Professional Scrum with Team Foundation Server 2010

Scrum敏捷开发高级教程

——使用Team Foundation Server 2010

[美] Steve Resnick
Aaron Bjork
朱永光
等著
译

清华大学出版社

Scrum敏捷开发高级教程

——使用Team Foundation Server 2010

[美] Steve Resnick
Aaron Bjork 等著

朱永光 译

清华大学出版社

北 京

Steve Resnick, Aaron Bjork, et al
Professional Scrum with Team Foundation Server 2010
EISBN：978-0-470-94333-5

北京市版权局著作权合同登记号 图字：01-2012-0979

图书在版编目(CIP)数据

Scrum敏捷开发高级教程——使用Team Foundation Server 2010 / (美)雷斯尼克(Resnick, S)，(美)比约克(Bjork, A.)等　著；朱永光　译. —北京：清华大学出版社，2013.1
书名原文：Professional Scrum with Team Foundation Server 2010
ISBN 978-7-302-30829-4

Ⅰ. ①S…　Ⅱ. ①雷…　②比…　③朱…　Ⅲ. ①软件开发—项目管理　Ⅳ. ①TP311.52

中国版本图书馆CIP数据核字(2012)第287874号

责任编辑：王　军　于　平
装帧设计：牛艳敏
责任校对：蔡　娟
责任印制：宋　林

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 刷 者：清华大学印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：18.5　　**字　　数**：394千字
版　　次：2013年1月第1版　　**印　　次**：2013年1月第1次印刷
印　　数：1～3000
定　　价：58.00元

产品编号：042226-01

目　录

第1章

软件产品的推出

本章内容：

- 了解推出软件的步骤：构思愿景、深入认识、筹备资源、规划进度和实现特性
- 了解3种软件开发管理方法学：Scrum、MSF和瀑布法(Waterfall)
- 对这3种项目管理方法学进行比较

本章涉及推出软件产品的高级别步骤。我们将首先从如下两个方面来讲解：构思吸引人的愿景和筹备满足构建产品所需的资源。从中还会学到相互冲突的限制条件之间的关系。接着，会谈到3种流行的软件项目管理技术：Scrum、Microsoft解决方案框架(Microsoft Solutions Framework，MSF)和瀑布法。本章最后会使用从MSF和瀑布法中学到的一些知识同Scrum进行比较，从而深入理解Scrum的精髓。

本书内容讲解的其实就是利用特别的工具Visual Studio Team Foundation Server (TFS)来在优秀软件产品推出的过程中使用Scrum敏捷方法。最根本的概念就是把客户价值放到所有事情的中心。当最大化客户价值的时候，就最大化了团队的生产力和每次发布的可预测性，也就创造了一个推出优秀软件的可持续团队环境。在本书中，你将学到如何使用TFS中的工具基于Scrum方法学来管理软件开发项目。

1.1 推出软件产品需要做什么

在推出优秀的软件产品之前，需要先开发它。在开发之前，需要构思它是什么样子的，能干什么。在构思之前，需要知道人们为什么要使用它，以及人们如何使用它。在这之前，需要深入分析要解决的问题。所以，推出优秀的软件产品理所当然应该从愿景开始并深入分析功能应该是什么样的，如何增强它的功能和体验，了解人们

关心的缘由。

本节将讨论推出软件产品所需的步骤：

- 构思愿景
- 深入认识
- 筹备资源
- 规划进度
- 实现特性

1.1.1 构思愿景

优秀的软件总是始于优秀的愿景。它最开始是一个简单的描述，解释了我们要开发什么、为谁开发以及为什么要开发。如果不能在开始项目之前定义这些，那么应该再仔细考虑一下。创建软件产品需要数量巨大的资源，为了获取和分配所需的资源也需要构思一个吸引人的愿景。

