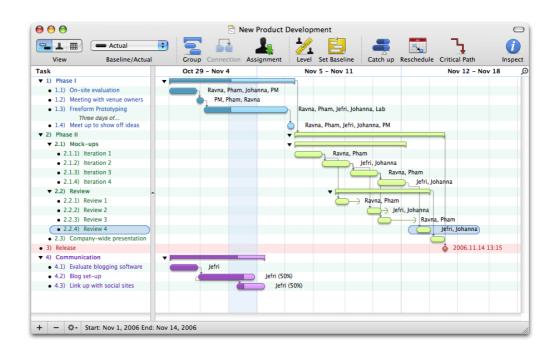
项目管理修炼之道

精选版



Johanna Rothman 著 郑柯 译

InfoQ企业软件开发丛书

免费在线版本

(非印刷免费在线版)

登录China-Pub网站购买此书完整版



了解本书更多信息请登录本书的官方网站

InfoQ 中文站出品



本书由 InfoQ 中文站免费发放,如果您从其他渠道获取本书,请注册 InfoQ 中文站以支持作者和出版商,并免费下载更多 InfoQ 企业软件开发系列图书。

本迷你书主页为

http://infoq.com/cn/minibooks/manage-it

序

大家好,欢迎阅读约翰娜的新书。我在软件行业已经有数十年经验了,目前是位于伯克利的Yahoo!的一名总监。不过,也许你听过数字设备公司(DEC,为互联网早期发展奠定了基石)和它的Alpha系统。那是我曾参与的一个意义重大的项目。

在Alpha系统的交付过程中,我扮演了非常重要的角色。那是一个名垂青史的项目: 2000多名工程师遍布世界各地,携手开发同一个系统的不同部分。这需要严谨的规划和项目管理才能成功。我们按照为期四年的时间表,在距离目标日期不到一个月的时间内交付了项目。所以,你大概也能想象得到,我觉得自己是个相当不错的项目经理! 不过,我后来才知道什么是真正杰出的项目经理。

1996年5月,我决定离开DEC。听说波士顿地区一家大型软件公司正在招产品团队总监,这正是我所期许的挑战,去领导一个陷入混乱的团队。我当时这么想:太棒了!这才是我想要的工作——循循诱导混乱的团队,帮助他们交付可以实际工作的产品——赶紧把我的大洋马牵过来!

我听说有个咨询顾问已经先期加入,试图根据团队产品beta版本的开发状况,分清轻重缓急,帮团队解决问题。这却更加让我坚信:不久之后,他们就会发现,我——才是他们一直在等待的大救星。

可是,我很快就感到了羞愧和震撼(而且感觉越来越强烈)。我知道咨询师们是干什么的,可是他们有谁能够通过实际行动、以实用的方式来厘清所面对的问题?可这位咨询师就做到了。 仅两、三个月的时间,她就让一切各就各位:项目章程、工程计划、项目计划、明确的角色和职责、明确的开发流程、恰到好处的度量标准、发布条件、参与beta测试的客户……所有这些成功项目的必备要素一应俱全。

要想把所有这些要素都安排妥当,怎么也得花上大半年时间,尤其是还面临启动资金不足的问题。可事实已经摆在那儿了!你可能已经猜到了,这位咨询师就是约翰娜·罗丝曼。(约翰娜在她的网站上放了一个案例研究,就是关于我们这次合作的;为了保护隐私,其中当事人的名字都使用了化名。)

认识约翰娜后的几年里,我先后在大大小小的几家公司里带过软件开发团队。很多时候,我都需要约翰娜的服务,帮助我的团队整体水平再上一个台阶。她的评估流程非常严谨,而且为有效的项目管理活动打下了坚实的基础。她主持的研讨会覆盖了很多话题——给我们讲过的有迭代产生项目需求、项目管理和QA。我曾聘请她出任临时的管理职位,让她使用自己丰富的技能完成一对一的培训指导。约翰娜拥有丰富的经验,曾在很多组织中处理过各种各样的复杂情况,她也总能拿出现实可行的方案,真正解决重大问题。

