大力作序推荐本书

Jim Highsmith Cutter Consortium公司敏捷业务主管

Gabrielle Benefield Yahoo! 公司敏捷产品开发部总监

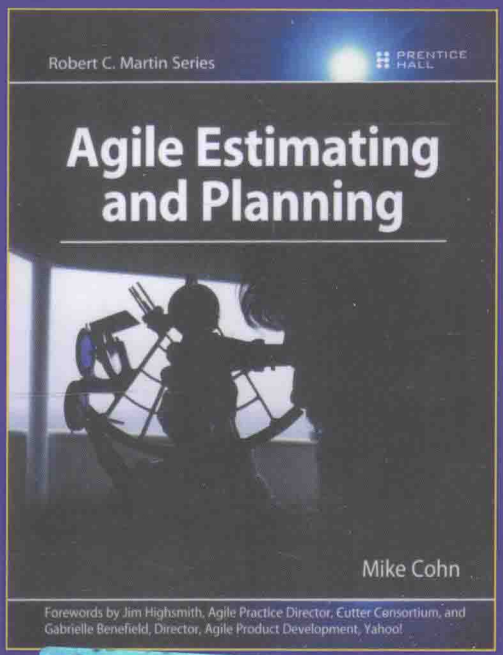
PEARSON

A MIKE COHN SIGNATURE BOOK

敏捷软件开发实践

估算与计划

Agile Estimating and Planning



[美] Mike Cohn 著

金 明 译

PEARSON
ALWAYS LEARNING ALWAYS LEARNING ALWAYS LEARNING

清华大学出版社

敏捷软件开发实践

估算与计划

[美] Mike Cohn 著
金 明 译

清华大学出版社
北 京

图书在版编目(CIP)数据

敏捷软件开发实践 估算与计划 /(美) 科恩(Cohn, M.) 著；金明 译. —北京：清华大学出版社，2016
书名原文：Agile Estimating and Planning
ISBN 978-7-302-42393-5

Ⅰ. ①敏… Ⅱ. ①科… ②金… Ⅲ. ①软件开发 Ⅳ. ①TP311.52

中国版本图书馆 CIP 数据核字(2015)第 296368 号

责任编辑：王　军　韩宏志
装帧设计：牛静敏
责任校对：成凤进
责任印制：王静怡
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：16　　字　　数：389 千字
版　　次：2016 年 3 月第 1 版　　印　　次：2016 年 3 月第 1 次印刷
印　　数：1~3000
定　　价：49.80 元

产品编号：066491-01

译 者 序

对于软件人士，持续交付无异于是皇冠上最耀眼的明珠——IT 提供持续交付业务需求的能力和产出，满足业务的价值或驱动业务的创新。长期以来，如何达成持续交付、业务与 IT 对齐，一直是各路前辈同道孜孜以求的目标。

自第一台计算机“ENIAC”诞生以来，软件前辈们筚路褴褛，从无数的软件项目中摸索出各种方法：瀑布方法、增量开发……乃至 CMMI 模型。然而，随着软件从科研殿堂进入大众市场，软件从“火箭”算法往“消费者”业务转移，这些强调预先确定、精确执行、严格控制的软件方法变得越来越笨重。Cloud、Mobile、VR……软件产品和商业模式日新月异，传统软件方法既无法控制业务需求的变更，又抑制阻碍了反馈的周期，伴随而来的是不可避免的延期和失败。这种情况下，敏捷的出现，不啻于给“沉重”的软件界吹来一缕清新的风，让软件不能承受之“重”又变轻盈。

我于 2008 年从传统的软件公司来到 ThoughtWorks，为国内外的客户交付了各类软件，也帮助很多公司导入和实施了“端到端”的敏捷。这些年中，很多公司从了解“敏捷”到尝试“敏捷”，引入了迭代、看板与站会等实践，也取得了不错的效果。然而，很多公司只是停留在这些项目日常活动的“敏捷”上面，一头一尾的需求计划与交付发布依旧倚赖着传统的瀑布式项目计划与 ITIL 运维管理——这类“敏捷”被戏谑性地称为“Water-Scrum-Fall”。敏而不捷，这不能不说是一种遗憾。“凡事预则立，不预则废”，软件项目的计划活动无疑更迫切地需要敏捷！