愿景就是对产品目标进行有意义的描述并展示一个振奋人心的结果。不需要一鸣惊人——如成为世界新的亿万富翁——但要能引起所有利益相关者的共鸣。愿景理应描述用户关心的需求和利益。它理应描述亟待解决的问题，以及解决问题后所带来的商机。愿景理应激起客户、发起方和项目团队的兴趣。

产品愿景应该保持相对简短，几段话通常就足够了。它应该包括问题和商机的描述，以及解决方案和获得利益的描述。项目团队中的每个人都需要阅读它，他们需要同同事、朋友和家人谈论它。产品愿景还应该足够清晰易懂，让这些听众都能产生共鸣。

产品愿景通常伴随着范围定义。范围清晰地设定了愿景的工作路径：从问题的提出到实现解决方案。它应该包括一些高级别的特性、时间进度和限制条件。可以定义好原型、测试版和前几个发布版本的范围。早期的里程碑粒度较大；后期的里程碑可能较为模糊。范围应该明确定义计划实现什么和不实现什么。

通常，愿景和范围会合并到一个文档中，统称为愿景/范围。基于这个文档可以规划最初的一系列发布。按照持续发布的角度来思考和进行沟通在Scrum中非常关键。可以定义要在每次发布中包含的粗略特性，以便在团队迭代地向解决方案前进的过程中，每个人都能学习到这种增量的开发过程。

例如，假设你喜欢玩Fantasy Football这个游戏，可以在其中创建虚拟橄榄球队，并基于每个球员的统计数据进行比赛。Fantasy Football网站为创建球队和联盟提供了很棒的特性，不过没有包含很好的搜索功能。使用Google或Bing进行搜索并不能获得所想要的数据。突然灵感一现，你决定创建一个这样的搜索引擎来解决这个问题。那么，可以为这个产品创建如下的愿景：

为Fantasy Sports的爱好者创建一个搜索引擎。

基于这个愿景，读者就能在脑海中呈现出这个产品的用途和功能。这是一个好的开始，不过正如愿景这个词的涵义那样，它比所要实现的功能要宽泛。因为想法围绕Fantasy Football，所以你创建出来的愿景可以被演绎得更明朗一些。因此，可以把愿景改写为：

> 很多网站都可以管理Fantasy Football联盟。它们有很棒的特性，不过缺乏强大的搜索功能。我们将创建一款Fantasy Football搜索引擎，以便爱好者们可以访问更多的数据，创建更好的团队。

这样就好一些了，因为它将问题限制为最初心中想要解决的对象——Fantasy Football。它也添加了目标：创建更好的团队。现在，需要为这个问题划定范围。你打算解决Fantasy Football的问题，不过也意识到解决方案或许对于其他运动项目也能使用。因此，可以扩宽愿景，并以如下的范围来限制它：

> 很多网站都可以管理Fantasy Sports联盟。它们有很棒的特性，不过缺乏强大的搜索功能。我们将创建一款Fantasy Sports搜索引擎，以便爱好者们可以访问更多的数据，创建更好的团队。
>
> Fantasy Sports的直接市场有50个专门的网站，涉及300万以上用户。间接市场(包括新闻和体育站点)大约有500个站点，涉及2 000万人以上。
>
> 第一次发布的版本将针对美式橄榄球，并为现有的网站提供接口。它包括所有NFL团队的数据，并基于球员的统计数据提供参数化搜索。后续版本将添加更多运动联盟；将支持管理团队、球员、名次和交易的特性；也将为移动应用提供访问界面。

从这个愿景/范围中，读者将对要解决的问题、解决方案带来的好处，以及随着时间的推移扩展产品的迭代路径有清晰的了解。之后，可以定义业务案例、技术解决方案概念和重要假设。

在网络上有很多愿景/范围文档的例子。它们通常分成几节：愿景、需求、解决方案、假设、限制和风险。虽然Scrum对于项目启动没有规定要有愿景/范围文档，不过在软件项目管理过程中创建这样的文档是一个很好的实践，因为可以强制你清楚地表达项目的宏伟图景。

