您可以根据这些点评估进度，例如，系统移交以进行测试。
- 可交付成果是交付给客户的工作产品，例如系统的需求文档。

项目调度流程

- <编号>

调度问题

- <编号>

- 估计问题的难度以及开发解决方案的成本是很困难的。
- 生产力与完成一项任务的人数不成正比。
- 由于通信开销，将人员添加到延迟的项目会使其延迟。
- 意想不到的事情总会发生。在计划中始终允许偶然性。

附表表示

- <编号>

- 图形符号通常用于说明项目进度。
- 这些显示项目分解为任务。任务不能太小。他们应该需要大约一两个星期。
- 条形图是项目进度表最常用的表示形式。它们将时间表显示为活动或资源与时间的关系。

任务、持续时间和依赖项

- <编号>

任务	工作量（人-天）	持续时间（天）	依赖关系
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2、T4 (M3)
T6	10	5	T1、T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3、T6 (M5)
T10	20	15	T7、T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10、T11 (M8)

活动条形图

- <编号>
- **活动网络**
- 这个活动网络并不完全等同于最后一张幻灯片。

人员分配图

- <编号>
- 使用横过条的对角线的兼职作业

23.4 敏捷规划

- <编号>
- 软件开发的敏捷方法是迭代方法，其中软件以增量方式开发并交付给客户。
- 与计划驱动的方法不同，这些增量的功能不是预先计划的，而是在开发过程中决定的。
    - 在增量中包含哪些内容的决定取决于进度和客户的优先级。
- 客户的优先级和要求会发生变化，因此制定一个可以适应这些变化的灵活计划是有意义的。

敏捷规划阶段

- <编号>

- (1) 发布计划，提前几个月并决定应该包含在系统发布中的功能。
- (2) 迭代规划，具有更短期的前景，侧重于规划系统的下一个增量。对于团队来说，这通常是 2-4 周的工作。

XP中的规划

- <编号>

基于故事的规划

- <编号>

- XP 中的系统规范基于反映系统中应包含的功能的用户故事。
- 项目团队阅读并讨论这些故事，并按照他们认为实现故事所需的时间对它们进行排序。
- 发布计划涉及选择和改进故事，这些故事将反映要在系统发布中实现的功能以及故事的实现顺序。
- 选择要在每次迭代中实现的故事，故事的数量反映了交付迭代的时间（通常为 2 或 3 周）和团队的速度。

关键点

- <编号>

- 系统收取的价格不仅取决于其估计的开发成本；它可能会根据市场和组织的优先事项进行调整。
- 计划驱动的开发是围绕一个完整的项目计划组织的，该计划定义了项目活动、计划的工作、活动时间表以及每项活动的负责人。
- 项目调度涉及创建项目计划的一部分的图形表示。条形图显示活动持续时间和人员配备时间表，是最常用的时间表表示。
- XP 计划游戏涉及整个团队的项目计划。该计划是增量开发的，如果出现问题，则对其进行调整，以便减少软件功能而不是延迟增量的交付。

第 23 章 - 项目规划

- <编号>

- 第二讲

23.5 估计技术

- <编号>

- 组织需要进行软件工作和成本估算。有两种类型的技术可用于执行此操作：
    - *基于经验的技术*未来工作量需求的估计基于经理过去项目和应用领域的经验。从本质上讲，经理对工作量要求可能是什么做出了明智的判断。
    - *算法成本建模*在这种方法中，基于对产品属性（例如尺寸）和过程特征（例如所涉及员工的经验）的估计，使用公式方法来计算项目工作量。

基于经验的方法

- <编号>
- 基于经验的技术依赖于基于过去项目经验的判断以及在这些项目中花费在软件开发活动上的努力。
- 通常，您确定要在项目中生成的可交付成果以及要开发的不同软件组件或系统。
- 您将这些记录在电子表格中，单独估算它们并计算所需的总工作量。
- 让一组人参与工作量估算并请小组中的每个成员解释他们的估算通常会有所帮助。
- 基于经验的技术的困难在于新的软件项目可能与以前的项目没有太多共同之处。

23.5.1 算法成本建模

- <编号>
- 成本估计为产品、项目和过程属性的数学函数，其值由项目经理估计：
  - Effort = A·SizeB·M
  - A 是与组织相关的常数，大小可能是代码大小或功能/应用点，B 反映了大型项目的不成比例的工作（通常在 1 到 1.5 之间），而 M 是反映产品、过程和开发属性的乘数。
- 成本估算最常用的产品属性是代码大小（SLOC——源代码的行数是基本的大小度量）。
- 大多数模型都相似，但它们对 A、B 和 M 使用不同的值。

估计精度

- <编号>
- 一个软件系统的大小只有在它完成后才能准确地知道。
- 影响最终尺寸的几个因素
  - 使用 COTS 和组件；
  - 编程语言；
  - 系统分布。
- 随着开发过程的进行，尺寸估计会变得更加准确。
- 对 B 和 M 有贡献的因素的估计是主观的，根据估计者的判断而有所不同。