所以,本书可以说是约翰娜管理才华与丰富经验的结晶。

她把自己多年一线工作积累的经验,以系统严密的方式展现给读者。本书提供了可供你分析 所处的实际环境,构建项目管理框架和理性的执行计划,然后予以推进的各种工具。而大量提示 和实例,则告诉你哪些路可以走,哪些路行不通,更重要的是如何避开诡秘的陷阱。对我来说, 即使已经有了这么多年的项目和工程管理经验,我还是可以在书中发现新东西。当我处于陌生的 境况、面临全新挑战时,当我需要一个好参谋帮我应对难题时,约翰娜是我一定要找的人。

我和约翰娜合作的第一个项目最后怎样了?啊对了,我们将产品交付给了做beta测试的客户,而它也确实可以正常工作!

我坚信,约翰娜的这本书同样可以助你一臂之力。

艾伦·R.索尔兹伯里(伯克利Yahoo!研发中心总监)

2007年4月

前言

说到项目管理,你一定被不计其数的技巧、实践轰炸得头昏脑胀,时不时地还有各种相关建议迎面袭来。它们全都在说:"瞧我,我是最正确的。"

哎,它们很多都是正确的——在特定情况下。每个项目都是独一无二的,你必须要评估项目的情境(项目、团队、所在公司),然后再实事求是地作出判断,看看哪些可行,哪些不可行。

你的项目每天都在加快节奏,你的客户变得越来越不耐烦,大家越来越不能容忍无法正常工作的产品。也许你之前的做法还算不错,让你获得不少好评,可将来很可能不太奏效。你必须运用各种的方法和技巧来减少项目的风险,这其中就包括在每个项目中使用敏捷方法。

本书从风险角度出发帮助读者规划和指导项目。项目经理、团队成员、软件经理都能通过本书学习成功之道。即使你要构建有形的产品,比如一座房子、某种电路板,或是要管理服务类型的项目,本书中的很多内容仍然适用。

本书假设读者负责管理高科技项目,而且项目至少涉及一些软件开发。也许你像我一样,已 经拥有了一些项目管理经验:包括纯粹的软件项目以及软、硬件结合的项目。我也管理过一些服 务项目,比如规划和主办会议;参与过一些建筑项目(一套新房子、一次小规模的重新装修、一次大规模的重新装修)。可是我主要的项目经验都来自软件或是软、硬件结合的项目。

相对于交付实体产品的项目来说,软件项目要更加难以管理。软件很难把握,它没有形状,不需要原材料,也不是由物质构成的,所以看不见、摸不着,也没有办法直接测量。很难看到产品实实在在地在我们眼前发生演变,很难发现和预测风险,因此也就更难以应对风险。而开发软件产品的方式也并非总有助于我们了解项目进度或者把握其方向。

如果管理的是开发有形产品的项目,项目经理可以看到产品逐步成型。你可以见到房屋的框架、完工的墙体从框架到墙体,以及所有的建造过程。对于服务型的产品,比如像会议这种会产生具体结果的项目,你可以深入了解一些项目的临时交付物,比如会议报告草稿或是会议日程,等等。所以,在运作项目期间,你可以看到有形产品项目和一些服务项目的具体进度。

要是不能直接看到项目进度,那该怎么办呢?当你发现项目有点儿不对劲儿,而且可能濒临险境时,你该怎么办?如果此时有些项目干系人不支持你的决定,你又该怎么办?