1.1.2　深入认识

拥有一个优秀的愿景是一回事；把愿景变为优秀的产品又是另外一回事。一般意义上的工程和具体意义上的软件工程也是如此。产品管理在工程和设计中是一个职责广泛的角色。它涉及把用户或市场的需求和期望转换为工程指示(instruction)。在这里

作者使用指示这个术语来指代很宽泛的东西，因为它可能表现为需求、故事板、用户故事、试验模型、可视化的演示或其他设计工件(artifact)。

正如将在第2章中读到的，产品管理是Scrum的一个基本元素。现在，假设你正和一个技能娴熟、愿意通力合作、经验丰富的团队一起工作，并且假设工作环境让团队激情昂扬。这已经有了一个很好的开端，不过还不足以构建一个成功的产品。在这种情况下，产品的成功依赖于产品管理。特别地，其依赖于产品管理来决定谁需要什么、为什么以及成本是多少。

产品所有者是Scrum中进行产品管理的角色，其属于Scrum团队的一员，紧密地和用户保持着联系。这个人必须知道用户的喜好厌恶、耐性和期望。产品所有者必须知道用户喜欢什么、不喜欢什么以及他们不关心什么。基本上，产品所有者必须对用户期望从这个产品上获得的价值有深刻的理解。

本章后面部分以及第2章和第3章中将会讲到，Scrum是一种开发软件的敏捷过程。它规定要预先进行规划和设计，并在迭代地构建解决方案的时候进行更多规划和设计。它一点也不随意和混乱，不过需要快速地前进，并允许出现意外的结果。鉴于此，产品所有者对用户需求的深入认识就显得尤为重要。

在Scrum中，如下几个因素影响着产品所有者角色对产品需求的认识：

- 做出决策——一个人而非一个组对产品管理的事宜做出决策。决策比传统软件开发中的模式要快，不过如果产品所有者缺乏对问题的足够了解，他或她将做出错误的决策，或缺乏信心来做出任何决策。
- 最简原型化——在项目开始之前，会做一个最简的原型，因为它是在项目早期的几个冲刺(sprint)中完成的。这意味着产品所有者的需求理解将可以很快地反映到产品上。
- 迭代即本性——产品特性被迭代式地和频繁地定义。基于产品所有者的决策，它们被划定到产品范围内或从产品范围中剔除，这直接影响着产品的价值。
- 客户反馈——我们会较早地和经常性地收到客户反馈。好的需求认识可从外来数据中甄别出有价值的那部分。

1.1.3 筹备资源

软件产品开发是一项资源非常密集的活动。它需要来自多种不同学科的人们协同工作，向一致的目标前进。这些人包括可视化效果设计师、领域专家、软件开发人员和产品经理。根据不同的领域，开发软件可能还需要业务分析师、安全专家、信息架构师和营销专家。

对于构建符合生产环境质量要求的软件来说，很容易对真实的成本估计不足。在你编写第一个计算机程序的时候是非常容易的：你是设计师、开发人员、测试员，也可能是客户。实际上，它还是充满乐趣的，我们中的很多人都对创造软件乐此不疲。

构建商业化的产品比创建自用的软件要困难得多。在开始构建之前，需要清楚地理解客户想要的东西。在开始购买相关技术或雇佣完成此工作的团队之前，需要明白客户愿意为此付出的代价。不管是只为一个客户还是成千上万的客户构建，开发所需的资源都远远多于为自己使用而构建所需的数量。

正如本章后面将看到的，在我们比较不同的软件项目管理策略的时候，Scrum通过迭代地往最终解决方案前进的方式优化了资源的利用。这使得我们可以基于面对的技术、团队和客户的现实情况来调整计划。

