同济大学

TONGJI UNIVERSITY

毕业设计(论文)

课题名称	基于软件成本估算模型的DDP数据分发平台项目开发管理的研究与实现
副标题	
学　院	软件学院
专　业	软件工程
学　号	1652813
学生姓名	国琳
指导教师	黄杰
日　期	2020-06-05

装 订 线

基于软件成本估算模型的 DDP 数据分发平台开发管理的研究与实现

摘 要

随着全球新一轮科技革命和产业变革的持续深入，中国软件行业蓬勃发展，软件正以不可思议的速度融入到经济发展和社会进步的方方面面。正如习近平总书记在党的十九大报告中所指出，“要建立全面规范、标准科学、约束有力的预算制度”。软件成本估算是软件项目管理的一项关键工作，而项目管理的成功与否直接决定了项目效益/效果。近二十年来，国际上已发布了多个软件规模(成本)估算模型/标准，但是鉴于软件项目、软件过程及开发组织等的差异性，直接使用这些模型/标准得到的估算结果常常差强人意，这很不利于软件项目管理的计划制定及有限资源的合理分配，因此，在实践中探索一种适合本组织的、有效的软件项目规模度量与成本估算方法是十分必要的。

本文将以作者所在实习单位从事开发与管理的软件研发项目“DDP 数据分发平台”为背景，针对目前软件公司缺乏自主研发软件项目之软件规模度量与成本估算标准以及基准数据的问题，从研究软件规模度量和软件成本估算模型/标准着手，结合参与“DDP 数据分发平台”项目的实际开发与管理工作的数轮迭代，采用《软件开发成本度量规范》国家标准和软件开发成本估算模型 NESMA 功能点分析等标准，分析、估算及交叉验证该项目一期的开发规模及组织的生产率，将此结果作为本组织开发同类项目的基准(Benchmark)，用以估算项目二期规模，指导项目二期工期与进度及资源计划的编排。在项目二期开发与管理实践中，运用挣值分析法监督、控制计划值与实际值的偏差，修正已有基准，最终得到适用于本组织的准确的生产率基准，为本组织实现高效的项目管理提供依据。

关键词：软件规模估算，功能点法，组织生产率，项目管理，DDP 数据分发平台

Research and Implementation of Data Distribution Platform Project Development Management Based on Software Cost Estimation

ABSTRACT

With the continuous deepening of the new round of scientific and technological revolution and industrial transformation in the world, the Chinese software industry is booming, and software is being integrated into all aspects of economic development and social progress at an incredible speed. As General Secretary Xi Jinping pointed out in the report of the Nineteenth National Congress of the Communist Party of China, "We must establish a comprehensive and standardized budget system with scientific standards and strong constraints." Software cost estimation is a key task of software project management, and the success of project management directly determines the project benefits or effects. In the past two decades, many software scale (cost) estimation models / standards have been published internationally. However, given the differences in software projects, software processes, and development organizations, the estimation results obtained by directly using these models / standards are often unsatisfactory. This is not conducive to the plan of software project management and the rational allocation of limited resources. Therefore, in practice, it is necessary to explore an effective software project size measurement and cost estimation method suitable for the organization.

This paper will use the software development project "Data Distribution Platform" that the author is engaged in development and management in the internship unit as the background. To address the current problem that software company lacks the software scale measurement and cost estimation standards and benchmark data of independent research and development software projects, from researching software scale measurement and software cost estimation models / standards were started, combined with several iterations of actual development and management work involved in the " Data Distribution Platform" project, using the national standard of "Software Development Cost Measurement Specification" and the software development cost estimation model NESMA function points analysis and other standards, analyze, estimate and cross-validate the development scale of the first phase of the project and the productivity of the organization, and use this result as the benchmark for the development of similar projects by the organization (benchmark) to estimate the scale of the second phase of the project and guide the second phase of the project and schedule and resource planning. In the development and management practices of the second phase of the project, the earned value analysis method will be used to monitor and control

the deviation between the planned value and the actual value, and the existing benchmark was revised to finally obtain an accurate productivity benchmark applicable to the organization to achieve an efficient project for the organization management provides the basis.

Key words： Software size estimation, function point method, organizational productivity, project management, Data Distribution Platform

装

订

线

目 录

1 绪论

本章将聚焦于软件成本估算的定义、估算过程、研究意义及国内外现状，章末阐述本文的研究内容特色与结构。

1.1 研究分析软件成本估算技术的意义

随着全球新一轮科技革命和产业变革持续深入，国内经济发展方式加快转变，中国的软件行业发展已经迎来了更大的机遇，“软件定义世界”已不仅仅是一句口号，软件正在以不可思议的速度融入到经济发展和社会进步的方方面面。

习近平总书记在党的十九大报告中指出，“要建立全面规范透明、标准科学、约束有力的预算制度”。软件造价，一直作为任何一个软件项目预算申报、审查、招投标、项目结算中的核心问题，但是在这方面，始终没有一个很好的解决方案。工业和信息化部《软件和信息技术服务业发展规划（2016 年-2020 年）》中已明确指出，软件市场定价与软件价值不匹配的问题是我国目前软件和信息技术服务业发展依然面临的迫切需要解决的突出问题。

此外，软件成本估算一直是软件项目管理中的重要步骤之一，它的成功与否直接决定了软件开发的最终效果，是将经济学分析中的概念技术与软件项目管理紧密地结合起来。软件成本估算不同于传统的工业成本估算，主要是指软件开发过程中产生的工作量及相应代价的计算，如果没有一种比较精确度量软件成本的模型，以及对产品、环境、人力等因素的影响进行估算，那么软件的成本效益、盈亏分析以及决策过程就很难做出判断。

目前，随着软件系统规模和复杂程度的日渐扩大，从上世纪 60 年代开始，频繁出现以大批量软件项目进度延期、超支预算和质量存在缺陷为特征的软件危机。

麦肯锡公司调查结果[1]显示：源于错误软件项目计划而失败的项目中，66%的软件项目是因为实际资源超出预算所导致，34%是由于实施进度跟不上计划进度而导致。同样的，Standish 组织在对全球 50 000 余个项目进行调查后，发现只有 27~31%最后结果为“成功（Succeeded）”，即可以在规定的预算和时间内顺利完成；17~22%为“失败（Failed）”，即未能如期完成或因其他原因取消项目；49~56%为“被质疑的（Challenged）”，即虽然完成但是预算相较于前期估算超支了 66~81%。

虽然目前有些研究认为，在 CHAOS 报告中关于 89% 项目存在预算超支问题有些被夸大，实际的比例可能在 30%~40%。但根据对软件成本估算技术所带来的影响进行跟踪了解，发现软件成本估算仍然是软件项目成功与否的关键因素[2][3][4](如表 1.1 所示)。近年来的研究更是表明人们由于对软件成本估算技术的认知不足确实导致很大比例的软件项目失控，而这点是和需求不稳定问题并列的两个导致软件项目失败的重要原因。

表 1.1 2011~2015 年 CHAOS 数据库分析[2][3][4]

/	2011	2012	2013	2014	2015
Successful	29%	27%	31%	28%	29%
Challenged	49%	56%	50%	55%	52%
Failed	22%	17%	17%	17%	19%

如今，无论是学术界还是产业界，越来越多的学者和专家认识到，准确的软件成本估算是有效降低软件项目预算超支的重要手段之一，这不仅可以辅助资源配置、预算管理、进度管理的高效进行，同时可以直接有助于合理的投资竞标等商业决策，也对确定项目预算、规划进度、使公司管理者对软件开发进行有效监督起着重要作用。正如美国 Southern California 大学的 Boehm 教授所言："理解并控制软件成本估算带给我们的不仅仅是更多的软件，并且还会是更好的软件"[5]。

而目前国内的大多数项目仍然采取专家经验法或 WBS 进行成本估算，这对通过成本估算指导项目进行和过程改进远远不足。就目前看，软件成本估算和管理仍然是软件项目中比较薄弱的环节之一，大量软件项目正是由于成本管理不善，导致造价提高、质量难以保证，进而项目延期或失败。因此，准确的软件成本估算对于软件开发过程和项目管理流程控制尤为重要。

1.2 软件成本估算的定义和过程

软件成本估算主要是指软件开发过程中所花费的工作量及相应的代价[6]。不同于传统工业成本计算方法，软件成本估算不包含原材料和能源的消耗，其中主要的成本集中于人力成本，此外，由于软件并没有明显的制造过程。因此，软件成本估算过程应该聚焦于软件开发的过程，从软件计划制定、需求分析、概要设计、详细设计、开发编码、测试等一系列软件工程过程，即项目开发的流程所消耗的人力、物力成本进行度量。在软件开发成本估算过程要充分考虑所开发项目的工作量和进度及项目属性[7][8][9]。

通常，一个软件项目开发成本估算分为以下三个阶段[10]：

（1） 软件规模估算

度量软件规模主要采用两种策略：1，功能点法估算软件规模；2，源代码行数估算软件规模。

（2） 软件工作量及工作进度估算

在这个阶段，根据已掌握的软件规模信息估算项目工作量、项目成本及编排工作进度。主要方法有专家经验法、类比法和软件成本估算模型三种估算方法；

专家经验法即根据多位专家对于软件成本估算的经验和相关数据分析进行，取多位专家估算的平均值。

类比法通常作用于有历史数据的项目，通过与以往一个或多个项目进行类比估算，类似的历史项目作为基准，根据两者之间的差异进行数据调整，最后得出该项目的成本估算

结果。

软件成本估算模型法通常前期选取适用于此类项目的模型，根据相关公式计算软件成本，并且预测该项目所需的资源和进度，不过由于软件成本估算模型一般不是普适的，需要针对特定的开发环境或软件类型，所以没有一种估算模型可以适用于所有的开发环境和软件类型。

（3） 估算结果反馈

这是软件项目成本估算过程中一个非常重要的阶段，不仅包括针对估算方法本身，也包括估算实践过程的阶段性结果反馈。

1.3 软件开发成本构成

软件开发过程是指从软件项目立项之初到项目完成验收之间所涉及的需求设计、概要设计、开发编码、测试、验收交付等一系列的项目管理支持活动。

软件开发成本仅包含软件开发过程中的所有人力成本和非人力成本之和（见图 1.2），不包括例如数据迁移或者软件维护等成本。

人力成本包含直接人力成本和间接人力成本，非人力成本包含直接非人力成本和间接非人力成本。

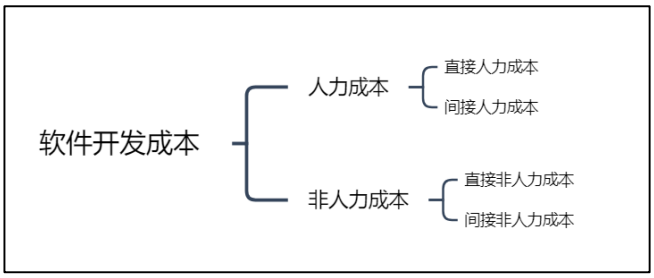

图 1.2 软件开发成本构成

1.4 国内外软件行业中成本估算现状

1.4.1 国外软件行业成本估算现状

（1）国外软件成本估算研究现状：

维也纳在 1967 年召开的第一届学术会议中正式提出项目管理的概念，同时国际管理协会第一次将项目管理归为一门学科，美国作为发达国家的佼佼者，在 1969 年专门设立了项目管理协会，针对项目管理工作开展了更加深入详实的研究，它的项目管理水平也一直在国际中处于领先的地位。1987 年，经过近 20 年的研究，美国项目管理协会正式推出了第一个项目管理知识体系，其后，1996 年和 2000 年两次对该体系进行了修改与订正。

1996 年，美国项目管理协会首次提出应将项目管理分成九个部分，其中，项目成本管理概念被重点描述。随后，项目成本管理知识被更多人逐渐认可。

项目成本管理也随后逐步形成了三个角度：项目全过程成本思想；项目全生命周期成本管理；项目全面成本管理。其中，项目全过程成本管理思想是由我国的学者和软件从业人员相继提出的。

随着有关学者和工作人员对项目生命周期成本管理的研究逐步深入，相关成本管理研究工作向好进行，但是实际过程中还是出现了各种问题，历史数据存量少成为了最大的难点，很多外国学者利用对历史数据的收集积累整理，将项目生命周期成本以不同种类进行分类。

20 世纪中期软件危机爆发，人们开始认识到软件项目成本估算和成本控制的重要性。当时的研究方向主要集中于软件的质量和软件开发人员的技术水平等。

（2）国外软件估算技术演变过程：

在国外，软件成本估算问题一直是软件领域的热点。经过了几十年的研究，现在已存在多种软件项目成本估算方法，下面是软件估算技术的演变过程：

20 世纪 60 年代，Nelsonk 首次对软件成本进行研究，通过建模分析软件项目成本。到了 80 年代，涌现了更加稳健准确的一批模型，如 Putnam 的 SLIM 模型，Park 的 PRICE-S 模型，Jones 的 Checkpoint 模型等[11]。1981 年，Barry Boehm 提出了 COCOMO 81 模型[12]，该模型主要基于参数估计原理，也被首先应用于 TRW 公司的 63 个项目，该模型适用于项目的前、中、后三个阶段，所以迅速在世界范围进行应用和推广。1995 年，Boehm 教授又对 COCOMO 81 模型进行改进，通过考虑和现代软件开发项目的实用性，开发出了 COCOMO II 模型[13]。2004 年，考虑到信息的模糊和不确定性，基于模糊逻辑，Zhiwei[14]等提出了新的软件成本估算模型。2008 年，Keung[15]通过消除启发式搜索的必要性，提出了 Analogy-X 模型，大幅提高了估算性能。2009 年，Li 等[16]克服了其他特征选择技术的局限性和复杂度，将案例推理技术应用到软件成本估算的特征选择。2015 年，Gharehehopogh 等[17]基于禁忌搜索和遗传算法的组合算法提出了优化软件项目的成本估算方法。

1.4.2 国内软件行业成本估算现状

国内从 20 世纪 90 年代开始逐渐重视软件成本估算，不过由于更加重视软件领域研发的研究，而忽略项目成本领域的研究。目前很多相关领域的书籍会提及一些估算方法，但是基本上还是阐述国外成本估算方法。2000~2010 年，周杰等人针对 Boehm 提出的 COCOMO 模型进行了研究；刘泽星等人通过使用两两对比法进行了软件规模估算；汤明端提出了通过扩展功能点法（EFP）评估软件项目规模；2015 年，吴登生[18]采用加权 CBR 模型提高了项目成本估算精度；2016 年，周启超[19]通过改进后的 BP 算法估算软件项目成本。

而近年来，国内很多金融机构也开始建立基于功能点的软件项目成本估算体系，军队和政府也成立了专门负责软件成本评估的部门。而各个软件开发组织或公司也开始将软件工程的思维方法应用于实践中，但是起步较晚、经验不足，很多只是刚刚开始进入规范管理阶段。对于国内的 IT 企业来讲，应用软件成本估算还存在很多困难：

（1） 认知不足

很多团队或者项目负责人认为估算仅仅是一个大概的范围，没有认识到估算对于软件项

目管理的重要性，并且很多可能还受限于商业范畴，例如只需要以合同周期或者资源情况进行软件的开发等。

（2） 资料缺乏

由于历史资料少，对于团队过去项目开发文档积累不规范或者不全面，难以用于软件成本估算。

（3） 辅助估算的工具较少

由于国外的各种估算方法较为复杂且种类繁多，掌握和实际应用都较为困难，国内企业的很多项目都为中小型项目，使用如此复杂的模型和方法会导致软件成本估算本身工作量增发，并且很少企业会针对此进行学习，多为依靠项目经验判断，导致不存在准确的量化标准支持所谓的经验判断。

（4） 估算方法不具备普适性

国外的各种估算方法大多基于欧美国家的软件企业项目实践，在建立之初，依据的是国外企业的历史数据，而国内外企业的工作习惯和生产效率差别较大，所以其中的各种参数其实并不完全适应国内的开发环境。

因此，研究适合于我国软件行业特点的本地软件成本估算势在必行，而本文也将聚焦于实习所在企业的实际项目情况，探索出适用的软件成本度量方法。

1.5 本文的研究内容及特色

本文案例 DDP 项目是作者实习亲身参加的项目之一。在该项目开发与管理过程中，作者亲历该项目管理的全过程，发现项目组在开展新项目时，确定项目成本多半采用经验法，在初步确定项目内容范围后，即拟定大概工期和成本估算。而在 DDP 一期项目进展过程中，由于资源预估不足以及工期分配不合理，导致项目延期一个半月上线。就目前了解调研的情况看，成本估算、项目过程管理对于本组来讲，是相对薄弱的环节，通过和 PM 沟通，发现对于研发过程中积累的一些软件成本估算方法，可以实现项目开始之初对项目可行性进行预估，但是却很难实现利用软件成本估算实现高效的项目流程管理控制。