软件项目计划三要素：范围、时间和资源——要做多大范围、需要多长时间、投入多少资源，以及如何统筹安排进度计划。无论敏捷与否，面对决策者、干系人、投资人，每个项目都需要回答这些问题。软件的“估算”与“计划”，作为团队集思广益、传递知识、建立信心的关键活动，对于传统或敏捷方法虽然出发点一致，然而在方法和效果上千差万别。

譬如敏捷估算与计划更关注纵向的特性，而非横向的活动。软件需求被拆分为独立的特性，估算则使用故事点。根据“大小 / 速度＝时间”以及“故事点 / 实际时间＝速度”的关系，敏捷项目计划可以灵活地结合时间、速度、大小这些变量来计划和调整。

又如敏捷估算与计划更强调优先级驱动和“刚刚好”的决策。产品愿景按照优先级和速度梳理出分层的发布计划或者迭代计划，再按照优先级进入日常的开发。往往只有最近一次新迭代的需求会被详细地分析和估算，其他迭代都只需要保证刚刚好的估算和计划范围。

再如敏捷估算与计划更强调集体合作和响应变化。整个项目团队一起参加软件的估算和计划，从各自的角度充分表达沟通各自的理解和分析，直到达成一致。敏捷计划会透明地展示给所有干系人，并“时时勤拂拭，莫使惹尘埃”，根据实际情况及时响应更新进度计划。

……

其实，以我蠡见，敏捷估算与计划之所以优于传统方法，全在于“敏捷”二字：以人为本、杜绝浪费、响应变化，以及这些原则背后的“沟通、勇气、反馈、透明”价值观。同样，在发布运维领域，植根于敏捷的 DevOps 运动在当下也是如火如荼，给沉重的发布运维也注入了全新的活力。在敏捷软件开发活动之外，拥抱敏捷估算与计划以及 DevOps，软件的全流程做到真正“端到端”的敏捷化，又何愁软件不持续交付？与软件诸位同道共勉！

古人以汉书下酒，本书的很多洞见至今读来依旧觉得豁然开朗、如沐春风。虽然黄钟大吕，作者却娓娓道来，相信读者一定能获得深刻的体会。当然，随着时代的发展，本书中的很多观点和实践得到了进一步的发扬，比如看板方法、精益创业和商业模式画布等。也希望你在阅读时多结合新兴的实践一起思考，追根溯源，相信这样你会更有所获。

最后，我要把本书献给我的妻子徐婷婷，以及我即将出生的宝贝。正是你们的支持和鼓励，让我可以专心地翻译，反复锤炼词句，呈现给读者。我还要感谢清华大学出版社的李阳等编辑老师，没有你们认真的审校和排版，这本书不会如此完美地展现在读者面前。

编 首 语

在敏捷开发的世界中，我总会听到这样一些相同的问题：

- 我应该如何为大型开发小组做计划？
- 我应该采用多长的迭代周期？
- 我应如何向管理层作进度汇报？
- 我如何确定用户故事的优先级？
- 我如何得到有关项目的全景？

这些问题，以及很多其他的问题，都在本书中得到了巧妙的回答。无论你是项目经理、项目领导、开发人员还是指导者，本书都为你提供了必需的工具，可以用来对任何规模的敏捷项目进行估算、计划和管理。

我已经认识 Mike Cohn 多年了。在敏捷宣言签署之后不久我就遇到了他。Mike 带着独特的热情和活力加入了敏捷联盟。无论他参与哪一个项目，都会完成那个项目，并且能够很好地完成它。他非常引人注目，对大家都有所帮助。他很快就成为新生的敏捷联盟中不可或缺的人物。

现在，他将同样的能力、透彻性和活力注入本书中。本书展现出他的这些品质，并且以一流的水准(完全可行的建议)展现出来。

这不是一本关于抽象理论的书。作为读者，你不需要花费大量的时间，好像在 30 000 英尺高空的云雾中一样考虑问题。与之相反，Mike 提供了具体的实践、方法、工具、图表、公式，而且最为重要的是，他还提供了恰到好处的建议。本书是一本有关“如何进行”估算和计划的手册。