1．时间和金钱

我们需要大量的资源来构建和推出优秀的软件产品。从最高级别来说，就是需要时间和金钱。当然，两者间不存在互换关系，哪一样更多些都没有意义。

所谓“时机等于一切”正是软件开发的真理。这就是说，在错误的时间发布完美的软件通常毫无用处。这样的说法很有意思，令人深省，甚至还有点搞笑；不过如果时机错误，产品也就没有用处。类似地，在正确的时间发布错误的产品也同样毫无用处。它可能会获得大量的关注，因为这种无用的结果是潜在的，但假如产品缺乏某些重要的元素，也不太会成功。另外一方面，在正确的时间发布正确的产品就很有价值：就算它有一些瑕疵，只要时机正确并且产品也正确，也会获得成功。

一个具备完美时机的优秀产品例子

在正确的时间发布正确的产品的一个例子是Microsoft Windows 95。它是一个具备完美时机的优秀产品。那是在90年代初期，Microsoft正在开发Windows 95作为下一代操作系统，用于个人计算、游戏和企业生产力。不过一系列的事件集中发生在开发周期内，促使Microsoft重新考虑开发计划。Mosaic浏览器被开发出来并以Netscape Navigator的名义发布；Al Gore参议员倡议立法增加对Internet的投资；诸如Amazon、eBay和Yahoo！这样的消费者网站开始创造出真正的价值。Microsoft面临一个进退两难的境地：延迟Windows 95以等着将开发好的Web浏览器内置其中或者先发布一个没有浏览器的Windows 95并同时继续开发集成的浏览器。

Microsoft知道人们会升级他们的PC来访问Internet，而公司也不想错过向人们兜售操作系统的机会。于是，Microsoft选择了后者，按计划发布了没有浏览器的Windows 95，不过还是及时地跟上了Internet的大潮。虽然Windows 95离完美还有一段距离，不过它的时机却是很理想的。Windows 95被安装到了数亿计算机上，这巩固了Microsoft在下一个十年中桌面系统的地位。

2．人员和技术

与时间不同，金钱很容易就可转化为需要的资源。软件开发所需的最明显的两个

资源是人员和设备。

对于人员，你可能想要一个混合了普通人员和专家的团队。需要产品经理来定义特性集以确切满足客户的需求。需要软件架构师从始至终呆在项目中来定义产品的技术架构和设计模式。可能需要用户界面专家、数据库专家、安全专家，以及性能优化工程师。另外还有大量的技术工作将要由普通人员来完成：训练有素的编码和测试工程师。

如果你的项目需要的专家的比例高于普通人员，那么就要提高时间和成本的预算。找到并替换专家比换掉普通人员花费的时间和金钱要更多。

可以通过雇佣团队把金钱转化为人员。某些团队成员是内部雇员——那些在项目期间和其他时候都需要保留的人。其他一些可以是合同工——那些具备专才，在项目特定阶段才需要的人。

还需要把金钱转化为技术。可以购买、租用或借用技术来实现远程访问。对于硬件和软件来说都是如此。可以在问题出现的时候为其分配技术资源，并在环境改变的时候又变为其他资源(如金钱)。例如，在项目的性能测试阶段可能需要大量的硬件能力。在这种情况下，可以临时性地获得这样的技术资源，并在完成工作后转作他用。

1.1.4 规划进度

规划进度是推出优秀软件的一项基本组成部分。规划进度就是在项目开始之前很好地安排时间，并在项目进行过程中安排更多时间。不管是使用Scrum还是其他项目管理方法，这都是必不可少的。依赖于所使用的方法学，特定的规划活动会有很大不同。

例如，使用传统的软件开发方法来规划项目进度涉及给需求定义、设计、开发、测试、用户验收、发布管理和支持分配时间。软件开发只发生在开发阶段。使用Scrum规划项目涉及给在固定期限的“冲刺”中要完成的特性分配时间。在每个冲刺中，团队还是遵循瀑布法中的同样活动，不过只是完成很小的一些特性。图1-1显示了瀑布法和Scrum规划过程的高级视图。