总体来说，本组织并没有形成一套完备的成本估算流程体系，并且可以使用成熟的软件估算方法来提高项目成本估算的准确性。此外，软件成本管理的意识还比较薄弱，成本管理能力有待提高。无论是前期的工程量测算方法，软件成本预估结果，以及通过软件成本估算规划项目工期进行项目管理方面，均存在很大的误差及不准确性，这不利于项目组计划的拟定和资源的合理分配。

故希望将以作者所在实习单位从事开发与管理的软件研发项目“DDP 数据分发平台”为背景，针对目前软件公司缺乏自主研发软件项目之软件规模度量与成本估算标准及基准数据的问题，从研究软件规模度量和软件成本估算模型/标准着手，结合参与“DDP 数据分发平台”项目的实际开发与管理工作的数轮迭代，采用《软件开发成本度量规范》国家标准和软件开发成本估算模型 NESMA 功能点分析等标准，分析、估算及交叉验证该项目一期的开发规模及组织的生产率，将此结果作为本组织开发同类项目的基准(Benchmark)，用以估算

项目二期规模，指导项目二期工期与进度及资源计划的编排。

在项目二期开发与管理实践中，将运用挣值分析法监督、控制计划值与实际值的偏差，修正已有基准，最终得到适用于本组织的准确的生产率基准，为本组织实现高效的项目管理提供依据。

装 订 线

1.6 论文结构

本文共分九个章节，各章节内容安排如下：

第一章 项目背景。介绍软件成本估算技术的定义、基本方法，本课题的研究意义，国内外软件行业的成本估算现状，指出了我国目前软件行业在成本估算方面的不足，同时引出本文将解决的问题，阐述本文研究的主要工作和架构。

第二章 各类软件成本估算模型介绍和优缺点比较。区分基于算法模型和非基于算法模型的方法介绍了模型的演进和基本使用方法，比较得出本文适用的模型。

第三章 本文立足的真实项目——“DDP 数据分发平台”介绍。详细阐述了项目背景、业务架构和系统架构，最后展示系统上线后的效果图。

第四章 现有项目的软件成本估算方法及存在的问题阐述。详细描述了现有软件公司缺乏自主研发软件项目的软件规模度量与成本估算标准以及基准数据的问题这一亟待解决的问题和本文拟采用的解决方案。

第五章 基准的建立。采用软件开发成本估算模型 NESMA 功能点分析法，分析、估算本项目的开发规模及组织生产率，并将此结果作为本组织开发同类项目的基准。

第六章 基准的误差修订。通过《软件开发成本度量规范》国家标准[43]，对已建立的本项目的组织生产率（基准）计算中可能产生的误差进行修正，最后得到本组织开发同类项目的基准。

第七章 基准的应用。将上述计算得到的结果应用于 DDP 二期项目，用以估算项目二期规模，指导项目二期工期与进度及资源计划的编排。

第八章 挣值分析法修订基准。在项目二期开发与管理实践中，运用挣值分析法监督、控制计划值与实际值的偏差，修正已有基准，最终得到适用于本组织的准确的生产率基准，为本组织实现高效的项目管理提供了依据。此外，引用 DDP 项目经理资深应用系统分析师姚俊杰先生的评价，从而侧面反应该基准的应用效果。

第九章 总结本论文的研究成果及展望。并且提出一些在未来项目实际开发中仍值得改进的地方，同时阐述自己关于软件开发成本估算和成本估算模型的相关思考。

2 软件成本估算模型介绍和比较

本章将按照基于算法模型的估算法、非基于算法模型的估算法以及组合估算法的顺序依次介绍各类软件成本估算模型的演进过程及基本使用方法，通过比较其优缺点和局限性，选取适用于本文案例项目的估算模型。

2.1 基于算法的软件成本估算法

基于算法模型的软件项目成本估算法，提供了一个或多个算法形式，如乘法模型、线性模型、分析模型、复合模型以及表格模型等，将软件项目成本估算化为一系列成本驱动因子变量的函数[20]。该方法将成本估算关系和系统特征与工作量、进度的预估值联系起来。从参数得到成本估算的规则、公式。不同的模型不仅在成本因子关系的表达式上有所区别，并且在因子的选取上也大相径庭[21]。

基于算法模型的方法的基本思路[22]：找出软件工作量的各种影响因子，判定对工作量所产生影响的程度的类型（可加的、乘数的、指数的），得到最佳的模型算法表达形式。其中，当只影响局部时，一般是可加性的。例如，如果给系统增加功能点实体、源指令、模块、接口等，一般情况下只会对系统产生局部的影响。当某个因子对系统具有全局性影响时，则说是乘数的或指数性的，例如，增加服务需求的等级等。

基于算法模型的方法，可以分为三个主要发展阶段[22]：

(1) 早期阶段(1965 年～1985 年)

寻找好的方式、模型和因子关系。

(2) 中期阶段(1985 年～1995 年)

活动求精、规模定量和风险分析；。

(3) 后期阶段(1995 年～2005 年)

随着发展和研究的逐步进行，针对新软件开发风格进行了扩展。其中 COCOMO 不断发展和改进，成为最为典型的算法模型，频繁出现在对软件成本估算方法进行研究或分析的文献中[23][24][25][26][27]。同时，基于它的扩展所实现的工具也在对软件估算工具研究的主流之中[28]。

2.1.1 COCOMO 81 模型

COCOMO II 81 总共有 3 个等级的模型[20]：

1 基本模型：用于软件项目相关信息极少的情况；2 中等模型，应用于需求确定之后；3 详细模型，应用于设计完成之后。

以实际项目演进为例：

开发初始阶段，软件项目的相关信息较少，只要确定了软件项目的模式与可能具备的规模，即可以使用基本 COCOMO 81 模型完成初始估算（Effort:工作量）：

$$Effort = \mathrm{a} * (\mathrm{KDSI})^{\mathrm{b}} * \mathrm{F} \qquad (2.1)$$

其中，a，b 为系数，具体值取决于建模等级（基本、中等、详细）及项目模型（组织型、半独立型、嵌入型）

随着项目进展和需求确定不同阶段，下一步使用中等 COCOMO 81 进行估算。该模型定义了 15 个成本因子进行度量，详情见表 2.1:

在该模型中，每个成本驱动因子按照对软件项目成本的影响程度，得到了不同的工作量系数，即调整因子 F。使用中等 COCOMO 81 模型的估算工作量时，一旦确定软件项目的模式和因子取值，即可估算工作量。

表 2.1 中等 COCOMO 81 模型驱动因子即等级列表[20]

Cost drivers	Rating levels						Remark
	Very low	Low	Nominal	High	Very high	Extra high	
Product attributes							
RELY	0.75	0.88	1.00	1.15	1.40		Required software reliability
DATA		0.94	1.00	1.08	1.16		Database size
CPLX	0.70	0.85	1.00	1.15	1.30	1.65	Product complexity
Computer attributes							
TIME			1.00	1.11	1.30	1.66	Execution time constraints
STOR			1.00	1.06	1.21	1.56	Main storage constraints
VIRT		0.87	1.00	1.15	1.30		Virtual machine volatility
TURN		0.87	1.00	1.07	1.15		Computer turnaround time
Personnel attributes							
ACAP	1.46	1.19	1.00	0.86	0.71		Analyst capability
AEXP	1.29	1.13	1.00	0.91	0.82		Applications experience
PCAP	1.42	1.17	1.00	0.86	0.70		Programming capability
VEXP	1.21	1.10	1.00	0.90			Virtual machine experience
LEXP	1.14	1.07	1.00	0.95			Language experience
Process attributes							

续表 2.1

Cost drivers	Rating levels					Remark
MODP	1.24	1.10	1.00	0.91	0.82	Use of modern programming practices
TOOL	1.24	1.10	1.00	0.91	0.83	Use of software tools
SCED	1.23	1.08	1.00	1.04	1.10	Required development schedule

当被估算的软件各个模块确定后，就可以应用详细 COCOMO 81 模型计算，下表为详细 COCOMO 81 工作量不同的成熟阶段差异示例：

表 2.2 详细 COCOMO 81 模型工作量不同成熟阶段差异示例[20]

Cost drivers	Development phase	Rating levels					
		Very low	Low	Nominal	High	Very high	Extra high
AEXP	PRD	1.40	1.20	1.00	0.87	0.75	--
	DD	1.30	1.15	1.00	0.90	0.80	--
	CUT	1.25	1.10	1.00	0.92	0.85	--
	IT	1.25	1.10	1.00	0.92	0.85	--

当被估算软件的各个模块确定后，就可以使用详细 COCOMO 81 模型估算更加准确的成本范围。

详细 COCOMO 81 模型通过更加细粒度的因子影响分析、不同阶段区别的考虑，使估算人员更加细致地分析和把控项目，有助于更好地首先预算控制和高效地项目管理。

2.1.2 COCOMO II 模型

COCOMO 81 模型以及之后的 Ada COCOMO[29]，虽然在当时较好地适应了通过该类模型建模的一类软件项目，但随着软件技术的不断发展，不断涌现出大量的新模型和新方法，不仅没有好的软件成本估算模型与之匹配，甚至由于过程模型、产品模型、属性模型和商业模型等之间的冲突等问题，不断导致项目的预算超支与失败，COCOMO 81 模型也越来越不灵活，精准度也逐渐下降。

针对此类问题，Boehm 教授等在改进和发展 COCOMO 81 模型的基础上，1995 年首次提出了 COCOMO II 模型[30]。

COCOMO II 模型由三个子模型组成[30]：

（1）应用组合(Application Composition) 模型：

基于对象点(Object Point) ，通过采用集成计算机辅助软件工具快速应用开发的软件工作量和进度估算，大多用于项目规划阶段。

（2）早期设计(Early Design) 模型：

基于功能点(Function Point,简称 FP) 或总可用代码行，用于信息不足以支持细粒度估算阶段。

（3）后体系结构 (Post-architecture) 模型：

发生在软件体系结构被完好定义和建立后，基于功能点或源代码行，结合五个规模指数因子、17 个工作量乘数因子，用于完成项层设计和获取详细项目信息阶段。

COCOMO II 模型需要以此输入项目规模估算值，产品、开发过程、平台和所需人力共四个项目属性(成本因子)，关于复用、增量开发与维护的参数，组织历史项目数据，同时，该模型通过可以得到的开发、维护的成本以及进度估算，按阶段、活动、增量开发分布的成本和进度；

同时，该模型针对不同的组织，在使用时可用历史数据进行参数校准。在 COCOMO II 模型参数校准方面，1997 年，研究人员对预先专家决定的模型参数总计进行了 10%加权平均的多重回归调整，所用数据来自航空、商业、政府以及 NGO 等领域的 83 个实际项目[31]。1998 年，研究人员又利用贝叶斯分析（Bayesian analysis) 方法调整专家判定的模型参数，该方法使工作量估算以实际 30%范围变动，从 1997 年的 52%提高到了 71%[32][33]；2000 年，统计项目数增加到 161 个，而项目工作量估算以实际值 30%范围变动时，也进一步增加到 75%。贝叶斯分析方法可以说成功地把专家判定和回归分析技术有效结合起来，从而可以通过逻辑一致的样本数据和经过专家判定的经验数据，合理地确定该模型参数的分布。

2.1.3 NESMA 功能点分析（FPA）法

FPA 是 IBM A.J.Albrecht 在 1974-1979 年间，通过对大量项目生产率进行研究得到的结果。FPA 在 1979 年被首次提出，在 1983-1984 年间基于实践经验进行了调整。

FPA 引入了一个单位，即功能点，用来度量被开发或维护应用的规模大小。在 FPA 的框架中，“应用”一词意味着“自动化的信息系统”。功能点表示应用提供给用户的信息处理数量。这种度量单位与在技术意义上实现的信息处理方式不同。功能点是抽象术语，类似于计算房屋出租价格的“租赁点”，租赁点是基于拥有房间的数量、房屋的面积、房间设施的数量和房子的位置，这些作为住所的度量项提供给潜在租户。

FPA 最初被用于度量系统开发和应用建成后系统维护的生产率。由于 FPA 所需的数据在项目早期是可以获取的，所以它很快就被用于支持项目预算。

2.2 非基于算法的软件成本估算法

非基于算法模型的软件成本估算方法是相对于基于算法模型的方法而言，通常采取的是除数学算法以外的计算方法，来进行软件成本的估算，其中，比较典型的有类比估算、专家

估算和回归分析。

2.2.1 专家估算法

专家估算，包含从毫无辅助的经验直觉到依据历史数据、清单、过程指引等支持的专家专业判断[34]。总的来说，是一个比较宽泛的定义，它的主要判定标准为，软件项目成本估算工作是由该任务专家的人来牵头，而且成本估算过程的很大部分基于不可重复、不清晰的推理过程，即“直觉(Intuition)”。

对于某一专家自己所用的软件项目成本估算法而言，大家经常使用工作分解结构(WBS)。该方法通过将所有项目元素置于一定的等级划分，以此简化预算估量与过程控制的相关工作。

通常来讲，WBS 包括两个层次的分解：一是软件产品自身的划分，将软件系统分解为不同功能组件及组件下的各个子模块；二是根据开发软件所需工作的划分，分解为需求、设计、编码、测试、文档等大的模块以及之后更具体的细分，比如系统设计、详细设计、数据库设计等。WBS 法可以帮助估算者来确定到底哪些成本是需要估算的。此外，如果对每个元素的成本都可以设定一个相应的概率，那么就可以对整个软件开发过程的费用计算出一个自底向上的全面期望值[35]。

但是由于专家作为个体，可能存在着很多的个人偏好，因此，通常情况人们会更信赖由多个专家一起讨论得出的结果，为了达成小组一致，于是引入了 Delphi 方法：

首先，在每位专家不与其他专家讨论沟通的前提下，每位专家先对某个问题给出初步匿名评定。当对第一轮评定的结果收集、整理后，会返回给每位专家进行第二轮评定。此轮判定中，所有专家们依然面对同一评定对象，所不同的是，他们会得到第一轮总的匿名评定情况。第二轮的结果大多数情况可以将评定结论缩小为一个小的范围，从而得到一个合理的成本中间范围取值。所以，Delphi 法在 COCOMO 模型成本估算中被广泛使用[30]。

所以，当仅有的可用信息只能依赖所有专家意见而非确切的数据时，这种方法无疑是解决软件项目成本估算问题的最佳选择。并且，不同专家根据自己的经验可以对实际项目与经验项的差异做出更细致的发掘，有些甚至可以以此洞察未来新技术对成本估算可能带来的影响。然而，这种方法缺点也很明显，那就是不同专家的个人偏好、相关经验差异与所在专业局限性都可能为软件成本估算的准确性带来风险。

2.2.2 类比估算法

类比(Analogy)估算法是 CBR(Case-based reasoning) 法的一种形式[36]，即使用一个或多个已完成的同类项目与待估算的类似项目进行对比完成对当前项目的成本与进度的预测[37]。

类比估算法需要解决的主要问题为：

（1） 如何描述待估算项目实例特征。

即如何从历史相关项目众多特征中提取最具代表性的特征；

（2） 如何选取合适的相似度/差异度的表达式，以便评价相似程度。

(3) 如何使用相似的历史项目数据得到最终估算值。

其中，特征量的选取是决定哪些信息可用的具体问题，通常会征求相关专家意见，用以找出那些可以帮助确认出最相似实例的特征。而当选取的特征不全面时，所用的解决方法也是专家意见法[38]。

2.2.3 回归分析法

在对软件项目成本进行估算时，通常情况下，研究人员可以得到相关软件研究组织或者该类软件产品的某些历史数据。如果充分利用历史数据，对软件成本预测与估算是很有帮助的。

回归分析，就是一种常用且有效的数据驱动方法[39]，它包括 CART（分类回归树，Classification and Regression Trees），RR（稳健回归，Robust Regression），OSR（最优子集回归，Optimized Set Reduction），SOLS（普通最小二乘回归，Ordinary Least Squares Regression），Tepwise ANOVA（逐步方差分析，Stepwise Analysis of Variance）等。