本书可以让你深入了解你的软件项目,并让你成功管理项目的风险,无论这些风险是伴随着项目开始而存在、还是在项目进行到中间阶段时才出现。从章程制定到产品发布,每一章都讨论了一种能帮助你看清软件项目本质的方式,让你从各个方面度量它、感受它、品味它、体会它。

但在本书中,你找不到项目管理的绝对真理,因为没有在所有项目中都颠扑不破的绝对真理。 你也看不到普适的最佳实践,我提出的能帮你和你的团队达成目标的实践,都有其针对的特定生 命周期。

你在书中会发现有很多前后交叉引用的内容。这是因为项目是非线性系统。早先所做的决策 会影响到项目如何结束,甚至可能影响到如何启动下一个项目。你管理项目的方法,也会影响你 管理产品待办事项列表或项目组合的思路。

书中的所有文档模板可以在本书的主页上找到: http://pragmaticprogrammer.com/titles/jrpm。

我想感谢所有为我撰写和修改本书提供帮助的人: Tom Ayerst、Jim Bullock、Brian Burke、Piers Cawley、Shanti Chilukuri、Esther Derby、Michael F. Dwyer、Mark Druy、Jenn Greene、Payson Hall、Peter Harris、George Hawthorne、Ron Jeffries、Bil Kleb、Michael Lee、Hal Macomber、Rob McGurrin、Andrew McKinlay、Erik Petersen、Dwayne Phillips、Frederick Ros、Ellen Salisbury、George Stepanek、Andrew Wagner和Jim Ward。我的编辑Daniel Steinberg提供了非比寻常的有益反馈。Kim Wimpsett再次证明他是一个极其出色的文字编辑。我要感谢Steve Peter在排版上的神奇表现。Rotate Graphics的Mark Tatro绘制了日程游戏一章(第6章)中所有的卡通图画。与Andy Hunt和Dave Thomas的再次合作,同样让我深感荣幸。书中任何错误都由我来负责。

书中讲述的故事全部源于真人真事,但考虑到保护隐私,相关的人名、公司名和事件细节都已做过修改。

我们开始吧。

约翰娜·罗丝曼 2007年4月



每月8号出版













商务合作: sales@cn.infoq.com 读者反馈/投稿: editors@cn.infoq.com

目 录

序	I	2.3.7 人员配备	20
前言	III	2.3.8 建议日程	20
	章 启动项目1	2.3.9 制订项目风险列表	
弗 Ⅰ □		2.4 制订发布条件	21
1.1	定义项目和项目经理1	2.4.1 确定当前项目最重要的因素	22
1.2	管理项目的关键驱动因素、约束和	2.4.2 草拟发布条件	23
	浮动因素2	2.4.3 让发布条件符合SMART原则	24
1.3	与客户或出资人讨论项目约束5	2.4.4 在发布条件上达成多方共识	25
1.4	决定项目的关键驱动因素6	2.5 使用发布条件	25
1.5	应对喜欢过多干预项目的出资人7	第3章 识别和避免日程安排游戏	27
	1.5.1 预测未来8	and the second s	
	1.5.2 使用与上下文无关的问题	3.1 给我一块石头	29
	识别项目真正的驱动因素8	3.2 "希望"是我们最重要的策略····································	31
1.6	编写项目章程,共享现有决策9		
	1.6.1 远景10	3.4 把灰扫到地毯下面	33
	1.6.2 需求10	3.5 幸福日期	
	1.6.3 目标10	3.6 屁股着火	
	1.6.4 成功标准11	3.7 分散注意力	
	1.6.5 ROI11	3.8 日程等于承诺	
1.7	理解质量对于项目的意义12	3.9 到了之后,我们会知道身处何方	40
第2章	章 规划项目14	3.10 日程安排工具总是对的,又称为梦幻	
2.1	踏上征程14	时间日程	42
2.1	噴工证程·······14 使项目足以启动的规划······14	3.11 我们必须拥有这个功能,否则就	
	开发项目规划模板16	完蛋了	
2.3	77 及项目规划模似····································	3.12 我们不能说"不"	
		3.13 日程小鸡	
		3.14 90%完成状态	48
		3.15 我们马上会变得更快	
		3.16 令人恍惚的日程	50
	2.3.5 项目组织	第 4 章 掌控项目	52
	2.3.6 日程总览19	カ T キ	