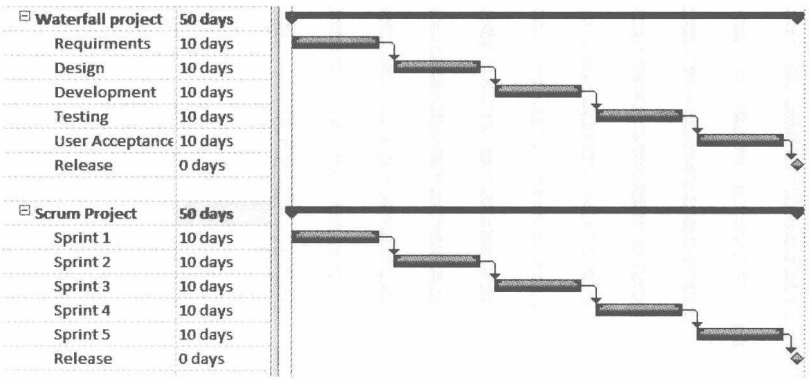


图1-1 瀑布法和Scrum的规划过程

瀑布软件管理方法是过去常用的一种管理软件项目的方法学。在使用瀑布法的时候，顺序地安排任务，在开始下一阶段之前需要完成本阶段的活动。在本章后面的1.2节中，我们将会更详细地研究瀑布法并和Scrum进行比较。

Scrum与诸如瀑布法这样的传统软件项目管理方法的差异之一就是可预见性。瀑布法假设能预测任务将花费的时间。可以根据这样的预测把人员和时间分配给这些任务，并相应地安排进度日程。Scrum却做了相反的假设。其认为无法准确地预测某些工作要花费多长时间，除非之前使用了同样的资源(技术和人员)完成过类似的工作。

如果不能预测一个任务要花费多长时间，那么又如何预测一系列任务要花费多长时间呢？如果使用Scrum，那么就要接受无法预测的事实。不是预测产品进度日程，而是预测在一个冲刺中能完成的较小工作单元。在每个冲刺结束后，相应的特性开发完成并包含到一个潜在可交付的产品中。通过这种方式，可以基于时间来预测特性而不是基于特性来预测时间，这本质上是一种“时间盒(time-box)”方式。

冲刺是Scrum中最小的周期。第9章会重点讲解冲刺的执行。冲刺可以是一两个月那么长，也可以是一两天那么短。在冲刺开始之前，ScrumMaster决定时间长短。每个冲刺都以冲刺计划会议开始，在会议期间团队浏览产品积压工作，决定哪些特性要在这个冲刺中开发。团队成员利用之前冲刺的经验来预测他们能在接下来的冲刺中完成多少东西。这和瀑布法有着显著的区别，后者预先估计发布和特性的时间，并把时间和人员分配给任务。

图1-2显示了使用瀑布法的项目计划中的一个甘特图。注意，每个任务都以确定的工期来安排日程。如果任务提早或推迟完成，都会影响到这次发布中的其他所有任务。如果对任务估算具有很高的信心，那么这种方式倒没什么问题；不过如果估计不正确，那么整个计划都会很快土崩瓦解。