其中，OLS 法是最传统的方法，它假定了将一个依赖变量和一个/多个独立变量互相关联的一种函数形式[40]。

2.3 软件成本估算模型优缺点比较

表 2.3 各类软件成本估算模型优缺点比较

方法名称	优点	缺点
COCOMO 81 模型	通过细粒度因子的影响分析、充分考虑了项目区别，更加细致地理解和掌控项目，有助于更好地控制预算和进行项目管理。	随着新模型和新方法的出现，商业模型、产品模型、过程模型、属性模型等产生的模型冲突，COCOMO 8 模型 1 可能出现不够灵活或者准确度不高的问题。
COCOMO II 模型	适用范围广、利于开发成自动化工具，有估算精确，易于使用的特点。	针对每个不同的环境都需要进行矫正。即使校验后，还存在大量的可变精度级别。
NESMA 功能点分析法（FPA）	从用户的角度来衡量工作量，可用于对软件项目进行规模评估。	部分系统级以及风险成本难以通过功能点方法来量化度量。
专家估算法	对历史数据要求低，适用于新研发的系统。	专家的水平、相关经验不一致，估算结果偏差较大。
类比估算法	通过参照历史项目，从中提取代表性的特征值，通过选取合适的相似度，进行成本估算。	筛选选取的特征时，通常采用咨询专家的方式，存在专家估算法的缺点。
回归分析法	虽然需要大量计算，但可用 Minitab，Splus，SPSS 等多种商业统计工具。	每一个观测值对于模型公式有同样程度的影响；需要大量的历史数据，至少是模型中参数数据量的 5 倍；

续表 2.3

方法名称	优点	缺点
回归分析法	/	需要满足软件工程数据严格的假设条件。
《软件开发成本度量规范》国家标准[43]	数据高可信度、计算简便、基准数据采集过程严格	估算精度可能不足、对于不同项目特征难以普适。

2.4 本章小结

本章根据对不同软件成本估算模型优缺点和局限性的比较，结合项目实际需求，将主要采用 NESMA 功能点分析（FPA）中的详细功能分析进行计算，同时使用《软件开发成本度量规范》国家标准[43]验证前期估算结果并确定误差范围。

装 订 线

3 DDP 数据分发平台项目背景及介绍

本章将阐述 DDP 数据分发平台的项目背景、业务架构及系统架构，章末将展示此系统上线后的界面效果图。

3.1 项目背景

该项目是作者实习单位于 2019 年 9 月启动的全新项目，全称为“DDP 数据分发平台系统”，分成一、二两期，最后将实现系统功能的实现和优化调整。

该项目与 11 个系统对接，日处理数据达 6000 余条，是实习单位众多平台系统间的“数据中间件”，汇集了公司全部组织人员信息，是各类人事数据到下游系统的中转通道，为下游系统提供标准化、安全可靠的 API。该项目最终不仅将完善日志功能，实现数据变更可追溯、接口调用可审计、Bug 出现可预警，也将降低接口耦合度和系统间调用的复杂性，提高数据中转的效率，提升数据的质量和安全性，大大降低运维成本，也为公司每个平台系统的数据准确、稳定运行提供有效的保障。

在该项目中，我全程参与 DDP 数据分发平台一二期的项目管理过程，同时参与部分项目前端开发工作（相关代码请见附录）。

故对整体的项目规划、管理均有较深参与和了解，本文也将立足于本项目，通过收集大量真实数据整理分析，希望可以探究出合适的本组织同类项目软件成本估算基准，从而用于项目管理流程控制、实现高效的项目管理。

3.2 DDP 数据分发平台业务架构

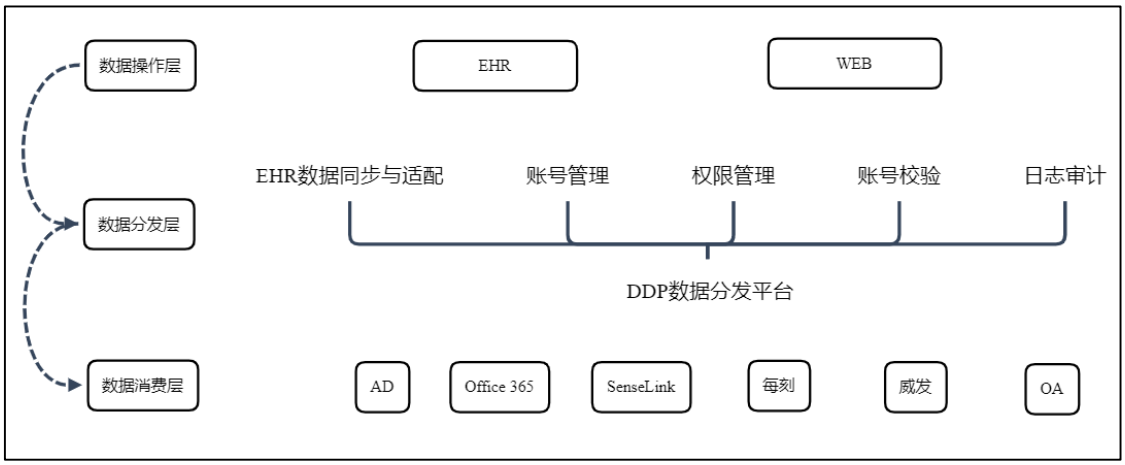

图 3.1 DDP 数据分发平台业务架构

3.3 DDP 数据分发平台系统架构

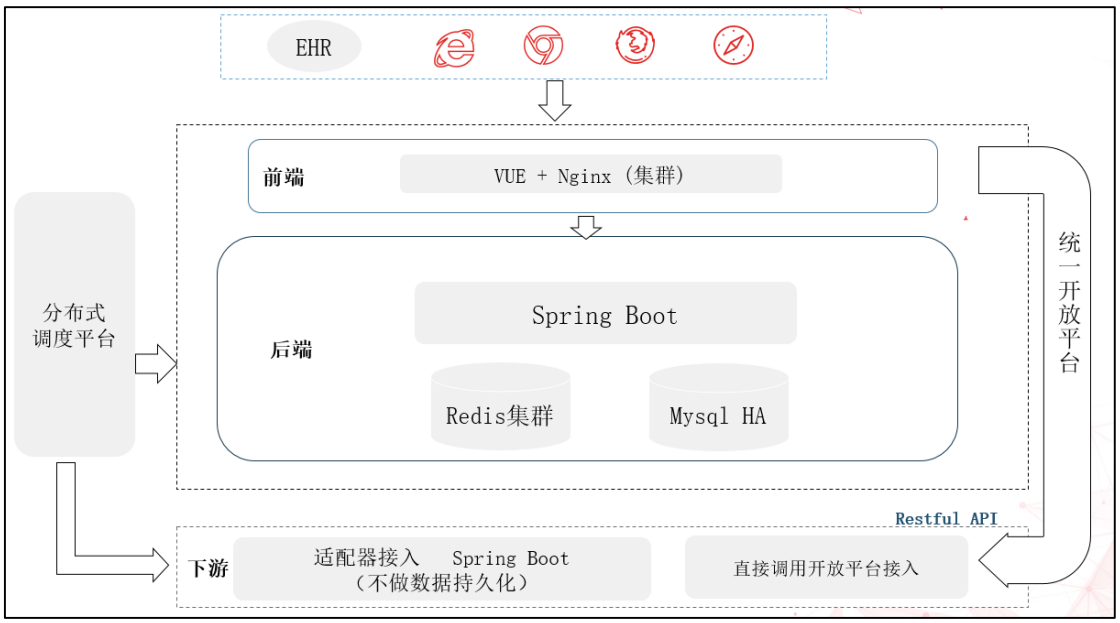

图 3.2 DDP 数据分发平台系统架构

3.4 DDP 系统上线后效果图

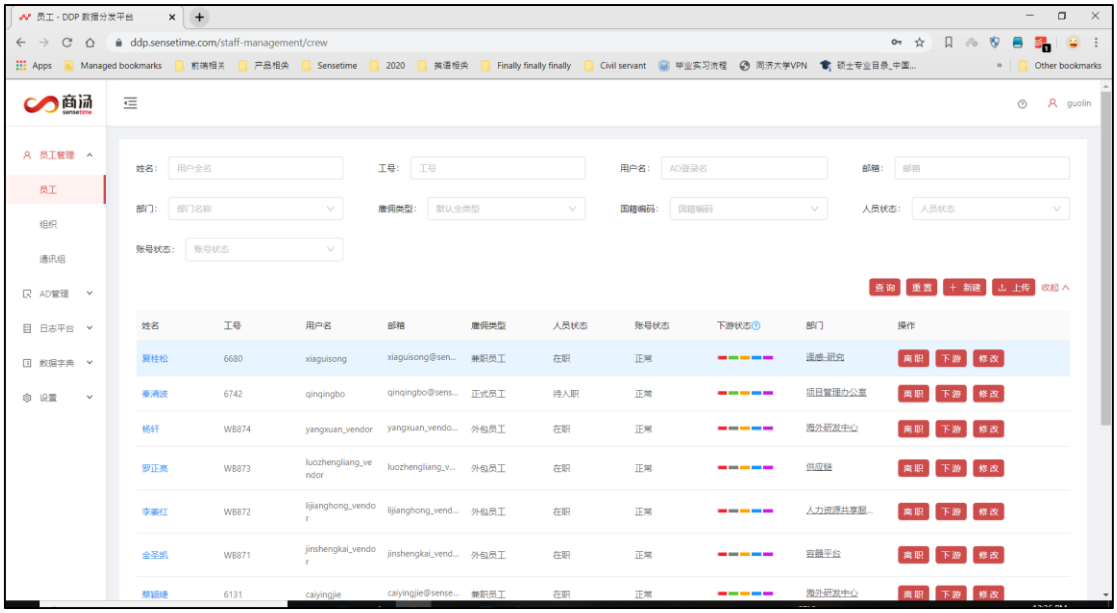

图 3.3 系统上线后效果图 – 用户信息界面

图 3.4 系统上线后效果图 – 用户详情界面

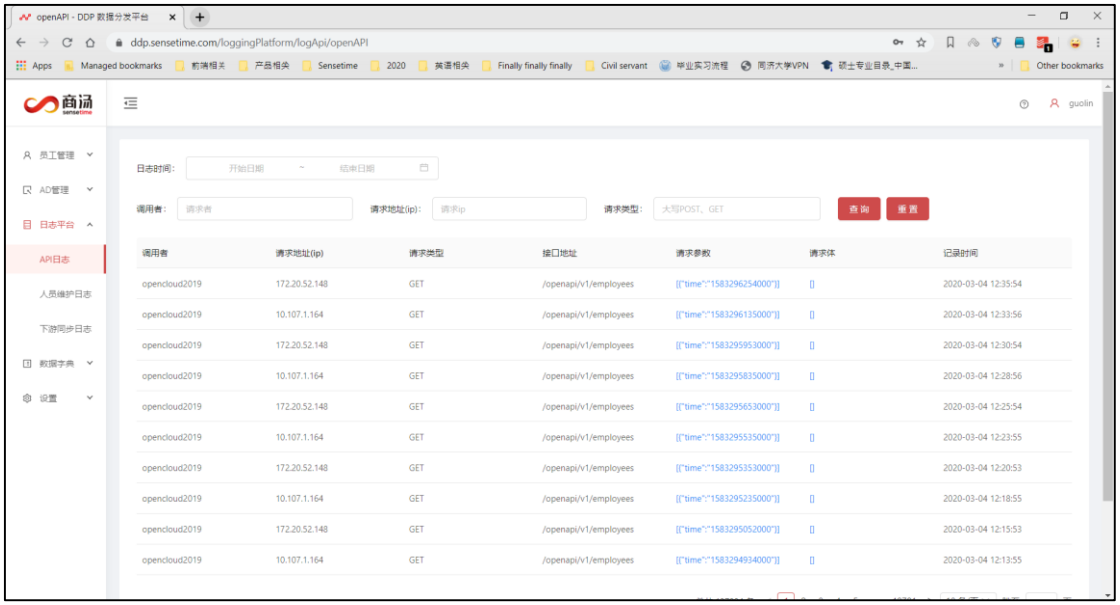

图 3.5 系统上线后效果图 – 日志信息界面

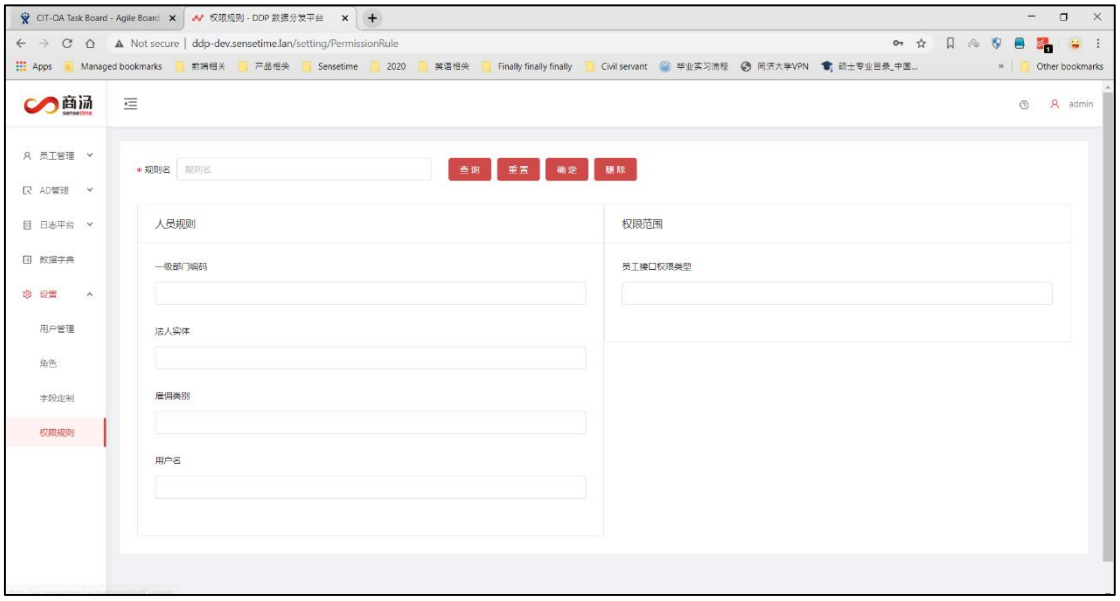

图 3.6 系统上线后效果图 – 权限规则界面

3.5 本章小结

本章主要介绍了作为软件成本估算基准建立对象的“DDP 数据分发平台”的背景和相关信息介绍，着重描述了该平台的业务结构、系统结构，章末展示了系统上线后的若干效果图。

4 现有软件成本估算方法及存在的问题

本章聚焦于现有软件成本估算方法及存在的问题，详细分析现有软件公司缺乏自主研发软件项目的软件规模度量与成本估算标准以及基准数据的问题，设计本文拟采用的解决方案。

4.1 现有软件成本估算方法现状介绍

4.1.1 组织架构情况

在此类项目的软件项目组中，主要人员配置如图 4.1 所示：

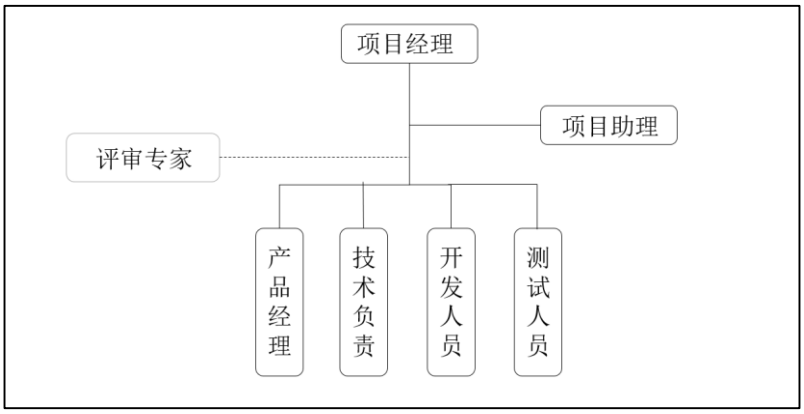

图 4.1 DDP 项目组织架构情况

项目管理人员（经理/助理）：项目的管理和组织者。

项目经理主要职责为根据业务实际需求，确定项目所需的全部资源，进行软件项目成本的估算，并且制定合理的项目计划。

项目助理负责协助项目经理开展相关工作，并为整体项目组提供服务，定期对软件项目的运行状况、资源管理状况进行分析汇总报告；

产品经理：主要职责为产品业务调研整理，并且负责用户与研发人员之间的沟通。当产品性能、功能、开发进度、成本等需求发生变化时，负责提出项目变更需求：而对公司提出的项目相关变更需求需进行判定并确认；