	25 12 - T D 44 - 15 - T D 44 - 15 - 15 - 15 - 15 - 15 - 15 - 15 -	FOR ESTERNIZED TO A TO 14 TO
4.1	掌控项目的节奏52	5.2.7 与项目团队人员分配情况
4.2	举行中途回顾53	相关的图表80
4.3	为需求排序56	5.2.8 判断项目的变化率82
4.4	用时间盒限定需求相关的工作57	5.2.9 查看开发人员是在取得进展
4.5	将迭代限制在4周或是更少的时间内58	还是在白费时间83
4.6	使用波浪式的规划和日程安排59	5.2.10 测量查找和修复问题的
4.7	创建跨职能团队62	成本84
4.8	根据项目的风险选择生命周期模型62	5.2.11 了解开发人员和测试人员是否
4.9	保持合理的工作时间63	在解决缺陷方面取得进展85
4.10	C 4	5.2.12 展示测试过程86
4.11		5.2.13 展示定性数据87
	4.11.1 应对其他项目干扰	5.2.14 用图表展示团队同意采用的
	4.11.2 应对其他人的干扰	实践88
4.12		5.2.15 度量敏捷项目90
		5.3 为出资人创建项目仪表板90
第5章		5.3.1 展示风险列表90
5.1	测量有风险70	5.3.2 展示在满足发布条件方面取
5.2	根据项目完成度来衡量进度72	得的进展90
	5.2.1 使用速度图表跟踪日程安排	5.4 使用项目气象报告92
	进度72	5.4.1 要小心定义气象报告的图示93
	5.2.2 使用迭代内容图跟踪总体	5.4.2 建立可信的气象报告95
	进度74	5.4.3 每周发布气象报告95
	5.2.3 计算软件项目的挣值并无	3.4.3 4月及州 (水水区
	实际意义74	
	5.2.4 用EQF跟踪最初的估算	
	5.2.5 更多的度量能提供更多信息78	
	5.2.6 如果只能度量项目日程,	
	那就这么做79	
	1 00 0 1111	

启动项目

1.1 定义项目和项目经理

项目到底是什么?我们首先要有一个一致的定义。

项目:一个独特的任务或是系统化的流程,其目的是创建新的产品或服务,产品和服务交付完成标志着项目的结束。项目都有风险,并且受制于有限的资源。^①

项目经理负责管理风险和资源。交付日期迫在眉睫,技术目标难以达成,产品质量漏洞百出,项目资金即将告罄,人手面临匮乏闲境——没有项目管理,贯穿于项目中的这些风险如何应对?

因为各自的关注点不同,每个项目都是独一无二的。项目经理要根据每个项目不同的实际情况,进行管理和规划。在启动之前,着手收集信息,了解项目要实现哪些目标。

我们的目标是什么?

——克里斯,项目经理

我是一个大公司的项目经理,主要负责软件和硬件的集成项目。我的同事妮姬负责主办各种活动。我的团队搭建电脑系统,而妮姬的团队负责活动组织——虽然彼此的产出根本不沾边儿,可我们都是项目经理。

我们面对的风险完全不同,对外交付的东西也根本不一样。我要的是软件、硬件以及一些 文档,而妮姬需要展览摊位、食品、以及其他有助于活动成功的必备之物。

① © 2007 R. 麦克斯·怀德曼, http://www.maxwideman.com。经许可后转载。

我们的工作有一个共同之处: 开始时要知道项目的目标,这样就可以马上知道我们接下来要做什么了。知道要做什么、"完成"意味着什么,这对我和妮姬的团队很有帮助。

每个项目都是独一无二的。项目团队规模不同,各自的能力也有所区别。有些项目的客户是 内部的,有些则是外部的。有些项目面临很紧迫的时间压力,有些项目则可能要面对其他威胁和 风险。产品是每个项目的核心,让我们再就产品的定义达成共识。