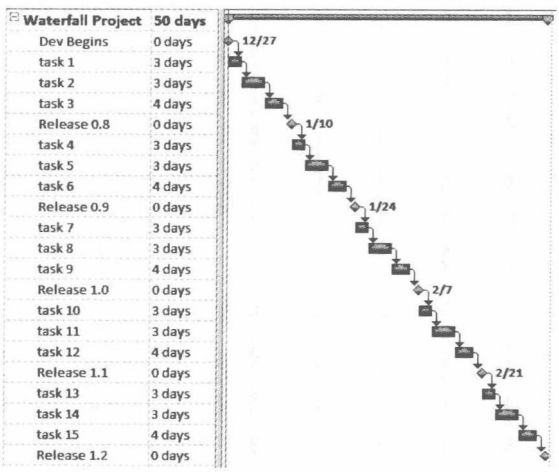
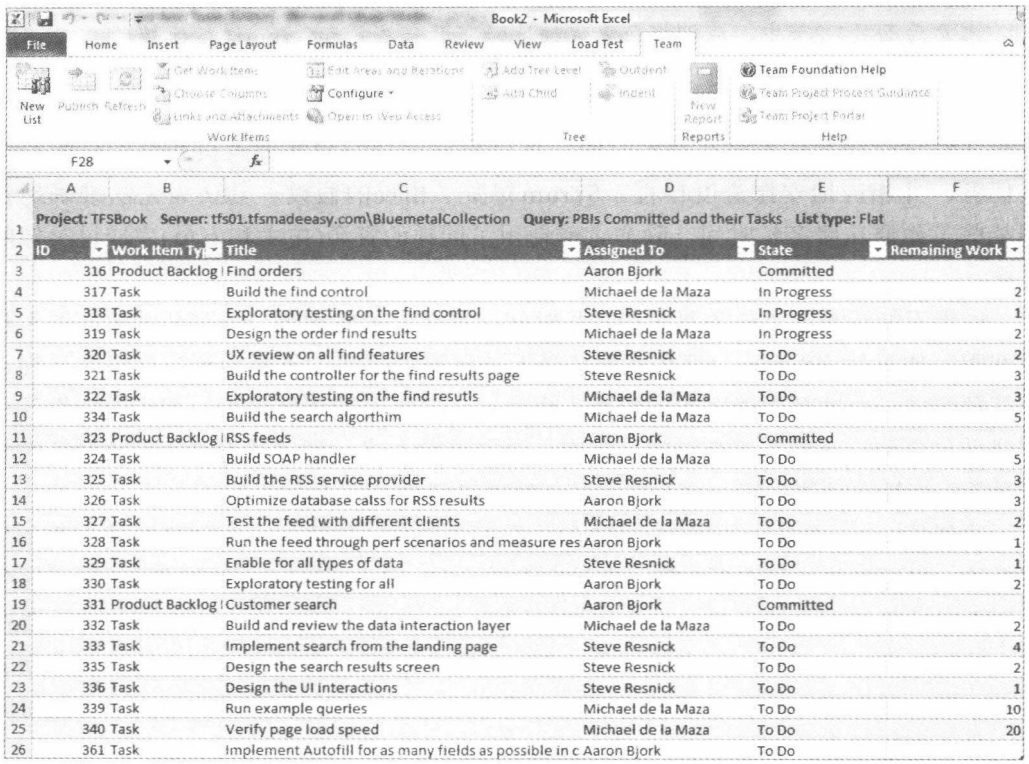


图1-2　瀑布法所用的甘特图

在Scrum项目中，必须规划任务。软件项目管理需要大量的规划，Scrum并没有改变这种需要。不过，它没有基于任务的前后依赖关系来规划，而是基于特性的发布情况来规划。团队更加专注于产品的开发，而非仅仅为了保持进度。Scrum中的日程也是很简单的，就是冲刺的周期。在一个冲刺中进行规划可以更专注于产品而不是进度，因为

进度本身是简单的。图1-3显示了用于冲刺内部规划的项目工件。这个报表以及类似于这个的其他报表将会在第6章中讲解。



Project: TFSBook Server: tfs01.tfsmadeeasy.com\BluemetalCollection Query: PBIs Committed and their Tasks List type: Flat