技术负责：主要职责为根据业务需求，对所需资源进行初步确认，同时负责软件项目的总体技术方案和技术架构；

开发人员：负责开展项目的具体研发工作；

测试人员：负责开展项目的各类测试工作；

评审专家：主要职责为对项目采用的技术、相关方案和资源估算等把关，提出发现的问题，并且审核问题的纠正措施。

其中评审专家并不作为项目组的成员，但是从项目立项开始到进行过程，一直负责参与项目的监督管理和协助工作，是重要的角色之一。

在项目组成员中，软件项目成本估算工作主要由 PM 负责，其估算过程将充分考虑开发、产品、测试等相关人员的建议，估算结果经评审无误后，上报领导进行。

4.1.2 现有项目成本估算情况

现有软件成本估算流程主要为：

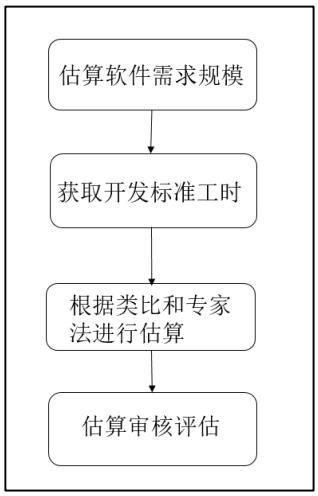

图 4.2 现有软件项目成本估算流程

（1）估算软件需求规模

在项目开始之初，产品经理对于产品的所有的功能需求进行分析，编制该项目业务需求规格说明书。

（2）获取开发标准工时

根据不同项目类别（如新开发项目、重构项目等）提供的各类标准工时，并参照相关《软件开发成本度量规范》[43] 国家标准。

（3）项目成本估算

项目经理对软件项目的成本进行估算。在软件项目成本估算过程中，主要使用类比法，根据提供的标准工时，对比以往相似项目的开发情况、目前部门的项目负载等因素，估算项目工作量。在估算过程中主要应考虑实际项目的难易程度、项目可复用程度以及技术人员的开发水平三方面因素。

（4）估算审核和评估

审核由评审专家组成委员会进行；评估主要由项目助理全程跟踪软件项目开发周期，记录估算的结果以及实际工作量，进行比较分析。

4.2 现有软件成本估算存在的问题

就目前调研情况看，对于本部门，软件项目成本估算、项目管理流程控制，是相对薄弱的环节，通过和项目经理的沟通，同时也发现在研发过程中积累的一些软件项目成本估算方法，可以实现项目之初对项目可行性及项目成本进行粗略的估计，但是却很难实现利用软件成本估算高效地完成项目流程管理控制。

总体来说，现有软件研发体系并没有形成一套完备的项目成本估算流程，并且实现使用成熟的软件估算方法提高项目成本估算的准确性，更好地辅助决策。此外，对于软件成本管理的意识还比较薄弱，重视程度不足，成本管理能力有待提高，故主要归纳为以下问题：

（1）估算精度不够，无法对决策起到明确的辅助作用

通过对项目的历史数据分析，目前依照经验判断的软件成本估算方法及流程，若对软件项目成本预估，估算结果大部分偏差较大，导致项目存在严重的预算超支问题或项目延期问题。以 DDP 项目为例，根据原有资源和方法估算，原计划一个半月之内争取上线，但因为成本估算及资源估算不准确，导致延期一个月上线。同时因为估算结果不精确，导致无法为管理层在进行产品定价、软件外包等市场决策时做出有效的帮助。

（2）估算方法单一

在现有的软件成本估算方法中，本项目组经常使用专家估算法和类比法进行软件项目成本估算。

对于迭代项目，专家估算法为参照历史项目的数据，先确定该项目的标准工时，之后项目经理进行软件成本的估算；对于变更类项目，采用类比法与同类项目对比，偏差一般较小；但是对于新研发项目，因为无同类项目参考，进行估算时，偏差一般较大，同时采用专家估算法确定的标准工时，因为主观性过强，难以被用户接受。

（3）人员主观影响因素大

在软件成本估算的过程中，采用专家估算法以及类比法这两种估算方法，估算人员的主观因素在整个软件估算过程中的影响往往很大，极大地依赖人员对软件项目估算过程的理解，对软件项目成本估算专业知识的掌握程度等都将对软件项目的估算结果产生很大的影响。

（4）估算方法无法适应环境的改变

在软件项目完成时，通常作为项目助理会将估算结果与实际工作量和进度进行比对，当预估结果与实际数据偏差较大时，及时地查找偏差产生的原因，并通过查找问题进行方法改进。

而在软件项目成本估算过程中，一旦出现了外在资源或开发环境、相关人员技能等因素改变，这种方式并不能及时自适应调整，这些将导致成本估算产生偏差和错误。

4.3 本章小结

本章探讨了本组织项目开发管理过程中实际暴露的问题，即无法利用软件成本估算完成高效地的项目管理。

后文提出了相应的解决思路，即从研究软件规模度量和软件成本估算模型/标准着手，结合参与“DDP 数据分发平台”项目的实际开发与管理工作的数轮迭代，采用《软件开发成本度量规范》国家标准和软件开发成本估算模型 NESMA 功能点分析等标准，分析、估算及交叉验证该项目一期的开发规模及组织的生产率，将此结果作为本组织开发同类项目的基准(Benchmark)，用以估算项目二期规模，指导项目二期工期与进度及资源计划的编排。

在项目二期开发与管理实践中，运用挣值分析法监督、控制计划值与实际值的偏差，修正已有基准，最终得到适用于本组织的准确的生产率基准，为本组织实现高效的项目管理提供依据。

5 基准的建立

本章将采用软件开发成本估算模型 NESMA 功能点分析标准，分析、估算本项目的开发规模及组织生产率，并将此结果作为本组织开发同类项目的基准。

5.1 软件开发成本估算基本流程

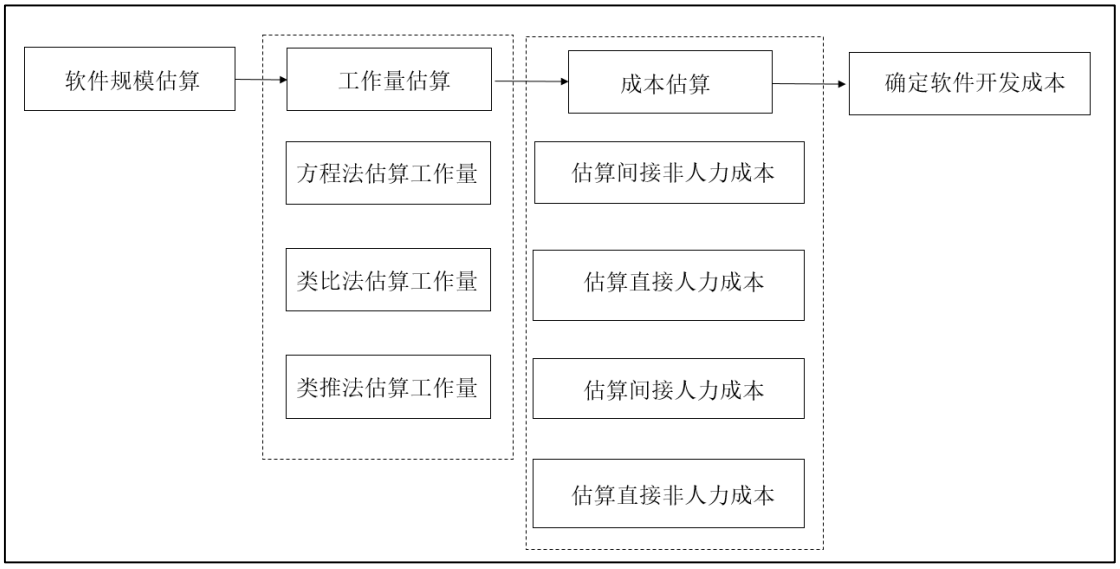

图 5.1 软件开发成本估算基本流程[41]

5.2 软件规模估算

根据第二章对各种软件开发成本估算模型的比较分析，本文将采取符合国际标准 ISO/IEC 14143－1：2007 的 NESMA 功能点分析（FPA）中的详细功能点分析方法进行估算。

5.2.1 NESMA 功能点分析法介绍

NESMA（荷兰软件度量协会）标准是目前国内外最通用、最实用的功能点分析方法，该方法借鉴了 IFPUG 方法中功能点类型的分类，与 IFPUG 方法兼容，该方法提出基于软件逻辑模型的分析规则，站在用户的视角，根据需求文档的详细程度采用不同精度的估算方法。

5.2.2 NESMA 功能点分析法发展历史

功能点分析法具有三十多年的发展的历史，最初是由 IBM 的工程师 Allan Albrecht 于 1984 年首次公开提出用于度量软件功能规模的 Albrecht 法[41].1986 年，国际功能点用户组（IFPUG）成立后，通过不断改进度量软件功能规模的 Albrecht 法，现已形成了较为详细的功能点度量法，主要从用户的角度进行识别数据功能（内部逻辑文件 ILF 和外部接口文件

EIF）以及事务功能（外部输入 EI/外部查询 EQ/外部输出 EO），通过计算项目复杂度并且结合了 14 个调整因子，得出了估算的软件功能点数（即软件规模）。

随后，NESMA（荷兰软件度量协会）对于功能点度量方法进行改进后，形成了国际标准 ISO/IEC 24570 《功能点分析应用定义和计数指南》。此标准指出，在软件项目不同的需求阶段，需采取不同的估算参数，比如在项目初期阶段，需求尚未全部明确以及拆分，FPA 中只需要计数内部逻辑文件（ILF）和外部接口文件（EIF）数量即可初步得到预估的软件规模。在系统需求逐步明确后，即可采用估算功能点法进行估算。

NESMA 标准继承并发展了 IFPUG 标准提出的功能点分析方法，在 2005 年正式升为国际标准，随后在巴西、美国、意大利、韩国、印度、德国、西班牙、日本、加拿大等国应用。

5.2.3 NESMA 功能点分析法的用户功能类型

对于 NESMA 功能点分析法而言，主要有五种用户功能类型：内部逻辑文件（ILF）、外部接口文件（EIF）、外部输入（EI）、外部查询（EQ）、外部输出（EO）。

A、 内部逻辑文件（Internal Logical Files, ILF）

定义

内部逻辑文件是从用户视角可看到的持久型数据的逻辑结合，通过被测程序的一个或多个基本的处理保存数据。一个内部逻辑文件必须符合以下标准要求：

a、 为被测应用程序所用；

b、 为被测应用程序所维护

标准

a、假定采用的为概念数据模型；

b、数据须由被测应用程序进行维护；

c、数据必须是持久性的；

d、对于用户来说，数据组必须是可识别的、可用的、可理解的，并且重要的。

内部逻辑文件复杂度矩阵

表 5.1 内部逻辑文件（ILF）复杂度矩阵

记录类型	数据元素类型		
	1-19	20-50	51 以上
1	低	低	中
2-5	低	中	高
6 个以上	中	高	高

注：

a、记录类型数量为包含实体类型的数量；

b、当数据元素类型或内部逻辑文件记录数量未知时，或者在对项目进行简略分析时，

待识别的内部逻辑文件的复杂度默认为低

B、 外部接口文件（External Interface Files, ELF）

定义

外部接口文件是仅用于参考引用的目的、相关逻辑数据、用户可识别的数据组。是从用户的视角看到的永久性数据的一个逻辑结合。一个外部接口文件必须符合以下标准要求：

a、 为被测应用程序所用；

b、 不是由被测应用程序所维护；

c、 是由其他应用程序进行维护的；

d、 可以直接为被测程序所用。

标准

a、假定采用的为概念数据模型；

b、数据不应该由被测应用程序进行维护；

c、数据必须是功能持久性的；

d、对于用户来说，数据组必须是可识别的、可用的、可理解的，并且重要的。

内部逻辑文件复杂度矩阵

表 5.2 外部接口文件（ELF）复杂度矩阵

记录类型	数据元素类型		
	1-19	20-50	51 以上
1	低	低	中
2-5	低	中	高
6 个以上	中	高	高

注：

a、记录类型数量为包含实体类型的数量；

b、当数据元素类型或外部接口文件记录数量未知时，或者在对项目进行简略分析时，待识别的外部接口文件的复杂度默认为低

区分 ILF 和 ELF 的概述

表 5.3 外部接口文件（ELF）和内部逻辑文件（ILF）区别概述复杂度矩阵

场景		标识为	
在被计数的应用程序中	在其他应用程序中	在被计数的应用程序中	在其他应用程序中
只使用	使用和维护	ELF	ILF
使用和维护	只使用	ILF	ELF
使用和维护	使用和维护	ILF	ILF

续表 5.3

场景		标识为	
只使用	只使用	N/A	N/A

注：

ILF=内部逻辑文件

ELF=外部逻辑文件

N/A=逻辑文件不可能只是被使用，必须使用有一个应用程序负责其维护

C、 外部输入（External Inputs, EI）

定义

外部输入是将数据从该应用系统外部带入系统内部，数据来源可能是界面直接输入，或者是通过其他系统传过来。数据可以是控制信息，也可以是业务逻辑信息。

标准

a、是一个独特的基本过程；

b、是用户定义了它；

c、数据跨越了被测数据应用程序的边界；

d、通常可以导致再一个或多个内部逻辑文件(ILF)中添加、修改和删除数据。

表 5.4 外部输入（EI）复杂度矩阵

引用文件类型（FTR）	数据元素类型		
	1-4	5-15	16 以上
0-1	低	低	中
2	低	中	高
3 个以上	中	高	高

注：

a、当外部输入的数据元素类型和逻辑文件的数量未知时，或者当制定简略分析时，外部输入的复杂度默认为中。

D、外部输出（External Outputs, EO）

定义

外部输出是一种被用户所认知的独特跨越应用程序边界的输出功能。它在规模大小上通常不确定，或需要进一步对其数据进行处理。这些外部输出的数据，是外部接口文件的数据和多个内部逻辑文件经过逻辑运算或计算公式所衍生出的。

标准

a、是一个独特的基本过程；

b、是用户定义了它；

c、分布的信息跨越了被测应用程序的边界；

d、可能包含了一些选择标准，或其他控制信息的输入，但并不一定非得包含此类输入；

f、输出规模大小不确定或输出可能包含数据的处理结果。

表 5.5 外部输出（EO）复杂度矩阵

引用文件类型（FTR）	数据元素类型		
	1-5	6-15	20 以上
0-1	低	低	中
2-3	低	中	高
4 个以上	中	高	高

注：

a、当外部输入的数据元素类型和逻辑文件的数量未知时，或者当制定简略分析时，外部输出的复杂度默认为中。

F、外部查询（External Inquiries, EQ）

定义

外部查询是一个被用户认识到的独特的输入/输出的组合，外部查询中，作为输入的结果，应用程序发布了不需要进一步数据处理、规模既定的输出。

标准

a、是一个独特的基本过程；

b、是用户定义了它；

c、分布的信息跨越了被测数据应用程序的边界；

d、可能包含一些选择标准；

f、输出规模大小确定；

g、输出中不包含数据进一步处理的结果；

h、不会发生对于内部逻辑文件的修改。

表 5.6 外部查询（EQ）复杂度矩阵

引用文件类型（FTR）	数据元素类型		
	1-5	6-19	20 以上
0-1	低	低	中
2-3	低	中	高
4 个以上	中	高	高

注：

a、当外部输入的数据元素类型和逻辑文件的数量未知时，或者当制定简略分析时，外部查询的复杂度默认为中。

5.2.4 NESMA 功能点分析法的应用

FPA 最初被用于度量软件系统开发和应用程序后期简称系统维护的生产率。由于 FPA 所需的数据在项目之初即可获得，所以很快被用于支持项目预算。

同时，根据软件需求的详细程度，可以选择对三种类型的功能点分析，按照详细程度分类，依次是指示功能点分析；简略功能点分析；详细功能点分析。这些分析不仅可以用来估算应用程序的成本效益，同时对于人力资源的配置也有很大的帮助。