本书从头至尾点缀了大量的轶事来说明 Mike 在使用他所说明的方法和工具时的经历。他会告诉你这些方法和工具何时有效，何时无效。他会告诉你什么地方会出错，以及如何做才是正确的。他不会给你什么承诺，提供什么“银弹”，或者做出什么保证。但是他为你提供了许多来之不易的经验。

曾经有许多书涉及过有关敏捷估算和计划的问题。确实有几本书是把它作为主题来写的。但是没有一本能够与本书的深度和实用性相媲美。本书完整地、有效地覆盖了这一主题，以至于我认为它应该被看作是权威之作。

好了，我知道我看起来过于激动，但我确实非常兴奋。我兴奋的原因是许多长久以来就存在的问题终于被足够称职的人回答了。我兴奋的原因是当客户提出一些困难的问题时，我终于可以向他们提供一个工具。我兴奋的原因是本书已经写好，而你即将读到它。

我很高兴并且荣幸地在我的丛书中加入本书。我认为它是一本成功的书。

系列书籍编辑 Robert C. Martin

序 1

在首次阅读一本书或者手稿时，我总会问自己同一个问题：“作者为这一主题的发展现状增添了哪些内容？”对 Mike 的这本书，答案有两个方面：他的书增加了关于“如何”进行估算和计划的知识，还增加了关于“为什么”特定的实践很重要这方面的知识。

敏捷计划是具有欺骗性的。在某个层面上，它相当容易——建立一些故事卡片，确定它们的优先级，把它们分配到不同的发布迭代周期，然后添加其他的细节来获得下一轮的迭代计划。你花几个小时就可以让一个小组学会计划的基本内容，然后他们再花几个小时就可以(对一个小项目)提出一个可被接受的计划。Mike 的书可以在很大程度上帮助小组从制订可被接受的计划转变成制订优秀的计划。在这里，我的用词是非常谨慎的——我并没有说完美的计划，因为正如 Mike 在本书中指出的，在一个(足够)优秀的计划和一个完美的计划之间的差异可能抵不上为之所须付出的精力。

关于 Mike 的这本书，我早期的想法和敏捷计划本身的概念有关。我常常因为缺乏对敏捷计划的理解而感到困惑或者是悲哀。我们常听到一些批评，例如“敏捷项目开发小组不做计划”，或者是“敏捷开发小组不会对日期和功能做出承诺”。甚至 *Balancing Agility and Discipline: A Guide for the Perplexed*(Addison-Wesley，2004 年出版)一书的作者 Barry Boehm 和 Richard Turner 在他们的书中讨论“计划驱动与敏捷方法”时也弄错了。实际上，Boehm 和 Turner 的观点是对的，但是表述不当。他们使用计划驱动来表示“更看重对预测的平衡而不是对预测的调整适应”，而敏捷方法意味着相反的做法。他们的问题在于“计划驱动”对“敏捷”这一比对传递了完全错误的信息——敏捷开发小组不做计划。这一理解实际上是完全错误的。《敏捷软件开发实践　估算与计划》传递了正确的信息——计划对任何敏捷开发项目都是不可缺少的组成部分。本书中有大量的关于计划为什么如此重要以及如何有效进行计划的看法。

首先，敏捷开发小组会进行大量的计划活动，但这些活动被更为均衡地分布于项目的整个开发过程。其次，敏捷开发小组会直接面对不确定性这一被许多非敏捷开发小组所忽视的关键因素。计划重要吗？——当然重要。随着知识的获取和不确定性的降低调整计划重要吗？——当然重要。我看到过太多的公司，一开始做出许多不切实际的早期承诺然后又无法实现它们，被看作是可被接受的；而那些试图做出更为现实的承诺(充分理解不确定性的影响)的人，则被看作“不能适应需要”，或“缺乏团队精神”。在这些公司中，似乎无法交付是可被接受的，而不能承诺(即使是对不切实际的目标)则是不可接受的。就像 Mike

巧妙指出的那样，敏捷开发方法强调实际交付价值而不是做出一些非凡的但是无法实现的计划和承诺。敏捷开发人员通常会说，我们会给你一个基于当前所了解的内容做出的计划；我们会随着开发过程中获得的新信息对项目和计划做出调整；我们希望你能够理解你所要求的两个目标——获得适应变化的应用环境的灵活性，与绝对地遵守原始计划——是相互矛盾的。《敏捷软件开发实践　估算与计划》对以上的每个论断都进行了说明。