ID	Work Item Type	Title	Assigned To	State	Remaining Work
316	Product Backlog	Find orders	Aaron Bjork	Committed	
317	Task	Build the find control	Michael de la Maza	In Progress	2
318	Task	Exploratory testing on the find control	Steve Resnick	In Progress	1
319	Task	Design the order find results	Michael de la Maza	In Progress	2
320	Task	UX review on all find features	Steve Resnick	To Do	2
321	Task	Build the controller for the find results page	Steve Resnick	To Do	3
322	Task	Exploratory testing on the find resutls	Michael de la Maza	To Do	3
334	Task	Build the search algorthim	Michael de la Maza	To Do	5
323	Product Backlog	RSS feeds	Aaron Bjork	Committed	
324	Task	Build SOAP handler	Michael de la Maza	To Do	5
325	Task	Build the RSS service provider	Steve Resnick	To Do	3
326	Task	Optimize database calss for RSS results	Aaron Bjork	To Do	3
327	Task	Test the feed with different clients	Michael de la Maza	To Do	2
328	Task	Run the feed through perf scenarios and measure res	Aaron Bjork	To Do	1
329	Task	Enable for all types of data	Steve Resnick	To Do	1
330	Task	Exploratory testing for all	Aaron Bjork	To Do	2
331	Product Backlog	Customer search	Aaron Bjork	Committed	
332	Task	Build and review the data interaction layer	Michael de la Maza	To Do	2
333	Task	Implement search from the landing page	Steve Resnick	To Do	4
335	Task	Design the search results screen	Steve Resnick	To Do	2
336	Task	Design the UI interactions	Steve Resnick	To Do	1
339	Task	Run example queries	Michael de la Maza	To Do	10
340	Task	Verify page load speed	Michael de la Maza	To Do	20
361	Task	Implement Autofill for as many fields as possible in c	Aaron Bjork	To Do	

图1-3　Scrum项目工件

所谓工件是对项目有用的东西，但不是产品本身的一部分。工件的常见例子包括任务列表、进度表和测试用例。图1-3显示了一个把积压工作条目和任务混合在一起的列表，这样可以更方便地组织和使用它们。

1. 预算规划

在开始之前，要准确地预测软件开发的成本是非常困难的。我们总希望能准确预测，不过事与愿违。有很多未知的因素会影响到成本，如需求、技术选型和团队组成。例如，你拥有一个概要性的或者是详细的需求列表，而在准备进行估计的时候，需求可能会产生变动。又例如，你假设某个用于之前项目的技术对于当前项目也适用，那么在为这个新项目开发某个特定特性的时候，可能会进入不熟悉的领域。而且，就人的情况而言，也不能假设某个特定的人员将加入项目团队。更不要说，在估算项目的时候，还要考虑诸如资深架构师或数据库开发人员这样的角色。

图1-4中的项目管理三角揭示了定义项目的3个限制条件：成本、时间和特性。如果你对特性完完全全了解，那么就能提供一个成本和时间的精确估计。对于汽车修理和

家居装修来说是这样，那么为什么对于软件开发就不可能呢？答案是我们要面对很多未知的因素；在软件开发的世界中，在开始之前无法知晓所有一切。而且，你所不知道的东西会成为影响这个三角的因素，导致成本、时间和特性的变化。

有些人和文字在提及项目管理三角的时候，会用“铁三角”这个词来强调不能弯曲它来适应自己需要的事实。

由于这些未知因素和三角中固定的限制，因此无法简单地为固定特性的产品提供固定成本的承诺。所以，有两个选择：改变成本或改变特性。如果必须满足固定的预算，那么答案很简单：改变特性。

Scrum提供这种方式的改变。Scrum团队的成本可以是固定的。如果有两名产品所有者、6名团队成员和一名ScrumMaster，那么就可以把他们的周薪加在一起来决定每周成本。如果冲刺时间固定为4周，那么就知道一个冲刺的成本将是团队成本的4倍。现在，就可以精确地预测出冲刺的成本和周期。不过，客户不会为冲刺付费，而是为实实在在的产品付费。需要考虑的就是三角剩下的第三个元素：特性。

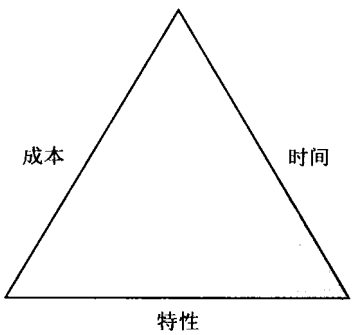


图1-4　项目管理三角

需要精心列出产品积压工作。积压工作条目必须基于提交给客户的价值排序成一个列表。通过优先级，可以找到首先需要实现的一组积压工作条目，甚至还可以识别出第二组或第三组接着要实现的积压工作条目。也就是说，通过列出积压工作，就能对于在一个固定期限内能够完成多少事情做出合理的估计。估计是针对固定时间期限和固定团队的，不过交付的特性集是可变的。因此，Scrum方法能满足预算控制的目标，也能解决软件开发中需要应对不确定因素的现实问题。