产品:项目产生的一系列可交付物。

作为一名成功的项目经理,或者想成为成功项目经理的你,应该花些时间来发现当前项目的独特之处。这样,你就有可能成功地启动、管理和结束这个项目。

我们现在已经有了项目和产品的定义,接下来该搞清楚项目经理的职责了。怀德曼说,一个项目经理"被赋予管理项目的权力和责任,并带领项目团队,通过项目管理来达成项目的目标"。[©]这个正式定义还挺不错的。不妨看看下面这几句大实话,比较而言,也许它给人的感觉更加模糊,但同时也更准确。

项目经理: 负责向团队清晰说明完成的含义,并带领团队完成项目的人。完成,是 指产品符合组织对这个产品的要求,也能满足客户使用这个产品的需要。

其中,完成的说法是暗含风险的。我可以确定任何项目的产品一定伴随着风险,我们会在后面讨论如何对这些风险进行识别和分类。在采取任何进一步行动之前,项目经理必须理解项目的关键驱动因素是什么。

无论规模大小——是项目就有风险

——埃里克,资深项目经理

我曾参与过延续数年、团队达数百人的大项目;也参与过一些小项目,我们的团队只有两三个人,客户也只有两到三个人配合,整个项目持续时间三个月。关于项目,我有一点非常确定:通往最终交付的道路上,总是有风险伴随在我们左右。

有时,风险因团队具体情况而不同。设想整理床铺这件简单的小事。对我来说,这只不过 是待办事项清单上的一个条目而已。可我的孩子们却觉得,铺床就像是一个满是风险的项目, 最主要的风险就是他们无法在睡觉之前把床铺好。

1.2 管理项目的关键驱动因素、约束和浮动因素

要完成一个项目,如果不能理解项目的背景,前进的路上必然充满艰险。

① © 2007 R. 麦克斯·怀德曼, http://www.maxwideman.com。(经许可后转载)。

项目的背景是由所在组织的关键因素决定的。项目的驱动因素是什么?是否存在约束?有没有可能在驱动因素和约束之间进行折中?项目经理能否为自己争取更多的自由运作空间?

当前项目的驱动因素与上一个不同

——斯图尔特,项目经理

我现在正在管理我个人的第二个项目。第一个项目的交付物是公司旗舰产品的一个补丁版本,因而很容易就能知道我们的工作重点:添加几个功能特性,并修复多个缺陷。

但是这第二个项目——哎,可完全不一样了。这个项目会被用作目标市场的探路石。我们需要很多新功能,这些功能还得没啥大问题。工作重点不是解决缺陷,因为交付日期马上就要到了,得赶紧让产品投入市场。老板跟我说可以多给我几个人,但是不会有外包公司参与,因为要控制成本。

能够识别出哪些事情在推动项目让我获益匪浅。我发现优先级最高的就是功能,其次是交付日期,接下来是人。只要能发布,公司对于缺陷和成本卡得就不是那么死了。

也许大家都听说过项目的"铁三角":成本和时间是这个三角形的两条边,第三条边一般是质量或范围(见图1-1)。其实铁三角过于简单了。斯图尔特的两个项目的驱动因素就存在很大差异。





图1-1 传统的铁三角

像斯图尔特这样成功的项目经理会权衡许多因素,而不仅限于铁三角。以为只要让客户从铁 三角中选择他最看重的两条边,然后就可以交付他想要的结果,这根本是痴人说梦。要是这么简 单的话,那任何人都能当项目经理了。

首先,要写下客户的期望——从客户的角度来看,项目的驱动因素是什么。这个列表中应该包括:客户想要什么(功能集合),他们期望何时收到交付物(发布时间),可交付物的质量如何(缺陷等级)。

接下来,写下项目的约束。环境是什么样子的?能否灵活安排团队的位置?必须遵守什么样的流程?手下的人有哪些?他们能做什么?预算有多少?这些约束是可以改变的(一般只有项目出问题的时候才能改变)。约束决定了项目的规模(还有持续时间和质量)。