此外，FPA 还可以用于例如监督项目的进展，根据前期预估结果制定项目进度和开发策略，当然也可以从不同角度，使用不同方法进行度量，使软件规模估算更具可靠性。

5.2.5 应用 NESMA 功能点估算法估算 DDP 一期项目规模

图 5.2 为 DDP 一期的数据流图，依据此图，将计算得出 NESMA 功能点分析法中的五类用户功能文件数量，从而估算 DDP 一期项目规模。

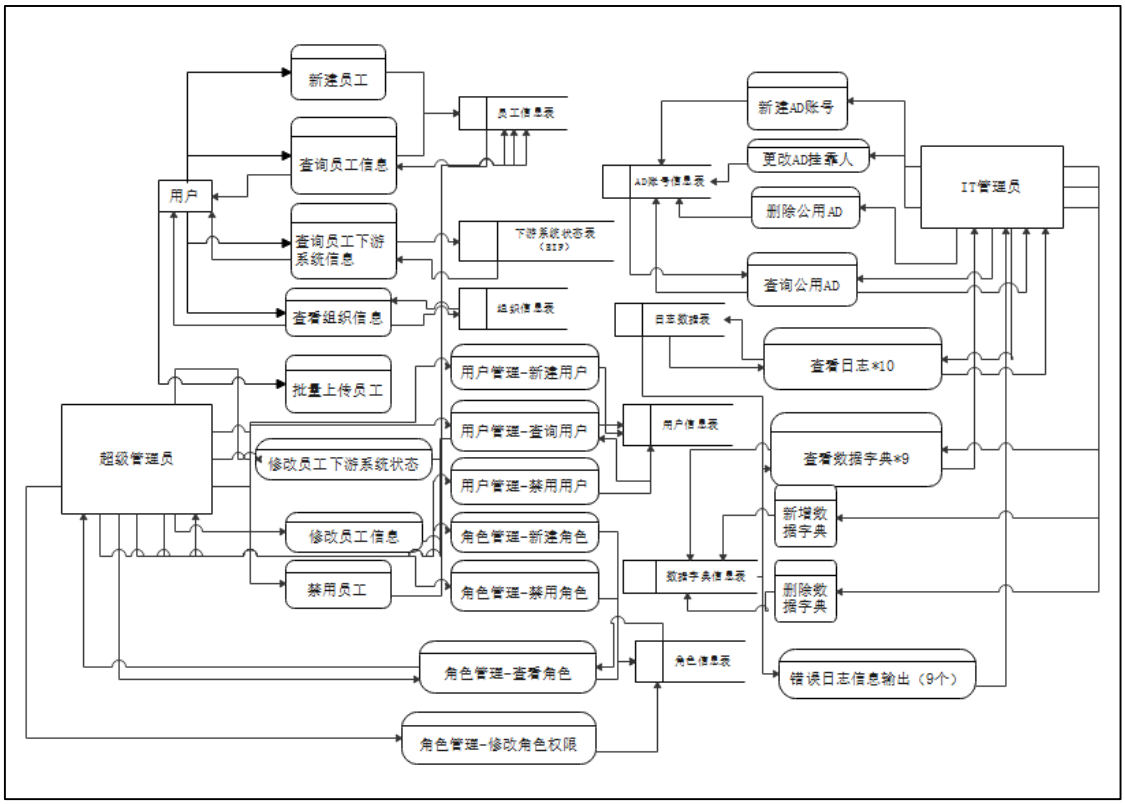

图 5.2 DDP 一期项目数据流图

根据对相关流程梳理，可得各类用户功能类型如下：

表 5.7 DDP 一期项目各类用户功能类型数量

	内部逻辑文件	外部接口文件	外部输入	外部输出	外部查询
数量	7	7	15	9	8

表 5.9 为根据图 5.2 数据流图以及针对该项目特征判断不同类型用户功能类型复杂度，得到的 DDP 一期项目估算明细：

表 5.8 功能点参照表

复杂度	功能类型				
	ILF	ELF	EI	EO	EQ
低	7	5	3	4	3
中	10	7	4	5	4
高	15	10	6	7	6

注：

ILF=内部逻辑文件　　EI=外部输入

ELF=外部接口文件　　EO=外部输出

EQ=外部查询

表 5.9 DDP 一期项目规模估算明细

要素	复杂度	权重	个数	功能点数
内部逻辑文件（ILF）	低	7	7	49
	中	10	0	0
	高	15	0	0
外部接口文件（ELF）	低	5	0	0
	中	7	6	42
	高	10	1	10
外部输入（EI）	低	3	0	0
	中	4	14	56
	高	6	1	6
外部输出（EO）	低	4	0	0
	中	5	9	45
	高	7	0	0
外部查询（EQ）	低	3	6	18
	中	4	2	8
	高	6	0	0

续表 5.9

总计：	233

考虑不同的估算阶段和历史数据规模变更系数，调整后的规模为：

$$S = US * CF \quad (5.1)$$

式中：

S —— 调整后规模，单位为功能点（FP）；

US —— 未调整规模，单位为功能点（FP）；

CF —— 规模变更因子，取值范围 1.0~2.0，计划阶段取值 1.0.

故调整后的规模计算结果为 233 FP

5.3 工作量计算

下述为 DDP 一期项目的 WBS 以及各类角色的工作比例、实际总工时情况：

5.3.1 项目分解结构（WBS）

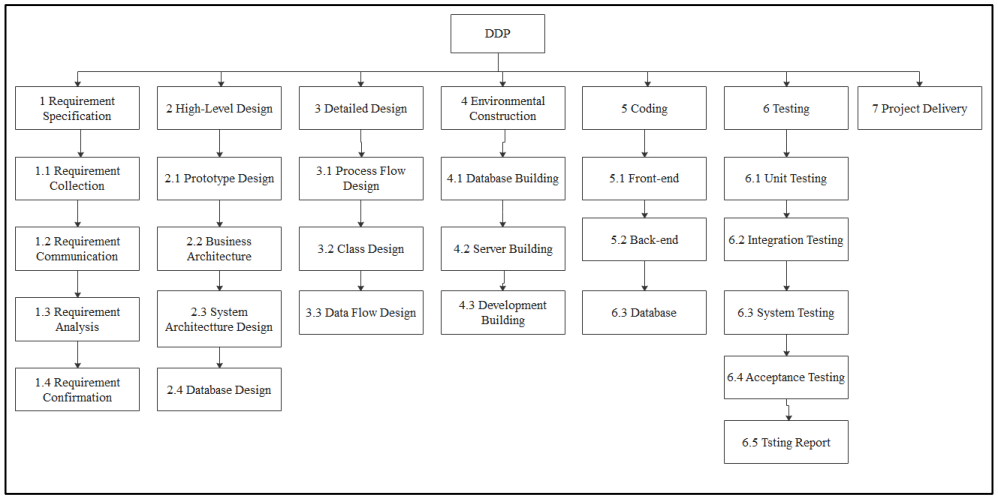

图 5.3 DDP 一期项目分解结构（WBS）

5.3.2 各角色投入比例

整理收集项目各类角色的工作比例和工时如图 5.4 及表 5.10，其中，各角色投入比例反应了每个角色在项目执行过程中需要投入的工作量比例：

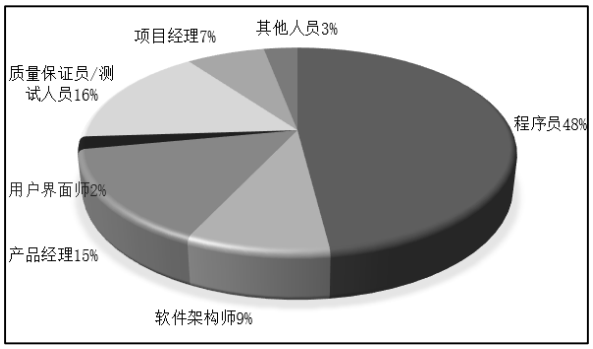

图 5.4　内部软件开发角色投入比例

表 5.10　各类角色人数和总有效人时

角色	人数	总有效人时
项目经理	1	100
软件架构师	1	130
产品经理	1	216
UI 设计	1	30
质量保证员/测试人员	1	228
程序员	3	698
其他人员	1	38

5.4 基准建立过程

5.4.1　基准建立目的

本基准是根据 NESMA 功能点分析法（FPA）和项目特征进行估算，同时在《软件开发成本度量规范》[43]国家标准的基础上进行误差调整，实现基于软件需求估算软件规模，建立同类项目的成本基准（即劳动生产率）。

5.4.2　基准适用范围

NESMA 功能点估算法（FPA）是针对软件功能需求和软件开发规模的间接定量测量，是基于客观的外部应用接口、通用性能以及主观的内部应用程序复杂度进行测量的。

而针对软件项目的独特性，在应用本基准时，项目还需要满足以下特征：

应用范围：本组织同类自研项目

质量要求：高可靠性、灾备、高可扩展性

采用技术：主流开发技术

开发团队：小规模开发团队，人员配比与图 5.4 相似

非功能性需求：用户友好

5.4.3 基准的输入

在基准建立之后，输入按系统模块划分或总的外部输入、外部输出、外部查询、内部逻辑文件、外部接口文件数量以及复杂度。

5.4.4 基准的输出

计算开发特定项目的工期，制定项目开发计划和资源需求。

5.4.5 基准的建立

DDP 一期项目总功能点数为233 个， 总实际有效工时为 1440 人时。

在结合 ISBSG 大数据[41]关于内部软件开发各角色在项目中投入比例以及有效项目真实数据[42]后，汇总 DDP 一期项目各角色实际投入的有效工时后，故本项目的劳动生产率和各角色所占比例如下：

$$劳动生产率 = 1440/233=6.18 \quad （5.2）$$

表 5.11 为中国软件行业基准数据报告（SSM-BK-201706）中生产率基准数据明细[41]，本文得出的劳动生产率在 P25-P50 范围内，考虑项目特殊要求以及人员配置，故符合实际情况。

表 5.11 生产率基准数据明细（SSK-BK-201706）[41]

生产率（单位：人时/功能点）				
P10	P25	P50	P75	P90
2.87	4.56	6.91	14.23	22.23

5.5 本章小结

本章基于第二章对于不同软件成本估算模型的分析比较，采取软件开发成本估算模型 NESMA 功能点分析法（FPA）中的详细功能点法对 DDP 一期项目的开发规模及组织生产率进行分析估算，得到劳动生产率为 6.18 p.h/FP ，并将此结果作为本组织同类项目的软件成本基准，用于后续实现高效的项目管理。

6 基准的验证与修订

本章将采用《软件开发成本度量规范》国家标准[43]对已建立的本项目的组织生产率（基准）计算结果进行验证并对可能产生的误差做出修正，章末得到本组织开发同类项目的基准。

6.1 基准误差验证与修订过程

6.1.1 验证方法

考虑规模估算以及相应计算可能存在的误差，本章将采取《软件开发成本度量规范》国家标准[43]对已建立的基准进行修订。

6.1.2 验证过程（应用《软件开发成本度量规范》国家标准[43]）

A、 国标适用范围

国家标准中规定了软件研发成本度量的方法及过程，包括软件研发成本度量过程、软件研发成本的构成、软件研发成本度量的应用。

本标准可适用于度量与软件规模相关的软件研发项目的成本。

B、 使用国标进行修订目的

在上述 NESMA 功能点分析法中，针对项目风险、项目功能复用等方面并没有充分考虑，所以在本章将采用《软件开发成本度量规范》国家标准[43]对本基准计算过程中可能产生的误差进行修订。

C、 估算准备

在进行工作量估算前，应：

a） 对项目风险进行分析。

风险分析时应考虑技术、管理、资源和商业多方面因素。例如：需求变更、时间或成本约束外部、外部协作、系统架构、人力资源、采用新技术以及外购或复用等；

b） 对实现功能复用的情况分析，识别可复用的功能以及程度；

c） 依据经验或者相关性分析结果，主要确定工作量的属性；

需要考虑的因素为（不限于以下因素）：

1） 软件规模；

2） 软件的完整性级别；

3） 应用领域；

4） 质量要求，如易用性、性能效率、可维护性、可移植性等等。

5） 工期要求，如紧迫度等；

6） 采用的技术，如开发平台、系统架构、编程语言等；

7） 过程能力，如团队规模、人员能力等；

8） 选择合适的工作量估算方法等。

D、 软件项目规模估算与调整

在进行工作量估算调整时，需考虑以下因素：

(a) 根据项目风险分析的结果，对估算方法或模型进行合理调整。如调整估算模型其影响因子的权重；

(b) 根据项目的可复用规模以及可复用程度进行调整；

同时，因为防止不同方法的估算结果产生较大差异，本结果将结合上述 NESMA 功能点分析法，对最终得到的结果进行交叉验证。

基于基准建立的回归方程如下式：

$$UE = C * S^a \tag{6.1}$$

式中：

UE —— 未调整工作量，单位为人时（p.h）；

C —— 生产率调整因子，单位为人时每工能点（p.h/FP）

S —— 软件规模，单位为功能点（FP）；

a —— 基于基准数据计算出的软件规模调整系数。

按照相关性分析和经验调整后工作量计算公式按下式计算：

$$AE = UE * A * IL * L * T \tag{6.2}$$

式中：

AE —— 已调整工作量，单位为人时（p.h）；

A —— 应用领域调整因子，取值范围 0.8~1.2；

IL —— 软件完整性级别，取值范围 1.0~1.8；（按照行业历史数据计算的软件完整性各级别参考范围是：A 级取 1.6~1.8，B 级取 1.3~1.5，C 级取 1.1~1.2，D 级取 1.0）

L —— 开发语言调整因子，取值范围 0.8~1.2；

T —— 最大团队规模调整因子，取值范围 0.8~1.2.

待估算项目的规模为 233 FP，参考基准[41]数据的功能点耗时率 25 百分位数、50 百分位数和 75 百分位数，C 取值分别为 4.56 p.h/FP、6.91 p.h/FP、14.23 p.h/FP，软件规模调整系统为，则计算出未调整工作量合理范围介于 1832.6 p.h 与 5718.7 p.h 之间，未调整工作量最可能值为 2777.0 p.h。

根据本组织实际情况，确定应用领域调整因子取值为 0.8，软件完整性级别取值为 1，开发语言调整因子取值为 0.8，最大团队规模调整因子取值为 0.8，则计算调整后工作量介于 938.3 p.h 与 2928.8 p.h 之间，最有可能值为 1421.8 p.h。

E、 验证结果

故调整的软件规模在 938.3 p.h 与 2928.8 p.h 之间，最有可能值为 1421.8 p.h。

6.1.3 修订基准

DDP 一期项目实际人时为 1440 p.h，修订结果范围为 938.3 p.h~2928.8 p.h，取最有

可能值 1421.8 p.h，即误差为 1.264 %。

故可计算劳动生产率（p.h/FP）：

$$\text{劳动生产率} = 1440/\{233 * (1 \pm 1.264\%)\}$$

$$\text{劳动生产率} = 6.18 * (1 \pm 1.264\%) \quad (6.3)$$

表 6.4 为中国软件行业基准数据报告（SSM-BK-201706）中生产率基准数据明细[41]，本文得出的劳动生产率在 P25-P50 范围内，故符合实际情况。

表 6.4 生产率基准数据明细（SSK-BK-201706）[41]

生产率（单位：人时/功能点）				
P10	P25	P50	P75	P90
2.87	4.56	6.91	14.23	22.23

6.2 本章小结

本章将第五章得到的基准通过《软件开发成本度量规范》国家标准[43]验证及修订误差，得到劳动生产率为 $6.18 * (1 \pm 1.264\%)$ p.h/FP。修订后适当地缩减了基准的误差范围，以便更准确的用于项目的后续应用开发管理。