让我们回到对不确定性进行管理这一关键问题上。Mike 完成了一项伟大的工作，分析了敏捷开发过程如何同时减少目标不确定性(我们到底要构建什么)和方法不确定性(我们如何构建它)。许多传统的计划人员没有理解一个关键概念——不确定性是不能被“计划”的。计划是基于我们在某个特定时间点上所知道的东西做出的，而不确定性则是对我们所不知道的事情——对目标或者方法——的另一种表述。对大部分不确定性(缺乏知识)而言，获取知识、减少不确定性的唯一办法是通过执行——做一些事情、构建一些东西或是模拟一些东西——然后获得反馈。许多项目管理方法是“计划、计划、计划-执行”。而敏捷开发方法是“计划-执行-调整”、“计划-执行-调整”。一个项目的不确定性越高，敏捷开发方法对取得成功就越是至关重要。

我喜欢用该书中的第 4 章和第 5 章来说明有关“如何做”和“为什么做”的问题。这两章详细阐述了如何估算故事点和理想人天，并对两者的优缺点分别进行了解释。虽然这两种方法我都和客户一起使用过，但 Mike 的话让我对估算故事点的优点认识得更为清晰。我认识到故事点是朝向简单性的进化的一部分。软件开发公司长期以来都在寻求“这个软件规模有多大”这一问题的答案。建筑师可以根据平方尺度对建筑工程做出合理的估算。不同建筑师做出的估算可能有所差异，但规模的大小是固定的(虽然收尾工作、材料特性和其他一些因素可能影响到估算结果)，保持不变。软件开发人员一直希望也能找到一个这样的度量方法。

在软件开发中，我们最初使用代码行(line-of-code)来度量产品的规模(这一方法当前仍在一定程度上被使用)。对大部分日常计划来说，有一些原因导致代码行的用途有限，其中之一就是由于对其进行估算时所需进行的先期工作太多。接下来就是功能点(function point)方法(以及一些类似的思路)。虽然功能点消除了代码行方法的一些问题，但是其计算仍然需要相当数量的先期工作(你需要对输入、输出和文件等进行估算)。功能点方法的复杂性不可避免地影响了对它的广泛应用。我的看法是随着计算方法复杂程度的上升——只要观察一下国际功能点用户组(International Function Point User Group，IFPUG)网站就可以了解到它有多复杂——普通人对它的使用就会减少。

然而，对软件项目规模进行估算的需求却从未消失。这两种曾经被采用的度量方法存在两个问题——它们的计算过于复杂，而且都是基于瀑布式的开发方法。我们仍然需要一种规模度量方法，我们需要一种易于计算，而且不必完成所有需求分析和设计阶段就可以实现的度量方法。

故事点方法与代码行或功能点方法的两个关键不同之处在于它更容易计算而且可以更早计算。为什么更容易计算？因为它是基于相对规模而不是绝对规模。为什么可以更早计算？因为它是基于相对规模而不是绝对规模。就像 Mike 指出的那样，进行故事点估算只需要大家坐在一起讨论用户故事(获取共同知识)，推测相对的故事规模。与对绝对规模进行估算相比，对相对规模进行估算要快得多。此外，经过几次规模估算和交付的迭代后，

一个小组的估算准确性会显著提高。Mike 对故事点和理想人天估算方法的“如何”和“为什么”的说明提供了对这一主题敏锐而深刻的理解。

第 9 章和第 11 章是体现 Mike 对这一问题理解的透彻性的另一表现。这两章是关于确定用户故事的优先级的。Mike 不满足于只告诉我们首先处理具有最高价值的用户故事，事实上他钻研了价值的各个关键因素：经济收益、代价、创新/知识，以及风险。他对价值的这些方面进行了仔细的定义(甚至包括了一个关于净现值(Net Present Value)、内部回报率(Internal Rate of Return)和其他金融分析工具的初级读本)，然后提供了使用这些价值因素对决策进行衡量的不同方案，每个方案分别具有不同的简单性。