对比客户的期望列表和项目的约束列表。你脑海里蹦出来的项目成功的必要因素有哪些?选 择其一,比如发布时间,这就是识别出来的项目的**关键驱动因素**。

列表上还剩下什么?应该还能看到功能集合、低缺陷率、发布成本等条目。在这些条目里,需要对哪些进行管理才能保证项目的成功?可以按重要性排列这些关注点。客户和出资人会在这些方面上对项目形成限制。从中选择两到三项,我们称之为项目的**约束**。

再看看剩下的条目。有些条目很重要,不过它们有很大的调整余地,我们称之为**浮动因素**。 一个项目至少要有三个浮动因素。

最后看看还没选择的条目,是不是还有哪个比已经选择的更重要呢?如果有,那就再调整一下;如果没有,这个项目的关键驱动因素、约束、浮动因素就都识别出来了。

我曾经见过有三个约束的项目,团队必须付出超乎寻常的努力,才能保证项目成功。以我的 经验,如果项目有一个关键驱动因素、两个约束、三个浮动因素,那它的成功几率还是不小的。 浮动因素越多,项目就越容易管理。

理想状况下,关键驱动因素应该只有一个,约束应该只有一个,而浮动因素可以有四个。大部分情况下,我们管理的都是不那么理想化的项目,所以如果项目有一个关键驱动因素、两个约束、三个浮动因素,这也是可以成功的。过多的关键驱动因素或约束,会让项目过于受限。这种情况下也许能够成功,但是项目经理和团队就得选择促成项目成功的组织形式和实践——不过还是要有点心理准备,因为可能无法获得百分之百的成功。

如果存在过多的关键驱动因素和约束,而且项目经理除了启动项目之外别无选择,这时所能做的就是选定一个关键驱动因素,并尽可能频繁地向出资人提交项目的产出,帮助他们决定自己真正想要的东西。在启动项目的时候,为了得到项目的成功条件,可以问一些与上下文无关的问题。还要定义出发布条件,这样就知道到什么程度算做完了。

提示: 要是约束太多了, 你就自己作决定吧

受到过多限制的项目难以逃避失败的厄运。过多的关键驱动因素,意味着没有人知道成功的条件是什么。过多的约束,意味着组织中没人愿意去判断孰轻孰重。

有必要的话,让管理层决定项目的关键驱动因素、约束和浮动因素。如果这不可行,那项目经理就自己作决定吧;项目和组织会因此而受益。

想创建出不受约束限制的项目需求是不现实的。假设有一个可以承重5斤的书包,如果你硬 要往里面塞10斤的书,不外乎两种结果:要么书包带断掉,要么书包底破掉。管理项目也是如此, 要根据约束来管理需求。要想同时应付过多的需求,那不管用什么办法,人手、时间、预算还有 工具这些资源可能就是不够用,发布出的产品也很难具备高层想要的全部功能,同时可能有很多 缺陷。

1.3 与客户或出资人讨论项目约束

要主动理解出资人想要的东西。下面这段谈话就是我在最近的项目中与出资人的对话。

JR: 哎, 克莱德。咱们看看到底是什么在驱动这个项目吧。

克莱德:可别!别再整我了。上个项目的时候,你就让我做过一次了啊。

JR: 没错。还记得吗? 你当时想添加新功能。你希望在产品发布之前塞进去尽可能多的功 能,我可以加,因为当时组织项目的方式还允许。那确实挺不好弄的,不过还是有可能做到。

克莱德: 啊,对,我忘了。不过这个项目不一样啊。

JR: 哦? 跟我说说。

克莱德: 你瞧, 管人是你的事情, 项目的运作情况你也得管。不过你不需要资金, 我也不担 心成本问题, 因为唯一的成本就是工资。

JR: 还有软件呢。

克莱德: 你还真挑啊。好吧, 如果需要什么软件, 跟我说。不过老实讲, 基本上工资就是这 次的全部成本——我也不用管,我最关心的是项目要多久才能完成。

JR: 那需要哪些功能呢? 对它们的质量有什么要求? 这是给财务部门用的一个小程序, 你 也知道他们总是要求完美。要是我不能保证给他们的东西毫无问题,莱斯丽会跟上面吹风的。

克莱德:没错,不过我会帮你扛着的。我就是希望给他们够用的功能就行了,这样你跟你的 团队就能很快搞定, 比如在十周之内。

JR: 如果到了第八周, 我们还没有开发完所有的功能, 而且还有很多bug要改。你会怎么想?