装 订 线

7 基准的应用

本章通过软件开发成本估算模型 NESMA 功能点分析法估算 DDP 二期规模后，利用修订后的基准计算项目工期，同时进行二期项目计划及资源计划的编排。

7.1 软件规模估算

7.1.1 DDP 二期各类用户功能类型数量

下图为 DDP 二期数据流图，依此可计算各类用户功能类型数量如表 7.1 所示：

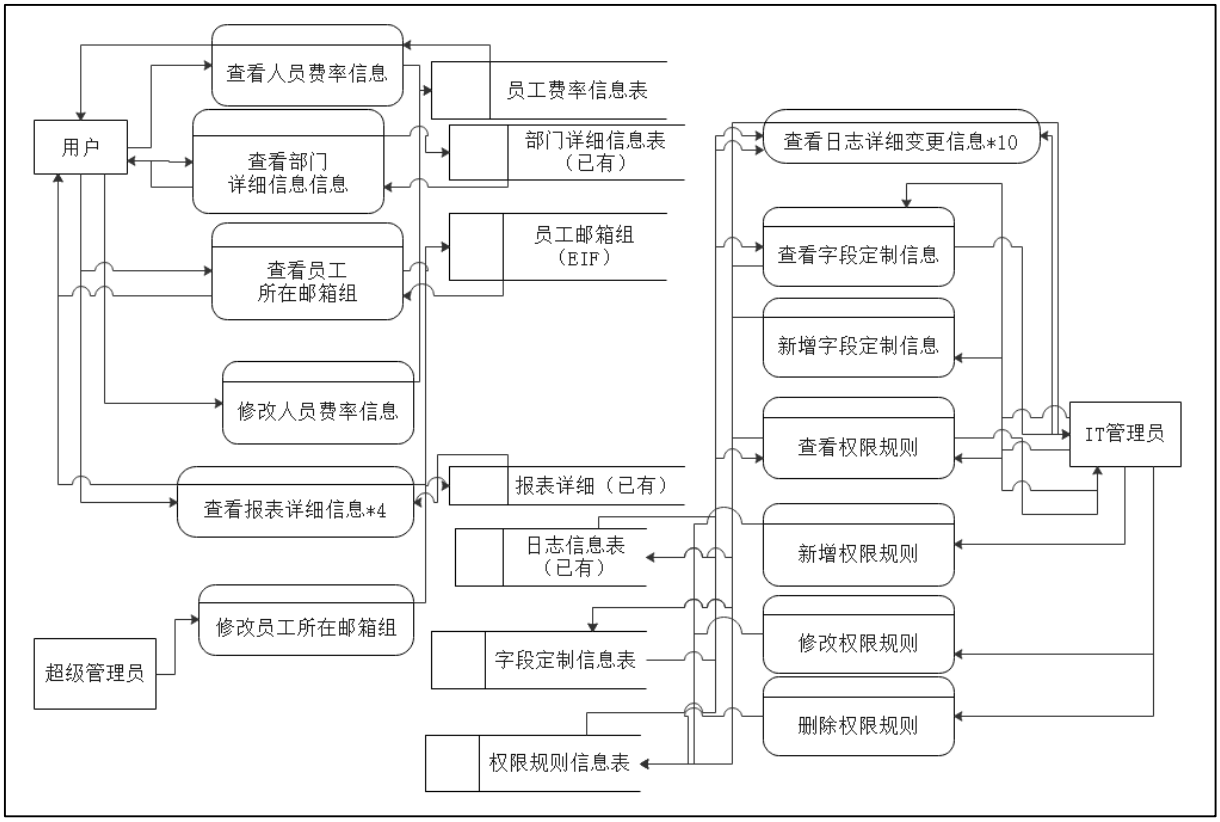

图 7.1 DDP 二期项目数据流图

根据对相关流程梳理，可得各类用户功能类型如下：

表 7.1 DDP 二期项目各类用户功能类型数量

	内部逻辑文件	外部接口文件	外部输入	外部输出	外部查询
数量	3	0	6	19	16

7.1.2 DDP 二期项目规模估算

表 7.3 为根据图 7.1 数据流图以及针对该项目特征判断不同类型用户功能类型复杂度，得到的 DDP 二期项目估算明细：

表 7.2 功能点参照表

复杂度	功能类型				
	ILF	ELF	EI	EO	EQ
低	7	5	3	4	3
中	10	7	4	5	4
高	15	10	6	7	6

注：

ILF=内部逻辑文件　　EI=外部输入

ELF=外部接口文件　　EO=外部输出

EQ=外部查询

表 7.3 DDP 二期项目规模估算明细

要素	复杂度	权重	个数	功能点数
内部逻辑文件（ILF）	低	7	1	7
	中	10	0	0
	高	15	2	30
外部接口文件（ELF）	低	5	0	0
	中	7	0	0
	高	10	0	0
外部输入（EI）	低	3	1	3
	中	4	5	20
	高	6	0	0
外部输出（EO）	低	4	1	4
	中	5	18	90
	高	7	0	0
外部查询（EQ）	低	3	2	6
	中	4	14	56
	高	6	0	0
			总计：	216

考虑不同的估算阶段和历史数据规模变更系数，调整后的规模为：

$$S = US * CF \quad (7.1)$$

式中：

S —— 调整后规模，单位为功能点（FP）；

US —— 未调整规模，单位为功能点（FP）；

CF —— 规模变更因子，取值范围 1.0~2.0，计划阶段取值 1.0.

故调整后的规模计算结果为 216 FP

7.2 工作量估算

7.2.1 不同项目规模的工期范围

表 7.4 和图 7.2 是 ISBSG 数据库中对于不同项目工期范围的计算所得出的标准[42]。

表 7.4 不同项目规模的工期范围明细[42]

项目规模（未调整功能点）	项目数量	最小值	P10	P25	中位数	P75	P90	最大值	均值	标准差
100 及以下	490	0.1	1	1.9	3	5	8	36	3.9	3.4
101-200	324	0.1	2	3	5	8	12	21	5.9	3.8
201-300	208	0.1	3	3.6	6	9	13	33	7.1	5.2
301-400	125	0.3	2.9	3.4	6	10	13	33.6	7.7	5.8
401-500	90	1	2.9	3.9	6	9.8	14.1	35	7.5	5.4
501-600	47	1.1	3	4	6	8	12	16	6.4	3.5
601-700	41	3.5	4.9	6	8	10	16	34	9.3	5.6
701-800	34	3	4.3	5.6	8.8	12.2	22.8	84	12	14
801-900	13	2	5	6	10	13	17	21	10.1	5.4
901-1000	20	0.6	2	3.8	9.5	12.2	14.3	68	11.1	14.2
1001-1200	29	1.7	5.2	6.3	9	12	16.8	24	10.3	5.5
1201-1400	21	2.7	3	6.9	9	12	16.9	20.4	9.8	5.1
1401-1600	15	4.1	5	6.5	10	11.5	16.8	20	10	4.7
1601-2000	19	0.9	3	5	10.7	12	17.2	37	10.5	8.6

续表 7.4

项目规模（未调整功能点）	项目数量	最小值	P10	P25	中位数	P75	P90	最大值	均值	标准差
2001-3000	24	3	5.2	6.7	9.1	12	19.4	24	10.9	5.7
3001 及以上	13	5	8.8	13.9	17.9	22	29.6	44	19.3	10

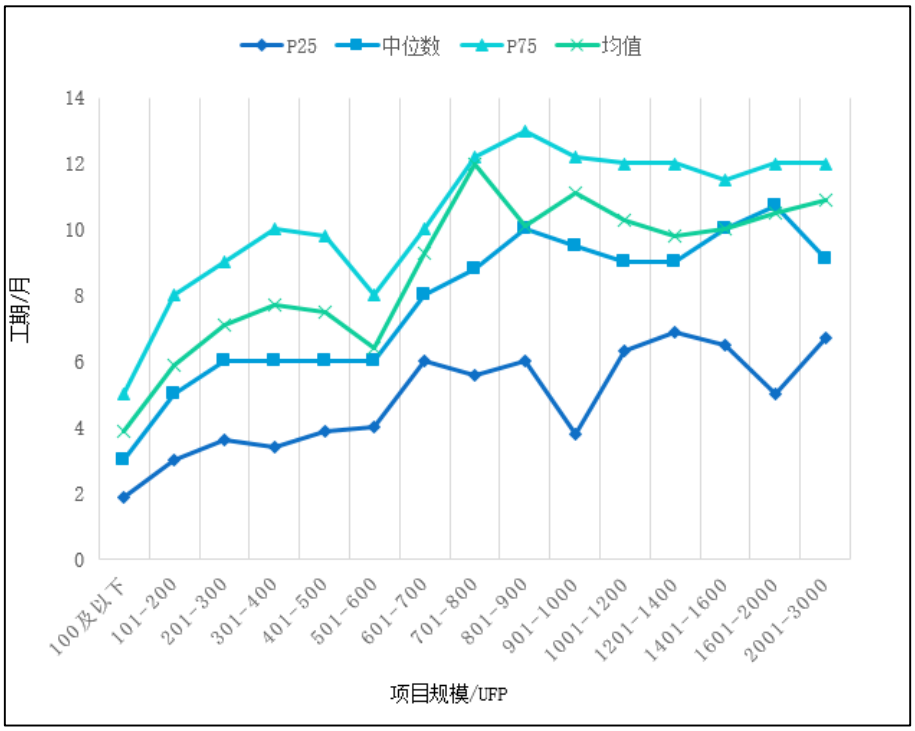

图 7.2　不同项目规模的平均工期[42]

在上述图表中，P 表示 percent，例如 P25 表示有 25%的项目满足所对应的工期范围。DDP 二期的 UFP 为 216 FP，项目工期约在 3.6-9 个月。

7.2.2　不同项目工作量的工期范围

表 7.5 和图 7.3 是 ISBSG 数据库中对于不同项目工作量类型的计算所得出的标准[42]。

表 7.5 不同项目工作量类型的工期范围明细[42]

工作量/小时	数量	最小值	P0	P25	中位数	P75	P90	最大值	均值	标准差
400 以下	229	0.1	0.8	1.2	2	3	6	16	2.7	2.4

续表 7.5

工作量/小时	数量	最小值	P0	P25	中位数	P75	P90	最大值	均值	标准差
401-800	211	0.2	1.6	2.1	3.2	5	7	13	3.9	2.4
801-1200	158	0.5	2	3	4	6	8	19	4.6	2.9
1201-1600	115	1	2.6	3	5	6.1	9.6	15	5.3	2.9
1601-2000	89	0.1	2.4	3.4	6	7	10	13	5.7	2.8
2001-2400	90	0.2	3	3.9	5.8	8	11.1	15	6.2	3.4
2401-2800	63	0.1	3	4	6	8.6	14.6	32	7.5	5.5
2801-3200	67	1	3	4	7	9.4	12.4	84	8	10
3201-3600	44	1.1	4	6.9	9	10	36	36	9.1	5.5
3601-4000	45	2.3	3.5	4	7	9	19	19	7.5	3.9
4001-5000	76	0.9	3	5	7.5	11.6	34	34	9	6.6
5001-6000	57	2	4	6	9	12	29	29	9.2	4.9
6001-8000	72	2.9	3	4	8.8	12	24	24	8.7	4.9
8001-10000	58	0.1	3	4.5	8	11	21	21	8.7	4.7
10001-20000	100	0.2	3	7	10	13	27.6	27.6	10.2	5.3
20001 及以上	39	4.9	6.8	14	19	24	68	68	20.5	12

装 订 线

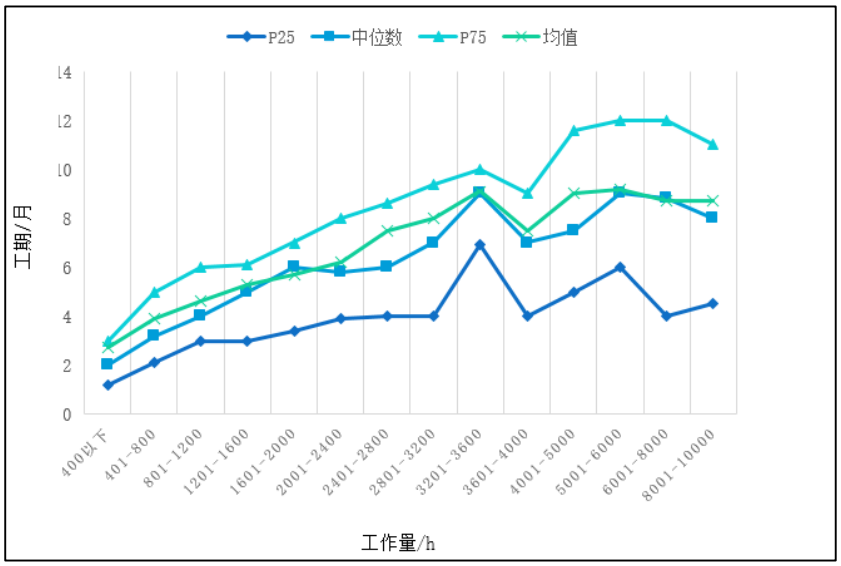

图 7.3　不同项目工作量类型的工期范围明细[42]

7.2.3　结论

DDP 二期总功能点为 216 个，根据第六章得到的基准

$$\text{总劳动生产率 } (\text{p.h/FP}) = 6.18 * (1 \pm 1.264\%) \qquad (7.2)$$

可计算 DDP 二期的工期为 1351.8 p.h － 1318.0 p.h，根据表 6.5，可估算工期为 3-6.1 个月，与表 6.4 得出的 3.6-9 个月取交集后，DDP 二期估算工期为 3.6-6.1 个月。

因为项目时间要求，并且综合估算结果，故以下的项目计划和预算均按照工期为 4 个月的标准。

7.3 成本估算

根据 2017 年中国软件行业基准数据[41]，采取上海地区的人月费率为￥25,358，可估算 DDP 二期项目成本（不包括直接非人力成本）约为：

$$\text{COST} =$$

$$2.16 * 6.18 * (1 \pm 1.264\%) \ /8/22.5 * \ ￥25{,}358$$

$$= ￥1{,}856{,}779 \sim ￥1{,}930{,}196 \qquad (7.3)$$

7.4 应用基准进行项目规划

根据上述通过基准计算得出的工期和相应成本，可对 DDP 二期进行如下规划：

7.4.1　关键路径(CPM)

下图为根据 DDP 二期各阶段任务和预计所用时间，绘制的关键路径分析，可用于找到

影响工期的最关键任务并加以监控。

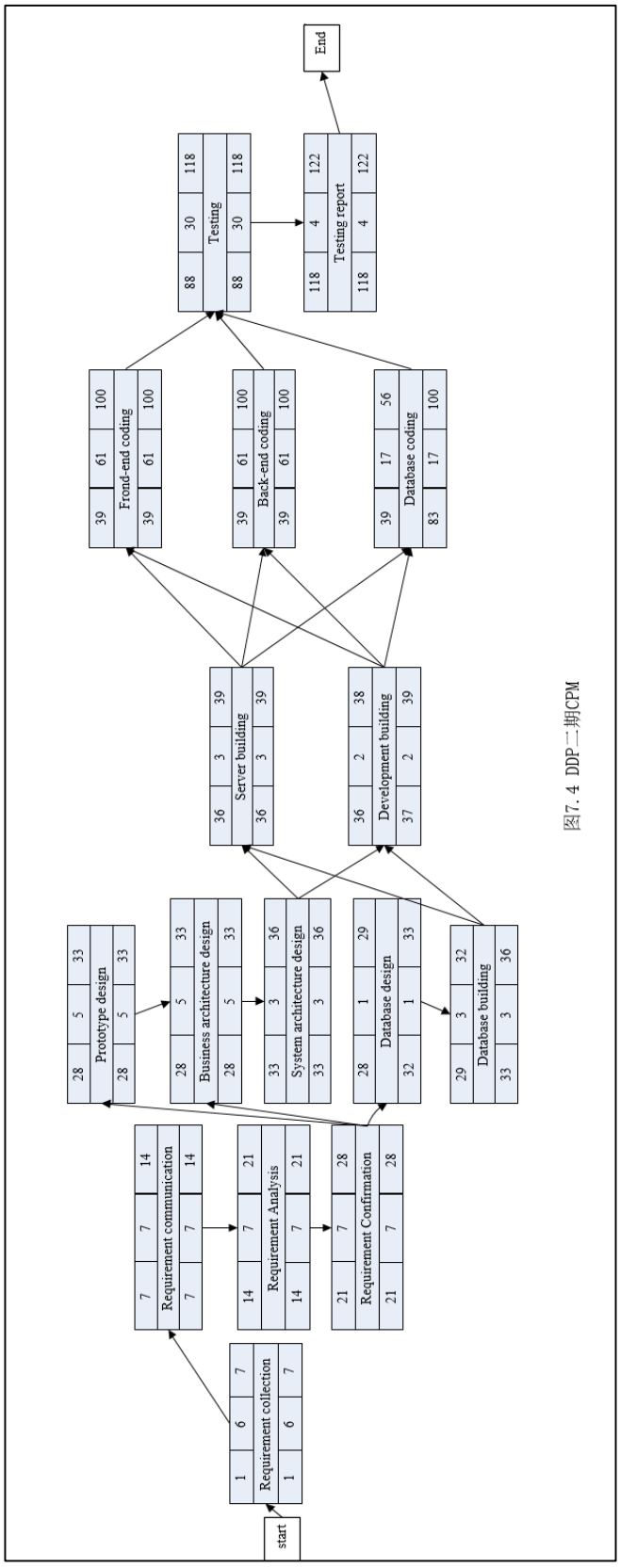

图7.4 DDP二期CPM

7.4.2 进度安排(Schedule)

下表为根据估算得出的工期和历史数据，通过商讨得出的 DDP 二期进度安排(Schedule)，依此监督二期项目的实施。

表 7.6 DDP 二期进度安排（Schedule）

Phase and Step	Duration	Starting Date	Ending Date	Relevant personnel
1 Requirement specification	26	2020/2/11	2020/3/9	project manager, users, programmer
1.1 Requirement collection	7	2020/2/11	2020/2/17	product manager
1.2 Requirement communication	7	2020/2/18	2020/2/24	product manager
1.3 Requirement Analysis	7	2020/2/25	2020/3/2	product manager
1.4 Requirement Confirmation	7	2020/3/3	2020/3/9	product manager，project manager
2 High-Level design	5	2020/3/10	2020/3/14	project manager,
2.1 Prototype design	5	2020/3/10	2020/3/14	project manager, product manager, interaction designer
2.2 Business architecture design	5	2020/3/10	2020/3/14	project manager, architect
2.3 System architecture design	3	2002/3/12	2020/3/14	project manager, architect
2.4 Database design	1	2020/3/13	2020/3/14	architect
3 Detailed design	10	2020/3/15	2020/3/25	project manager, programmer
3.1 Process flow design	10	2020/3/15	2020/3/25	programmer
3.2 Class design	10	2020/3/15	2020/3/25	programmer
3.3 Data flow design	10	2020/3/15	2020/3/25	programmer
4 Environmental construction	5	2020/3/26	2020/3/30	project manager, programmer
4.1 Database building	3	2020/3/26	2020/3/28	programmer
4.2 Server building	3	2020/3/26	2020/3/28	programmer
4.3 Development building	2	2020/3/29	2020/3/30	programmer