刚接触敏捷开发的人往往认为如果遵循了特定方法学的 12 个、19 个或 8 个实践方法，那么就是在采用敏捷开发、极限开发、Crystal Clear 开发，或者是别的什么开发方法。但实际上，只有在充分了解这些实践方法并能根据你所处的特定环境调整它们时，才是真正采用敏捷开发、极限开发或者别的开发方法。不断地学习和调整是敏捷开发的核心。Mike 在本书中做得最好的就是为我们提供了许多看法和经验，来帮助我们的敏捷估算和计划的实践达到这一更高的层次。Mike 深入地告诉我们“如何做”——例如那些有关故事点和理想人天估算的材料。Mike 深入地告诉我们“为什么做”——例如那些关于故事点和理想人天估算的优缺点。虽然他通常给我们一些个人的建议(他更倾向于故事点)，他也为我们提供了足够的信息，让我们可以自信地对各种实践方法进行裁剪以适应特定的环境需要。

所以，最后，Mike 对此方面技术发展所做出的显著贡献在于——他帮助我们在一个新的知识和经验深度上对估算和计划进行思考(如何做)，然后帮助我们决策如何利用这些新的知识来调整实践方式以适应新的、独特的或者仅仅是特定的环境(为什么做)。在我通常会推荐给客户的 6 本书中，有 2 本是 Mike 写的。对于那些希望了解敏捷项目管理在此方面发展现状的人，本书是我推荐给他们的“必读”书目。

Jim Highsmith
Cutter Consortium 公司敏捷业务主管
于 Arizona Flagstaff

序 2

“除了一些小型项目以外，敏捷开发在 Yahoo！公司根本就行不通，原因是那样的话开发小组就不会做任何计划，小组成员也无法对自己的工作进行估算。需要有人告诉他们应该做些什么。”

这是人们的原话，从我在 Yahoo！公司开始指导采用敏捷开发方法以来就反复听到这样的话。那些不理解敏捷开发概念的人认为敏捷开发不过是消灭掉所有的文档和计划，让开发小组到处乱逛的许可。事实与真相的距离不会比这种看法更远。

“小组不做任何计划。”说这种话的人忘记了敏捷开发小组每隔一周就会花半天的时间来列出任务列表，表上是为了在 2 周的时间结束时他们能够交付一些对用户有价值的功能所需要完成的工作。开发小组让计划活动扩展到了项目开发的整个过程，而不只是在一开始就先期完成所有计划，结果常被看作缺乏计划。事实并非如此。在 Yahoo！公司，与传统开发小组相比，敏捷开发小组所创建的产品让我们的产品经理更为满意。

“小组成员无法对自己的工作进行评估，需要有人告诉他们应该做些什么。”这是一个非常典型的错觉。从商务角度来看，让产品经理或者项目经理具有圣人一样的能力，预计其他在其所从事的行业是专家的人到底能做些什么，简直就是自杀行为。通常，这是一种在被要求交付不现实的目标时，用来向销售人员做出承诺的办法。然后，开发小组的成员被迫连轴转或投机取巧。难怪在我们的行业中大家总是显得筋疲力尽而且士气低落了。

在人们向我提出的问题中，最多的疑问集中在估算和计划等方面，新小组尤其如此。获得一个简单的方法为整个项目过程而不仅仅是一个迭代周期进行计划，其价值是无法估算的。产品经理需要注意满足财务目标并获得一个可预测的发布计划。开发小组仍然可以保持灵活性，并根据需要改变进程，但遵循一个路线图是非常重要的。如果走错了方向，仅仅加快速度是不够的。如果想在公司中成功地实现敏捷开发，学会如何进行估算和计划是最重要的内容之一。

Mike 的估算和计划课是我们在 Yahoo！的敏捷开发课程中最受欢迎的。它为开发小组提供了技巧与工具，让他们可以为得到最佳的结果而进行恰到好处的计划。如果遵循 Mike 的建议，真会起作用吗？是的。敏捷开发在 Yahoo！公司取得的成功是不可思议的。我看到过小组成员一从 Mike 的课上回来就把他的建议投入应用。我们可以更快地向市场上推出产品，而开发小组真实地爱上了敏捷开发方法。

为什么敏捷估算和计划方法比传统方法更有效？因为它们专注于交付价值，在销售小

组与项目小组间建立信任。让所有的事保持高度透明，让销售人员从一开始就了解发生的所有变化，意味着业务人员可以迅速调整以做出最佳的决策。在我工作的上一家公司，我亲眼看着我们从只有一张非常模糊的路线图而无法交付产品的长期混乱状态，转变成了终于可以给我们能够交付的产品签字的可预测状态。业务部门说也许他们不总是那么喜欢我们的答案(他们第二天总会想要些新东西)，但至少他们可以相信我们的答案，而不用因为觉得总是被欺骗而沮丧了。