估计不确定性

- <编号>
- 在开发规划期间，随着项目的进展，估计会变得越来越准确。
- 如果所需工作量的初始估计是*x*个月的工作量，那么在交付系统时测量的实际工作量的范围可能是 0.25 *x*到 4 *x*。

23.5.2 COCOMO 2 模型

- <编号>
- COCOMO（Constructive cost model），一种基于项目经验的经验模型。
- 有据可查的非专有估计模型，与特定软件供应商无关。

- 从 1981 年发布的初始版本 (COCOMO-81) 到 COCOMO 2 的各种实例化的悠久历史。
- COCOMO 2 考虑了软件开发、重用等的不同方法。

- 现代方法，例如：
- 使用动态语言快速开发，
- 按组件组成开发，
- 使用数据库编程。

COCOMO 2 模型

- <编号>

- COCOMO 2 包含一系列子模型，可生成越来越详细的软件估计。
- COCOMO 2 中的子模型是：
    - 应用程序组合模型。当软件由现有部件组成时使用。
    - 早期设计模型。当需求可用但设计尚未开始时使用。
    - 重用模型。用于计算集成可重用组件的工作量。
    - 后架构模型。一旦设计了系统架构并且有关系统的更多信息可用时使用。

COCOMO 估计模型

- <编号>

(1) 应用组合模型

- <编号>

- 支持原型项目和有广泛重用的项目。
- 基于应用程序（对象）点数/月的开发人员生产力的标准估计。
- 考虑到 CASE 工具的使用。
- 公式是
    - PM = (NAP•(1 - %reuse/100)) / PROD
    - PM 是以人月为单位的工作量，NAP 是交付系统中的应用程序点总数，PROD 是生产力（如下一张幻灯片所示），'%reuse' 是对开发中重用代码量的估计.

应用点生产力

- <编号>

开发人员的经验和能力	非常低	低的	标称	高的	很高
ICASE 成熟度和能力	非常低	低的	标称	高的	很高
PROD (NAP/月)	4	7	13	25	50

- 应用程序组合模型产生一个近似估计，因为它没有考虑重用所涉及的额外工作。

(2) 早期设计模式

- <编号>
- 在同意要求后可以进行估算。
- 基于算法模型的标准公式
    - **PM** = A·SizeB·M 其中
    - **M** = PERS&RCPX&RUSE&PDIF&PREX&FCIL&SCED；
    - （基于七个项目和流程属性 - 下一张幻灯片）
    - **A** = 2.94 初始校准，
    - KSLOC中的**大小**（源代码的千行数），
    - **B 值**从 1.1 到 1.24 不等，具体取决于项目的新颖性、开发灵活性、风险管理方法和过程成熟度。
    - 这导致工作量计算如下：
    - **PM** = 2.94·尺寸(1.1~1.24)·M

乘数

- <编号>
- 乘数反映了开发人员的能力、非功能需求、对开发平台的熟悉程度等。
    - RCPX——产品的可靠性和复杂性；
    - RUSE - 所需的重用；
    - PDIF——平台难度；
    - PREX——人员经验；
    - PERS——人员能力；
    - SCED - 所需的时间表；
    - FCIL - 团队支持设施。

(3) 复用模型

- <编号>
- 考虑无需更改即可重用的黑盒代码以及必须进行调整以将其与新代码集成的代码。
- 有两个版本：
    - 不修改代码的黑盒重用。开发工作量为零。黑盒代码是无需理解代码或对其进行更改即可重用的代码。
    - 修改代码的白盒重用。计算相当于新源代码行数的大小估计。然后调整新代码的大小估计。

重用模型估计 1

- <编号>
- 对于生成的代码：
    - PM = (ASLOC * AT/100)/ATPROD
    - **ASLOC**是重用/生成代码的行数
    - **AT**是自动生成代码的百分比。

    - **ATPROD**是工程师在集成此代码时的生产力。
- Boehm, et al. (2000) 测得 ATPROD 每月大约有 2,400 个源语句。因此，如果一个系统中总共有 20,000 行重用的源代码，其中 30% 是自动生成的，那么集成生成的代码所需的工作量是：
        - (20000 * 30/100)/2400=2.5 人月 //生成的代码

重用模型估计 2

- <编号>
- 以下公式用于计算源代码的等效行数：
    - ESLOC = ASLOC * AAM。
    - ESLOC 是新源代码的等效行数。
    - ASLOC 是组件中必须更改的代码行数。
    - AAM 是一个适应调整乘数，它调整估计以反映重用代码所需的额外工作。简单地说，AAM 是三个组成部分的总和：
        - 表示对重用代码进行更改的成本的适配组件（称为 AAF）。
        - 一个理解组件（简称SU），表示理解要重用的代码的成本和工程师对代码的熟悉程度。
        - 代表重用决策成本的评估因子（简称为 AA）。AA 从 0 到 8 不等，具体取决于所需的分析工作量。
- 如果可以自动完成某些代码调整，则可以减少所需的工作量。因此，您可以通过估计自动适应代码 (AT) 的百分比并使用它来调整 ASLOC 来调整估计值。那么最终的公式是：
    - ESLOC = ASLOC * (1-AT/100) * AAM。
    - 计算 ESLOC 后，您可以应用标准估计公式来计算所需的总工作量，其中 Size 参数等于 ESLOC。然后将其添加到集成已计算的自动生成代码的工作中，从而计算所需的总工作量。