续表 7.6

Phase and Step	Duration	Starting Date	Ending Date	Relevant personnel
5 Coding	61	2020/3/31	2020/5/31	project manager, programmer
5.1 Front-end	61	2020/3/31	2020/5/31	programmer
5.2 Back-end	61	2020/3/31	2020/5/31	programmer
5.3 Database	17	2020/3/31	2020/4/16	programmer
6 Testing	30	2020/5/15	2020/6/13	project manager, tester
6.1 Unit testing	7	2020/5/15	2020/5/21	tester
6.2 Integration testing	7	2020/5/22	2020/5/28	tester
6.3 System testing	7	2020/5/28	2020/6/3	tester
6.4 Acceptance testing	7	2020/6/3	2020/6/9	tester
6.5 Testing report	4	2020/6/10	2020/6/13	project manager, tester
7 Project Delivery	1	2020/6/13		project manager

装 订 线

7.4.3 甘特(Gantt)图

下表为根据估算得出的工期和历史数据，通过商讨得出的 DDP 二期甘特(Gantt)图，由于 Gantt 图可通过活动列表和时间刻度显示出特定项目的顺序与持续时间，直观地表示计划何时开始，每项计划的总时间和分支计划的时间规划，故有利于管理者弄清项目的剩余任务，合理地评估工作进度。

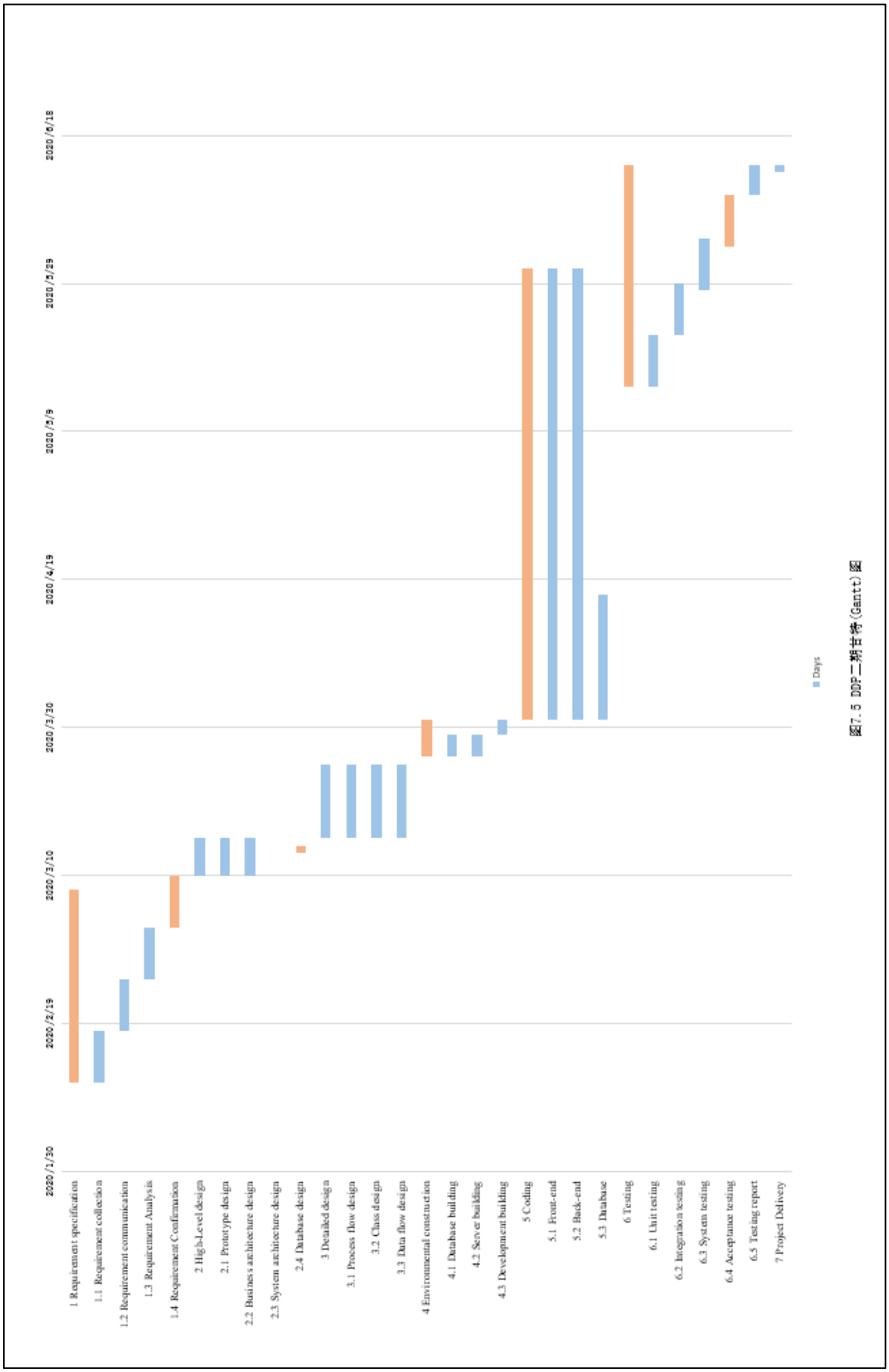

图7.5 DDP二期甘特(Gantt)图

7.5 本章小结

本章应用第六章误差修订后的基准指导 DDP 二期项目，通过软件开发成本估算模型 NESMA 功能点分析法估算 DDP 二期项目规模为 216 FP，使用基准计算工期为四个月，随后，根据项目实际情况规划二期项目的进度安排(Schedule)、绘制关键路径(CPM)和甘特(Gantt)图，便于 DDP 二期项目管理及进度监督等。

8 基于挣值分析（EVA）修订基准

考虑 DDP 二期实际进展和预期之间可能存在一定的误差，在本章，将采用挣值分析法监督、控制计划值与实际值之间的偏差，修正已有基准，最终得到适用于本组织的准确的生产率基准。

8.1 挣值分析简介

挣值分析（EVA）主要用于项目成本和进度监控，是指将实际进度和成本绩效与绩效测量基准进行比较。挣值通过项目开始时制定的计划和实际所完成的工作进行对比，给出了一个项目何时完工的估算，通过从项目已经完工的部分进行推算，就可以估计出项目完工时所耗费的资源。

8.1.1 挣值分析基本参数

（1） 计划值（PV）

又称计划工作量的预算费用（BCWS），是指项目实施过程中某阶段计划要求完成的工作量所需的预算工时（费用），计算公式为：

$$PV = BCWS = 计划工作量 * 计划单价 \quad (8.1)$$

PV 主要反映进度计划应当完成的工作量，而不是反映所消耗的工时费用。

（2） 实际成本（AC）

又称已完成工作量的实际费用（ACWP），是指项目实际实施过程中某阶段实际完成的工作量所消耗的工时（费用），主要反映项目执行的实际消耗指标。计算公式为：

$$AC = ACWP = 已完成工作量 * 实际单价 \quad (8.2)$$

（3） 挣值（EV）

又称已完成工作量的预算成本（BCWP）是指项目实际实施过程中某阶段实际完成工作量及按预算定额计算出来的工时（费用），计算公式为：

$$EV = BCWP = 已完成工作量 * 计划单价 \quad (8.3)$$

8.1.2 挣值分析评价指标

a） 进度偏差（SV），SV 是指检查日期 EV 和 PV 之间的差异。

$$SV = EV - PV = BCWP - BCWS \quad (8.4)$$

当 SV 为正值时，表示进度提前；

当 SV 等于零时，表示计划和实际相符；

当 SV 为负值时，表示进度延误。

b） 成本偏差（CV），是指检查期间 EV 和 AC 之间的差异。

$$CV = EV - AC = BCWP - ACWP \quad (8.5)$$

当 CV 为正值时，表示实际消耗的人工（或费用）低于预算值;;

当 CV 等于零时，表示实际消耗的人工（费用）等于预算值;

当 CV 为负值时，表示实际消耗的人工（费用）高于预算值。

c) 成本执行标准（CPI），是指项目挣值和实际费用之比（或工时值之比）。

$$CPI = EV/AC = BCWP/ACWP \quad (8.6)$$

当 CPI>1 时，表示低于预算，即实际费用低于预算费用;

当 CPI=1 时，表示实际费用与预算费用吻合;

当 CPI<1 时，表示超出预算，即实际费用高于预算费用。

d) 进度绩效标准（SPI），是指项目挣值和计划值之比。

$$SPI = EV/PV = BCWP/BCWS \quad (8.7)$$

当 SPI>1 时，表示进度超前;

当 SPI=1 时，表示实际进度和计划进度相同。;

当 SPI<1 时，表示进度延误。

装

8.2 项目不同阶段误差情况

根据 DDP 二期实际进展情况，以月为单位进行分析，可得实际工期和预计工期比较如图 8.1，误差原因和为实现项目管理流程控制的措施如下：

订

8.2.1 Earned Value Tracking Chart

线

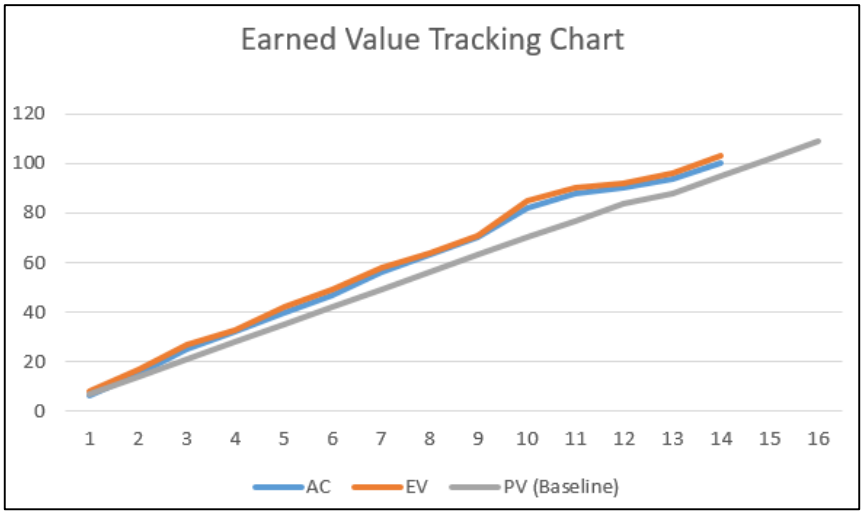

图 8.1 Earned Value Tracking Chart

注：以项目工作量为标准进行分析

其中：PV（计划值），项目实施过程中某阶段计划要求完成的工作量所需的预算工时;

AC（实际成本），项目实施过程中某阶段实际完成的工作量所消耗的工时；

EV（挣值），项目实施过程中某阶段实际完成工作量及按预算定额计算出来的工时。

8.2.2 偏差分析和措施

通过分析每周计划工时和实际工作量可以清晰的比较各阶段的生产效率和预估工作量可能产生的偏差，及时地分析误差产生的原因，并采取相应措施。

以 4.28-5.4 周期为例，预估时只是以周为单位，因节假日原因，出现了两名员工请假两天，所以出现计划和实际偏差较大，针对曲线中不同节点计划和实际的差异，分析原因后，便于项目经理及时调整计划，从而实现高效的项目管理。

8.2.3 完工预计

据二期项目实际进展情况，最后一次统计时间截至 5.15，项目总体进度较计划提前 5 日，计划总工期 123 日。

A、进度偏差

$$SV = EV - PV = 100 - 95 = 5 > 0 \quad (8.8)$$

B、进度绩效指标

$$SPI = EV/PV = 100/95 = 1.05 \quad (8.9)$$

C、完工预计

$$\text{修正后的计划总工期} = 123/1.05 = 117 \quad (8.10)$$

8.3 基准修订结果

DDP 二期实际总工期为 117 日，较计划提前 5 天，经挣值分析法及误差考虑，修订后的劳动生产率（基准）为

$$\text{劳动生产率}(\text{p.h/FP}) = 6.18 * （1 \pm 1.264\%)/1.05$$

$$= 5.89 * （1 \pm 1.264\%) \quad (8.11)$$

适用范围：

项目类型：同类自研项目

质量要求：高可靠性、灾备、高可扩展性

采用技术：主流开发技术

开发团队：小规模开发团队，人员配比与图 5.4 相似

非功能性需求：用户友好

8.4 基准的应用评价

基准的建立，不仅解决了目前软件公司缺乏自主研发软件项目之软件规模度量与成本估算标准以及基准数据的问题，还通过实际项目修正，得到适用于本组织的准确的生产率基准，

为本组织实现高效的项目管理提供了依据。

该基准不仅有助于项目管理人员在项目初期对项目工期进行预估，合理地制定项目计划，而且在项目进展过程中，通过预估结果和实际进展对比，项目管理人员还可以通过分析偏差产生原因，合理调整项目计划。

此外，将项目测量的规模、成本和劳动生产率等数据还可以纳入组织的基准数据库，为以后类似项目的成本估算提供参考数据，也可将测量的功能点耗时率和生产率和行业或组织的基准数据进行对比，从而发现改进的机会。

DDP 项目经理资深应用系统分析师姚俊杰说：“该项目基准的建立，对于 DDP 二期项目工期的预估，资源的分配起到了重要的辅助作用，DDP 一期资源分配不足导致的延期问题，由于前期成本估算相对准确，在 DDP 二期中并无出现，后续通过实际数据对于基准的修订，也使其更加符合项目的实际开发情况和需要，从而为之后的项目实现高效的管理提供依据”。

8.5 本章小结

本章在项目二期开发与管理实践中，运用挣值分析法监督、控制计划值与实际值之间的偏差，修正已有基准，最终得到了适用于本组织的准确的生产率基准 $5.89 * (1 \pm 1.264\%)$，单位为 p.h/FP。与此同时，引用了 DDP 项目的项目经理姚俊杰先生的评价，侧面证明了本基准的实效性和应用价值。

9 结论与展望

9.1 结论

本文将以作者在实习单位从事开发与管理的软件研发项目“DDP 数据分发平台”为背景，针对目前软件公司缺乏自主研发软件项目之软件规模度量与成本估算标准以及基准数据的问题，从研究软件规模度量和软件成本估算模型/标准着手，结合参与“DDP 数据分发平台”项目的实际开发与管理工作的数轮迭代，采用《软件开发成本度量规范》国家标准[43]和软件开发成本估算模型 NESMA 功能点分析等标准，分析、估算及交叉验证该项目一期的开发规模及组织的生产率，将此结果作为本组织开发同类项目的基准(Benchmark)，用以估算项目二期规模，指导项目二期工期与进度及资源计划的编排。在项目二期开发与管理实践中，运用挣值分析法监督、控制计划值与实际值的偏差，修正已有基准，最终得到适用于本组织的准确的生产率基准，为本组织实现高效的项目管理提供了依据。

本论文主要完成了以下工作：

（1） 完成软件成本估算模型的相关理论研究。基于现有国内外软件估算技术现状和相关模型的使用方法和优缺点进行调研，同时系统且深入的学习了软件成本估算的知识，较为全面地分析了现在成本估算对软件开发的重要意义，尤其是在项目管理方面。同时筛选了合适的估算模型用于本文研究。

（2） 完成 DDP 数据分发平台的相关数据收集工作和现有软件成本估算问题收集整理。首先 DDP 平台全程亲历，对相关软件特性和基本架构了解，在此基础上，结合软件成本估算模型的相关数据要求，完成了本文所需数据的收集和整理工作。其次，针对部门现有软件成本估算现状进行汇总，同时针对暴露的问题进行整理后，通过本文探究合理的解决方法。

（3） 完成通过 NESMA 功能点分析法（FPA）分析、估算及交叉验证 DDP 一期项目的规模估算。首先，对 NESMA 功能点分析法的发展历史、5 种用户功能类型以及应用手段进行研究。其次，通过逻辑模型计算各类用户功能数量后进行规模估算，结合国家标准修订的误差后，将此结果作为本组织开发同类项目的基准（Benchmark）。

（4） 完成 DDP 二期规模估算和应用基准指导二期项目工期与计划及资源计划的编排。首先，使用 NESMA 功能点分析法（FPA）对 DDP 二期项目规模进行估算。其次，通过上述得到的基准估算项目工期后，完成工期的拟定及资源计划和计划的编排。

（5）完成通过挣值分析监督、控制计划值与实际值的偏差，并修正基准。首先，对挣值分析相关理论和相关参数进行研究和收集。其次,通过此方法在项目实施过程中实时分析、监督、控制计划值与实际值的偏差，并对基准进行修正。最终得到适用于本组织的准确的生产率基准，为本组织实现高效的项目管理提供了依据。

9.2 展望

软件成本估算作为项目管理关键因素之一，近年来逐渐受到学者的广泛关注和研究，但

其实就企业应用现状来讲，重视程度和应用方法还存在很大的局限性。首先，对于企业来讲，现有软件研发体系并没有形成一套完备的项目成本估算流程，并且可以合理高效地使用成熟的软件成本估算技术提高项目成本估算的准确性，从而更好地辅助决策。二是估算方法单一且过度依赖人员主观判断。现有估算大多利用经验值，并没有准确的基准或者规范可参照，专家对于软件成本估算专业知识的掌握程度等都将对估算结果产生较大影响。

所以，对于企业现有软件成本估算还有很多需要深入挖掘。现有估算方法并不需要全然舍弃，针对现状，它也是一种很好的解决方案，可以有效地进行粗略的成本估算，并且通过目前众多项目的检验。但是随着对于软件成本估算技术的深入研究，如何更好的利用成熟的模型进行估算，更加合理的在项目之初估算和规划项目，提高项目的成功率，降低因估算误差产生的项目延期或失败比例，应该是重点去研究的。

未来，也将在企业具体工作中继续结合真实项目探究软件成本估算理论和相关实际应用，思考如何更好更高效地利用软件成本估算实现高效地项目管理。

参考文献

[1] Dewi R S, Subriadi A P, Sholiq. A Modification Complexity Factor in Function Points Method for Software Cost Estimation Towards Service Application[J]. Procedia Computer Science, 2017, 124: 415-422.