本书真实地反映了事实。它并不会告诉你如何使你的估算绝对准确。那是不可能实现的，只能是白白浪费精力。《敏捷软件开发实践　估算与计划》并未试图给你一个完美的填空式的模板，而是让你思考和学习如何分析问题、解决问题。没有两个项目、产品或者公司是完全一样的，所以学会思考方式和基本原则更为重要。Mike 在本书中生动地再现了他的亲身经历和个性特点。本书是真实的、诚实的。毫无疑问，本书应该排在你的阅读书目的最上面。

Gabrielle Benefield
Yahoo!公司敏捷产品开发部主任

前言

本书本来被命名为《估算与计划敏捷项目》。不过，书名最终确定为《敏捷软件开发实践　估算与计划》。两者的差异似乎微不足道，但实际上并非如此。现在的书名明确了*估算*和*计划*过程本身就应该是敏捷的。不采用敏捷*估算*和*计划*，项目就不可能是敏捷的。

本书的大部分内容是关于*计划*，我把它看作是用来回答“我们要构建什么以及何时完成？”这一问题。但是，要回答关于*计划*的问题，我们还必须解决关于*估算*(“它的规模有多大？”)和进度安排(“什么时候能完成？”和“我到那时能得到多少？”)的问题。

本书由 7 个部分共 23 章组成。每一章的结尾都有对该章重点的小结和一组讨论题。由于*估算*和*计划*应该是整个团队的工作，因此我希望对本书的阅读方式是团队成员每周聚在一起，对看过的内容以及每章结尾的讨论题进行讨论。由于敏捷软件开发在全世界都受到欢迎，因此尽可能避免让本书显得过分以美国为中心。为了达到这一目的，我使用了一种通用的货币单位，将金额写作 500“币”，而不是 500 美元或者 500 欧元。

本书的第 I 部分介绍了*计划*为什么重要、我们常会遇到的问题，以及敏捷方法的目标。第 1 章是本书的起始，介绍了*计划*的目的、一个优秀的计划由哪些部分组成，以及什么才是敏捷*计划*。第 2 章中介绍了为什么传统*估算*和*计划*方法是导致结果难以令人满意的最重要原因。最后，第 3 章首先简要地重述了敏捷的含义，然后概括介绍了本书其他部分在不同层次上所采取的敏捷*估算*和*计划*方法。

本书的第 II 部分介绍了*估算*的一个主要原则，即对大小和时间长度的估算应该相互独立。第 4 和第 5 章介绍了两个适用于对要开发的特性大小进行*估算*的计算单位：故事点和理想人天。第 6 章讲述了采用故事点和理想人天进行*估算*的技巧，并包括了对计划扑克的介绍。第 7 章讲述了何时以及如何进行重新估算。第 8 章则提供了有关如何在故事点和理想人天之间进行选择的建议。

第 III 部分“为价值进行*计划*”提供的建议告诉项目团队如何确认他们正在构建尽可能好的产品。第 9 章介绍了在确定特性的优先级时需要综合考虑的一些因素。第 10 章展示了对特性或特性集合的财务回报进行建模的一种方法，以及如何对财务回报进行比较以便开发团队首先处理最具价值的特性。第 11 章主要讲述有关如何评估产品用户对各个特性的需求程度并确定其优先级的建议。第 12 章对本部分进行总结，给出了一些建议，帮助将大的特性分解成更小的、更易管理的特性。

在第 IV 部分中，我们将注意力转移到了有关项目时间进度安排的方面。第 13 章首先

讨论了对一个相对简单的、单一开发团队的项目安排进度表时所涉及的步骤。第 14 章讨论了如何制定迭代的计划。第 15 章和第 16 章讨论了如何选择合适的迭代周期长度以及如何*估算*开发团队的初始速率。第 17 章详述了如何安排一个具有很高不确定性的或是在时间进度上很可能出错的项目的进度表。第 18 章是这一部分的结尾，讲述了对由多个团队共同开发的项目进行*估算*和*计划*所必需的其他步骤。