(4) 后架构级

- <编号>
- 使用与早期设计模型相同的公式，但有 17 个而不是 7 个相关的乘数。
- **PM** = A·尺寸B·M
- 代码大小估计为：
    - (1) 待开发的新代码行数（SLOC）；
    - (2) 使用重用模型（ESLOC）计算的新代码等效行数的估计；
    - (3) 对必须根据需求变化而修改的代码行数的估计。
    - 我们将这些参数的值相加，以计算您在工作量计算公式中使用的总代码大小（以 KSLOC 表示）。

指数项

- <编号>
- 指数项 (B) 取决于 5 个比例因子（以 0 到 5 的六分制评分）。为了计算 B，我们将评分相加，除以 100，然后将结果与 1.01(sum/100+1.01) 相加。

比例因子	解释
优先性	反映该组织以前在此类项目中的经验。非常低意味着以前没有经验；超高意味着组织完全熟悉这个应用领域。
开发灵活性	反映了开发过程的灵活性程度。非常低意味着使用规定的过程；超高意味着客户只设定一般目标。
架构/风险解决方案	反映了所进行的风险分析的程度。非常低意味着很少分析；超高意味着完整和彻底的风险分析。
团队凝聚力	反映了开发团队之间的相互了解和协同工作的程度。非常低意味着非常困难的交互；超高意味着一个整合的、高效的团队，没有沟通问题。
工艺成熟度	反映组织的过程成熟度。该值的计算取决于 CMM 成熟度问卷，但可以通过从 5 中减去 CMM 过程成熟度级别来实现估计。

- **后架构模型中指数计算中使用的比例因子**

- <编号>
- 例子：
- 一家公司在一个新领域开展了一个项目。客户没有定义要使用的流程，也没有时间进行风险分析。该公司拥有 CMM 2 级评级。
    - 先例——新项目（4）
    - 开发灵活性 - 无需客户参与 - 非常高 (1)
    - 架构/风险解决方案 - 无风险分析 - V. 低 .(5)
    - 团队凝聚力-新团队-名义（3）
    - 过程成熟度 - 一些控制 - 名义上 (3)
- 因此比例因子为 1.17，计算公式为：
- B=(4+1+5+3+3)/100+1.01=1.17
- **指数项**

乘数

- <编号>
- 产品属性
    - 关注正在开发的软件产品所需的特性。
- 电脑属性
    - 硬件平台对软件施加的约束。
- 人员属性
    - 将项目工作人员的经验和能力考虑在内的乘数。

- 项目属性
    - 关注软件开发项目的特殊性。

成本动因对工作量估算的影响

- <编号>

指数值	**1.17**	
系统规模（包括重用和需求波动的因素）	128,000 DSI	
没有成本动因的初始 COCOMO 估算	**730人月**	
可靠性	非常高，乘数 = 1.39	为关键成本动因分配最大值
复杂	非常高，乘数 = 1.3	
内存限制	高，乘数 = 1.21	
工具使用	低，乘数 = 1.12	
日程	加速，乘数 = 1.29	
调整后的 COCOMO 估计	**2,306 人月**	

成本动因对工作量估算的影响

- <编号>

指数值	**1.17**	
可靠性	非常低，乘数 = 0.75	为关键成本动因分配最小值
复杂	非常低，乘数 = 0.75	
内存限制	无，乘数 = 1	
工具使用	非常高，乘数 = 0.72	
日程	正常，乘数 = 1	
调整后的 COCOMO 估计	**295人月**	

- 我们可以看到，成本动因的高值导致工作量估计值是初始估计值 (730) 的三倍以上 (2306)，而低值将估计值降低到原始估计值的大约三分之一 (295)。
- 这凸显了不同类型项目之间的显着差异以及将经验从一个应用领域转移到另一个领域的困难。

23.5.3 项目工期和人员配备

- <编号>

- 除了工作量估算外，管理人员还必须估算完成项目所需的日历时间以及需要员工的时间。
- 可以使用 COCOMO 2 公式估算日历时间
    - TDEV = 3 · (PM)(0.33+0.2*(B-1.01))
    - PM 是 COCOMO 模型计算的工作量，B 是上面讨论的复杂性相关指数（对于早期原型模型，B 是 1）。此计算预测项目的名义进度 (TDEV)，忽略与项目进度相关的任何乘数。
- 所需时间与从事该项目的人数无关。

人员配备要求

- <编号>
- 不能简单地通过将总工作量除以所需的项目进度来计算所需的人员。
- 参与项目的人数因项目的阶段而异。
- 在项目上工作的人越多，通常需要的总努力就越多。
- 人员的快速聚集通常与日程延误有关。

关键点

- <编号>
- 软件的估算技术可能是基于经验的，管理人员判断所需的工作量，或算法，其中所需的工作量是从其他估计的项目参数计算出来的。
- COCOMO II 成本核算模型是一种算法成本模型，它使用项目、产品、硬件和人员属性以及产品规模和复杂性属性来推导出成本估算。