[2] Shane Hastie[EB/OL]. https://www.infoq.com/articles/standish-chaos-2015.html, 2015.

[3] Boehm B.W. Software Cost Estimation Meets Software Diversity[J]. International Conference on Software Engineering Companion. IEEE, 2017:495-496.

[4] 李明树，何梅，杨达等. 软件成本估算方法及应用[J]. 软件学报，2007，04(18):775-795.

[5] Boehm B.W. Valerdi R.Achievements and challenges in cocomo-based software resource estimation[J]. IEEE software, 2008, 25(5): 74-83.

[6] 甘早斌. 软件开发成本估算技术综述[J]. 计算机工程与科学，2005 年第 6 期.

[7] 张海藩编著. 软件工程导论[M]. 北京：清华大学出版社，1998 年.

[8] 郑人杰主编. 软件工程-初级[M]. 北京：清华大学出版社，1999 年.

[9] Boehm W. Boehm. Software Engineering Economics[M]. Prentice Hall, 1991 年.

[10] 王平，丁浩芳，李韬. 结构型软件成本估算模型的研究与改进[J]. 软件技术与数据库，2002 年 12 月.

[11] B.W. Bohem, C. Abts, W.A.Brown, S.Chulani, B.A. Clark, E. Horowitz, R.Madachy, D.Reifer, and B.Steece, COCOMO II model definition, in Software Cost Estimation with COCOMO II , Prentice Hall, Englewood Cliffs, NJ, 2001, pp. 12-59.

[12] Boehm B. Abts C, Chulani S.Software Development Cost Esitmation Approaches – A survey[J]. Annals of Software Engineering, 2000, 10(1-4):177-205.

[13] B.W.Boehm, The goals of software engineering, in Software Engineering, Prentice Hall, Englewood Cliffs, New Jersey, 1981, pp. 14-25.

[14] Xu Z, Khoshgoftaar T M. Identification of fuzzy models of software cost estimation[J]. Fuzzy Sets & Systems, 2004, 145(1):141-163.

[15] Keung J W, Kitchenham B A, Jeffery D R. Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation[M]. IEEE Press, 2008.

[16] Li Y F, Xie M, Goh T N. A study of mutual information based feature selection for ease based reasoning in software cost estimation[M]. Pergamon Press, Inc.2009.

[17] Gharehchopogh F S, Rezaii R, Arasteh B. A new approach by using Tabu search and genetic algorithms in Software Cost estimation[C]International Conference on Application of Information and Communication Technologies. IEEE, 2015:113-117.

[18] 吴登生，李建平，孙晓蕾. 基于加权案例推理模型族的软件成本 SVR 组合估算[J]. 管

理工程学报，2015, (02):210-216.

[19] 周启超. BP 算法改进及在软件成本估算中的应用[J]. 计算机技术与发展, 2016, (2):195-198.

[20] Boehm B.W. Software Engineering Economics. Englewood Cliffs: Prentice Hall, 1981.

[21] Pfleeger SL. Software Cost Estimation and Sizing Methods: Issues, and Guidelines. Santa Monica: Rand Corp., 2005.

[22] Boehm B.W. Valerdi R. Achievements and Challenges in software resource estimation. Technical Report, No.USC-CSE-2005-513,2005. http://sunset.usc.edu/publications/TECHRPTS/2005/usccse2005-513.pdf.

[23] Kitchenham BA, Taylor NR. Software project development cost estimation approaches – A survey. Annals of Software Engineering, 2000,10(1-4):177-205.

[24] Kemerer CF. An empirical validation of software cost estimation models. Communication of the ACM, 1987,30(5):417-429.

[25] Navlakha JK. Choosing a software cost estimation model for your organization: A case study. Information and Management, 1990, 18(5):255-261.

[26] Srinivasan K, Fisher D. Machine learning approaches to estimation software development effort. IEEE Trans. On Software Engineering, 1995,21(2):126-137.

[27] Shepperd M, Cartwright M. Predicting with sparse data. IEEE Trans. on Software Engineering, 2001,27(11):987-998.

[28] McGibbon T. Modern cost estimation tools and modern empirical cost and schedule life cycle processes: COCOMO 2.0. Annals of Software Engineering, 2005,18(4):20-25.

[29] Boehm B.W., Royce W. Ada COCOMO and the Ada process model. In: Proc. Of the 5th Int'l COCOMO User's Group Meeting. Pittsburgh, 1989.

[30] Boehm B.W, Clark B, Horowitz E, Westland C, Cost models for future software life cycle processes: COCOMO 2.0, Annals of Software Engineering, 1995,1:57-94.

[31] Boehm B.W, Clark B, Chulani S. Calibration results of COCOMO II 1997. In: Proc of the 22nd Software Engineering Workshop. 1997.

[32] Chulani S, Boehm B.W., Steece B. Calibration software cost models using Bayesian analysis. Technical Report, USC-CSE-98-508, 1998.

[33] Chulani S, Boehm B.W., Steece B. Bayesian Analysis of empirical software engineering cost models. IEEE Trans. On Software Engineering, 1999,25(4):573-583.

[34] Jorgensen M. A review of studies on expert estimation of software development effort. Journal of Systems & Software, 2004, 70(1-2):37-60.

[35] Baird B. Managerial Decisions Under Uncertainty. New York: John Wiley & Sons, 1989.

[36] Shepperd M, Schofield C. Estimating software project effort using analogies. IEEE Trans. on Software Engineering, 1997,23(12):736-743.

[37] Delany SJ, Cunningham P, Wilke W. The limits of CBR in software project estimation. In: Gierl L, Lenz M, eds. Proc. of the 6th German Workshop on Case-Based-Reasoning. Berlin: Springer-Verlag, 1998.

[38] Shepperd M, Schofied C., Kitchenham B. Effort estimation using analogy. In: Proc.of the 18th Int'l Conf. on Software Engineering. IEEE CS Press, 1996. 170-178.

[39] Briand LC, Wieczorek I. Resource estimation in software engineering. In: Marcinak JJ, ed. Encyclopeia of software Engineering. New York: John Wiley & Sons, 2002. 1160-1196.

[40] Berry WD, Feldman S. Multiple Regression in Practice. Newbury Park: Sage Publications, 1985.

[41] 国家市场监督管理总局，中国国家标准化管理委员会. GB/T36964-2018 软件工程 软件开发成本度量规范，2018.12

[42] 中国软件行业协会软件造价分会，SSM institute. Nesma 功能点分析及应用指南（2.3 版）. 北京：中国标准出版社，2018.5

[43] 中华人民共和国国家质量监督检验检疫总局，中国国家标准化管理委员会. GB/T11457-2006 信息技术 软件工程术语，2006.3

谢辞

凡事过往，皆有序章。行文至此，意味着和同济一起走过的四度春秋，随着本次特殊时期开展的线上答辩，即将画下回忆满满的句号。

桃李不言，下自成蹊。本论文在黄杰老师的悉心指导和严格要求下完成，从最初课题的选取和拟定，到具体的写作过程，从论文初稿到最终定稿，共半年有余，这些无不凝聚着黄杰老师的心血和耐心的指导。毕业设计的选题方向也是从黄杰老师执教的《软件项目和过程管理》中受到启发，而在毕业设计写作期间，黄杰老师带给我的专业知识的指导和大量富有创造性、专业性的建议和指导也使我深受启发。在此向黄杰老师表示深深的感谢和崇高的敬意！

一开始选定软件成本估算的话题，除了不同于以往项目以开发为中心、如何结合具体实例，才能使本文研究更有实践意义，这些对我来说都是很大的挑战。为此，在短时间，我系统学习了软件成本估算的相关理论和模型使用方法，同时，因为大四企业实习岗位恰好为项目管理，接触到了一手的项目资料和项目开发管理全过程，这些经历，使我将软件成本估算技术得以落地，并赋予它实际的研究价值。在今后的工作中，自己也将结合更多实例，探究软件成本估算在项目管理过程中更高效和实际的应用。

岁月虽清浅，时光亦潋滟。其次感谢在大学遇见的每一位朋友曾给我在学习、生活上的鼓励和支持，在生活上给予的关心和帮助，愿有前程可奔赴，愿有岁月共回首。

父母之爱子，则为之计深远。借此机会特别感谢我的父母，求学十余载，感谢父母的无条件的支持和付出，我也会带着你们的期望和祝福走向接下来的人生。

大学毕业，是结束也是全新的开始。四年光阴，有遗憾也有满满收获，我会带着学到的知识和收获的经验走向工作岗位，未来即将面临为真实社会生活可能会更加残酷，但是我会用努力充实自己，积极迎接挑战。在此，再次感谢大学四年遇到的每一位老师教授的宝贵知识和每一位一起成长小伙伴，同时感谢永远支持我的父母，我也一定会继续努力，以更加饱满的精神和昂扬的姿态迎接未来。

2020 年是特殊的一年，纵然山河有恙，不敌世间盛情。借此机会，特向战役过程中的逆行者们致以最崇高的敬意，幸而有你，国家安好。

最后，在此衷心地愿各位老师、同学，一切顺利，前程似锦！

谢谢！

附录

装 订 线

1 “组织树”前端代码节选

```
const { TreeNode } = Tree;
const levelList = [10, 21, 22, 23];

const detailList = [{
      key: ‘orgNameCn’,
      valye: ‘组织名称’
 },{
      key: ‘supOrgCode’,
      value: ‘上级编码’
 },{
      key: ‘orgCode’,
      value: ‘编码’
 },{
      key: ‘manager1’,
      value: ‘负责人 1’
 },{
      key: ‘manager2’,
      value: ‘负责人 2’
 },{
      key: ‘modifyDatatime’,
      value: ‘最后修改时间
 },{
      key: ‘createDatatime’,
      value: ‘创建时间
 }];

@Form.create()
class Department extends PureComponent {
      constructor(props){
            super(props);
            this.state = {
```

```
            currentSelected: {},
            treeData: [
            {
                title: “商汤集团”,
                key: “10000000”,
                level: -1,
                isLeaf: false,
                createDatatime: “2019-10-25”,
                dataFlag: “A”,
                modifyTimestamp: 1571996154548,
                orgCode: “10000000”,
                orgNameCn: “商汤集团”,
                status: “A”,
                supOraCode: “-1”,
            }],
            openVisable: true,
        }
    }
}

// 当前组件渲染主函数（主入口）
rendor () {
    const { treeData, currentSelected, openVisable, spinning } = this.state;
    return (
        <Spin {…{delsy: 300, size: “small”}} spinning={spinning}>
            <Card bordered={false} style={{position: ‘relative’,minHeight: ‘80vh’}}>
                <div className={styles.flexContent}>
                    <div className={styles.leftSider}>
                    <Row><h3>组织</h3></Row>
                    <div style={{maxHeight:~65vh~,overflowY: ‘auto’ }}
                    className={styles.infoScocll}>
                        <Tree loadData={this.onLoadData} onLoad={this.defaultExpand}
                        DefaultSelectedKeys={[“10000000”]}
                        defaultExpandedKeys={[“10000000”]}

    OnSelect={this.selectTreenode}>{this.renderTreeNodes(treeData)}</Tree>
```

```
                    </div>
                </div>

                <div className={style.rightSider}>
                    <Row>
                        <Col><h3>详情</h3></Col>
                    </Row>
                    {
                        detailList.map(item => {
                            if (openVisiable) {
                                return(
                                    <Row type="flex" justify="start" style={{marginTop:
                                    ~${30}px~, wordbreak:'break-al}} key={item.key}>
                                        <Col span={8}
                                        style={{color:'rgba(0,0,0,0.85)'}}>
                                        {item.value}: </Col>
                                        <Col span={8}>{cirrentSelected[item.key]||''}
                                        </Col>
                                    </Row>
                                    )
                            }return ''
                        })
                    }
                </div>
            </Card>
        </Spin>
    }
```

2 “角色设置”前端代码节选

```
// 删除当前角色
delectCurrentRole = record => {
    const { dispatch } = this.props;
    dispatch({{
        type: 'role/deleteRole',
    }). then(()=>{
```

```
            This.handleFormReset()
        });
};

// dialog 主渲染入口
rendor() {
        const { roleAddModalVisible, modelTitle } = this.pros;
        const { currentStep, formVals } =this.state;
        return (
            <Modal
                Width={640}
                BodyStyle={{ padding: '32px 40px 48px' }}
                destroyOnClose
                title={modalTitle}
                visible={roleAddModalVisible}
                footer={this.rendorFooter(currentStep)}
                onCancel={() => this.handleAddModalVisible(false)}
                afterClose={() => this.handleAddModalVisible()}
            >
                <Steps stytle={{ marginBottom; 28 }} size="small" current={currentStep}>
                    <Step title= "基本信息"/>
                    <Step title= "配置菜单"/>
                    <Step title= "配置权限"/>
                    <Step title= "配置人员"/>
                </Steps>
                {this..rendorContent(currentStep, formVals)}
                </Modal>
            );
        }
}
```

3 “注册当前下游服务”前端代码节选

```
registerCurrentDownStream = (statusUrl,statusKey) => {
        const { selectedStaff } = this.pros
        const { registerLoding } = this.state;
```

```
        registerLoding[~${statusUrl}~] = true
        this.setState({
            registerLoding:{…registerLoding}
        })
        RegisterDownsystem({url:statussUrl, data:selectedStaff.staffCode}),then((res)=>{
            registerLoding[~${statusUrl}~] = false
            this.setState({
                registerLoding:{…registerLoding}
            })
            if (res.data){
                this.searchDownsystem(statusKey, “register)
            }
        }).catch();
};
```

4 “员工操作”前端代码节选

```
// 查询员工信息
*searchStaff({ payload, callback }, { call, put }) {
    const staffResponse = yield call(selectStaff, payload);
    yield put({
        type: ‘save’,
        payload: { staffResponse },
    });
    if (callback) callback();
}

// 员工离职操作
*leave({ payload, callback }, { call, put }) {
    const leaveResponse = yield call(leaveManage, payload);
    yield put({
        type: ‘save’,
        payload: { leaveResponse },
    });
    if  ( callback) callback();
}
```

5 “登陆界面”前端代码节选

```
*logout(_, {call, put }) {
    yield call(logoutApi);
    yield put({
        type: ‘changeLoginStatus’,
        payload: {
            statusCode: false,
            currentAuthority: ‘guest’,
        }
    });
    reloadAuthorized();
    const { redirect } = getPageQuery();
    // redirect
    if (window.location.pathname !== ‘/user/login’ && !redirect) {
        yield put(
            routerRedux.replace({
                pathname: ‘/user/login’
            })
        );
    }
}
```

装 订 线