一旦建立了计划，团队就必须和整个公司的其他部门进行交流，并根据计划进度对开发团队的进度进行监督。这是第 V 部分的 3 章的主要内容。第 19 章主要关注对发布计划进行监督，而第 20 章关注对迭代计划进行监督。这一部分的最后一章，第 21 章主要解决如何就计划及其进度进行沟通。

第 22 章是第 VI 部分唯一的一章。这一章与第 2 章讲述的“为什么传统方法会失败”相对照，讨论了为什么敏捷*估算*和*计划方法*会有效。

第 VII 部分是全书的最后一部分，也只有一章。第 23 章是一个展开的案例分析，以小说的形式重述了本书的重点。

目录

第Ⅰ部分　问题与目标

第 1 章　*计划*的目的……3
1.1　为何要进行估算和*计划*……4
1.1.1　减少风险……5
1.1.2　降低不确定性……5
1.1.3　提供更好的决策支持……5
1.1.4　建立信任……6
1.1.5　传递信息……6
1.2　优秀的计划是什么……7
1.3　敏捷*计划*是什么……7
1.4　小结……8
1.5　讨论题……8

第 2 章　*计划*失败的原因……9
2.1　基于活动而不是基于特性进行*计划*……9
2.1.1　活动不会提前完成……10
2.1.2　延误沿着计划表向下传递……10
2.1.3　活动不是互相独立的……11
2.2　多任务处理导致更多的延迟……12
2.3　不按优先级开发特性……13
2.4　忽视了不确定性……13
2.5　把估算当作承诺……14
2.6　小结……14
2.7　讨论题……15

第 3 章　敏捷方法……17
3.1　项目的敏捷开发方法……18
3.1.1　敏捷团队作为一个整体工作……18
3.1.2　敏捷团队按短迭代周期工作……19
3.1.3　敏捷团队每次迭代交付一些成果……19
3.1.4　敏捷团队关注业务优先级……20
3.1.5　敏捷团队进行检查和调整……21
3.2　敏捷*计划*方法……21
3.2.1　*计划*的不同层次……22
3.2.2　满意条件……23
3.3　小结……25
3.4　讨论题……25

第Ⅱ部分　估 算 大 小

第 4 章　使用故事点估算大小……29
4.1　故事点是相对的……29
4.2　速度……31
4.3　小结……33
4.4　讨论题……33

第 5 章　使用理想人天进行估算……35
5.1　理想时间和软件开发……36
5.2　以理想人天作为对大小的度量……37
5.3　给出一个而不是多个估算值……37

5.4 小结 ····· 38
5.5 讨论题 ····· 38

第 6 章 估算方法 ····· 39
6.1 共同估算 ····· 40
6.2 估算的尺度 ····· 41
6.3 得到估算值的方法 ····· 42
6.3.1 专家意见 ····· 43
6.3.2 类比 ····· 43
6.3.3 分解 ····· 43
6.4 *计划*扑克 ····· 44
6.4.1 更小规模的会议 ····· 45
6.4.2 何时玩计划扑克 ····· 45
6.5 为什么*计划*扑克会有效 ····· 46
6.6 小结 ····· 46
6.7 讨论题 ····· 47

第 7 章 重估 ····· 49
7.1 SwimStats Web 站点 ····· 49
7.2 不进行重估的情况 ····· 50
7.3 需要重估的情况 ····· 51
7.3.1 场景 1：不进行重估 ····· 52
7.3.2 场景 2：重估完成的故事 ····· 52
7.3.3 场景 3：相对大小改变时进行重估 ····· 52
7.4 重估部分完成的故事 ····· 52
7.5 重估的目的 ····· 53
7.6 小结 ····· 53
7.7 讨论题 ····· 54

第 8 章 在故事点和理想人天之间进行选择 ····· 55
8.1 有利于故事点的考虑因素 ····· 55
8.1.1 故事点有助于驱动跨功能的行为 ····· 55
8.1.2 故事点估算不会过期 ····· 56
8.1.3 故事点是对大小的纯粹度量 ····· 56
8.1.4 故事点估算通常更快 ····· 57
8.1.5 我的理想人天不等于你的理想人天 ····· 57
8.2 有利于理想人天的考虑因素 ····· 58
8.2.1 理想人天在团队以外更容易解释 ····· 58
8.2.2 理想人天估算更容易开始 ····· 58
8.2.3 理想人天便于预测速度 ····· 58
8.3 建议 ····· 58
8.4 小结 ····· 59
8.5 讨论题 ····· 59