2．人员规划

Scrum项目天生就是团队驱动的活动，所有团队成员都要参与到项目的所有方面。团队的构成是很简单的，只用定义3个角色。团队倾向小规模的——最多10个人。由于只有如此之少的角色和很少的人员，团队成员会高度相互依赖。团队中的个体将会一同成功或失败。

Scrum需要紧密团结的团队，成员在项目早期就要进入团队，并在随后的冲刺中都始终固定。当然，无论在生活当中还是工作当中，总有人聚人散，项目不一定符合我们的预期，但是在Scrum项目中团队的稳固相较于其他项目管理方法学而言更为重要。

在后面的章节中，我们会描述冲刺规划如何以速度的度量为起点，速度指的是团队实现产品积压工作条目的速度。对于每次冲刺，团队会估计完成一组积压工作条目

所需的工作量。团队在承诺完成积压工作条目后，就实际开始工作了。团队估算的准确度会随着每次冲刺的进行而提高，只要没有东西出现巨大的改变。速度的度量会随着团队不断执行冲刺而变得更为准确。

在冲刺之间对团队进行调整对冲刺的预测性是具有负面影响的。例如，如果Manny、Moe和Jack被Tom、Dick和Harry代替，那么可以直接假设他们能完成原始团队同样的工作量吗？在不知道新团队成员能做多少工作的情况下，团队在完成积压工作条目的时候就很难估算时间。

虽然冲刺之间改变团队成员会对生产力和可预见性造成混乱、产生负面影响，但是在一个冲刺期间调整团队成员还是可行的。如果有可能还是应该避免这种情况。调整人员不仅对团队内部会造成波动，对客户也会有潜在的影响。例如，如果产品所有者告诉客户某个特性将在一个4周的冲刺之后就可以看到，但是由于团队在冲刺期间进行了调整导致这个特性无法按期完成，客户肯定会不满意的。

和任何其他过程一样，重要的一点是参与者要知道谁应该做什么和何时能做。在人们知道对他的期望后，通常能够做得更好。相反，不确定性会导致压力。因此，应该在项目开始之前对团队进行Scrum方面的培训。关于Scrum方法学有很多优秀的书籍和资源，可以用来辅助培训团队，让其知道所要承担的责任。附录B提供了一些学习的建议。

1.1.5 实现特性

对于产品的构建来说，时间和金钱——后者可以转化为人员和设备——都是有限的资源。这些资源和最终定义产品的特性集直接相关。假设产品特性是变化无常的，那么你的工作就是优化时间和资源的使用，从而构建出一系列可以让产品价值最大化的特性。

在某些业务讨论中，人员会被称为资源。这会冒犯某些人并误导其他人。这里，我们所说的资源正如其听起来那样只是指资源的供给。资源可以指劳动力、资金或设备的供给。资源可以直接地(例如，支付工资把资金转换为劳动力)或者间接地(例如，支付费用建造设备把资金转换为劳动力再转换为设备)交换。

质量是你能控制的一个特性。创建的产品是好是坏，取决于如何支配时间和金钱这两种资源。在一个项目中，如果在完成足够的测试之前已经用完了时间，那么可能只好降低质量。只要质量是一个需要资源来保证的特性，项目团队就可以完全地控制其好坏。

发布是你能控制的另外一个特性。你可以创建产品，但不一定发布它。在这种情况下，你可能不会获得太多的收入，但是认识到它是一种特性是很重要的。与产品的质量或其他特性一样，发布也要花费时间和金钱，所以分配这两者是很重要的。例