第III部分 为价值制定计划

第 9 章 确定主题的优先级 ····· 63
9.1 确定优先级时的因素 ····· 63
9.1.1 价值 ····· 64
9.1.2 成本 ····· 64
9.1.3 新知识 ····· 65
9.1.4 风险 ····· 66
9.2 综合 4 个因素 ····· 68
9.3 一些例子 ····· 68
9.3.1 基础设施 ····· 68
9.3.2 用户界面设计 ····· 69
9.4 小结 ····· 69
9.5 讨论题 ····· 70

第 10 章 确定经济优先级 ····· 71
10.1 收入的来源 ····· 72
10.1.1 新收入 ····· 73
10.1.2 增量收入 ····· 73
10.1.3 留存收入 ····· 73
10.1.4 操作效率 ····· 73
10.2 例子：WebPayRoll ····· 74
10.2.1 计算新收入 ····· 74
10.2.2 计算增量收入 ····· 75
10.2.3 计算留存收入 ····· 76
10.2.4 计算操作效率 ····· 76

10.2.5 估算开发成本 …… 77
10.2.6 整合 …… 78
10.3 经济指标 …… 78
10.3.1 金钱的时间价值 …… 79
10.3.2 净现值 …… 79
10.3.3 内部收益率 …… 80
10.3.4 投资回收期 …… 82
10.3.5 折现回收期 …… 83
10.4 对利润的比较 …… 83
10.5 小结 …… 84
10.6 讨论题 …… 84

第 11 章 确定渴望度优先级 …… 85
11.1 客户满意度的 Kano 模型 …… 85
11.2 相对权重：另一种方法 …… 89
11.3 小结 …… 91
11.4 讨论题 …… 91

第 12 章 分解用户故事 …… 93
12.1 何时分解用户故事 …… 93
12.2 按照数据边界分解 …… 94
12.3 按照操作边界分解 …… 95
12.4 去除横切考虑 …… 96
12.5 忽略满足性能限制 …… 97
12.6 分解具有混合优先级的用户故事 …… 97
12.7 不要把故事分解成任务 …… 97
12.8 避免相关变化的诱惑 …… 98
12.9 组合用户故事 …… 98
12.10 小结 …… 98
12.11 讨论题 …… 99

第Ⅳ部分 进 度 计 划

第 13 章 发布*计划*精粹 …… 103
13.1 发布计划 …… 103
13.1.1 确定满意条件 …… 105
13.1.2 估算用户故事 …… 105
13.1.3 选择迭代周期长度 …… 105
13.1.4 估算速度 …… 105
13.1.5 确定用户故事优先级 …… 105
13.1.6 选择用户故事和发布日期 …… 106
13.2 更新发布计划 …… 107
13.3 例子 …… 107
13.3.1 确定满意条件 …… 108
13.3.2 估算大小 …… 108
13.3.3 选择迭代周期长度 …… 108
13.3.4 估算速度 …… 109
13.3.5 确定用户故事优先级 …… 109
13.3.6 选择用户故事 …… 109
13.4 小结 …… 110
13.5 讨论题 …… 110

第 14 章 迭代*计划* …… 111
14.1 迭代*计划*时不分配任务 …… 113
14.2 迭代*计划*和发布*计划*的区别 …… 113
14.3 速度驱动的迭代*计划* …… 114
14.3.1 调整优先级 …… 115
14.3.2 确定目标速度 …… 116
14.3.3 确定迭代目标 …… 116
14.3.4 选择用户故事 …… 116
14.3.5 把用户故事分解成任务 …… 117
14.3.6 对任务进行估算 …… 119
14.4 承诺驱动的迭代*计划* …… 121
14.5 我的建议 …… 124
14.6 任务估算值和故事点的联系 …… 124
14.7 小结 …… 126
14.8 讨论题 …… 126

第 15 章 选择迭代长度 …… 127
15.1 选择迭代长度时考虑的因素 …… 127
15.1.1 发布的总时间长度 …… 127