克莱德: JR, 别啊。我需要全部的功能。你们开发团队曾经完成过类似的工作, 我相信这次 也一定行。

JR: 克莱德,如果我能理解你到底想要什么,你知道,我们开发团队会尽力而为的。

克莱德: 好吧。千万不能有不完整的功能。你尽管开始,把它做完,而且要让莱斯丽喜欢它。 否则,她非把我头砍下来不可。文斯已经启动了一个大型的项目计划,过段时间,我想让你们开 发团队加入到那边去。他说已经准备好在十个星期后接收更多的人了。这样,你也有足够的时间 来给莱斯丽一点儿用得上的东西。

1.4 决定项目的关键驱动因素

在与克莱德的对话中,我让他说明他最看重的是什么。克莱德说一个大型计划会在十个星期 之后启动,很明显,日期就是这个项目的关键驱动因素。

在项目早期,似乎一切皆有可能,特别是没有估算任何工作的时候。出资人可能会说:"我们想要这5个功能,在8月1日之前完成,还不能有严重的问题。我们还想让你把成本控制在一百万美元之下。给你这6个人。OK?"

可别轻易说OK。

估算了工作量之后,就能知道用这么多钱、这么长时间,这6个人能不能做完项目。要是可以,蛮好。不过很可能没有办法按照出资人对时间、金钱、人力的要求,在规定的工作环境中,提供符合他们质量要求的产品。此时,出资人需要做出艰难的决策,要考虑组织中的项目驱动因素,比如发布日期、功能集合、成本、人员分配及其分配时间、将要采用的技术和实践,等等。

要是出资人不愿意判断项目的驱动因素是什么,这个责任就落到项目经理肩上了。如果项目经理不做决定,团队自己会决定。但是他们会各拿各的主意,而且这些主意也不一定会符合出资人的想法。实际上,每个人——不管他或她在项目中的角色是什么——都会根据自己的想法进行判断。有的人会优先考虑发布日期的限制,从而把减少缺陷的想法抛在脑后。有的人会根据日程安排,仅仅实现一个功能——一个包含完整回归测试套件的功能,其余的功能却都没做完。有的人会优先考虑实现功能集合,开发出尽可能多的程序桩(stub),当测试人员发现问题时再去填充,直到时间用完为止。每个人都会做自己认为正确的事,而不是从出资人(或项目经理)的角度出发,因此做出的决策彼此不同。

通过制定优先级列表可以解决这个问题,其过程跟做数独游戏有些类似,即将所有可能的驱动因素列在左边,右边空出来等着填数字。

使用矩阵表明项目的优先级

下面是克莱德的排序矩阵。

项目驱动因素	排序
发布成本	5
发布日期	1
功能集合	2
减少缺陷	3
人员配备	4
工作环境	6

在这个项目中,发布日期是最主要的驱动因素。如果产品今年不能发布,这个项目就没有什么存在的意义了。但是完备的功能也很重要——功能不齐全,即使及时发布,整个项目也没有价值。而且,由于公司业务属于受政府管制的行业,产品的缺陷率必须很低。接下来是人员配备,因为只要这些人能在十个星期之后参与下个项目计划就可以了。项目的成本控制不太重要,因为项目的价值会很高。工作环境排在最后,为了保证及时交付,我可以灵活调整某些事情。

即使已经有了排好顺序的优先级,我们还是没有多少灵活性。不过了解了项目的关键驱动因素,我就可以定义出项目的成功条件,并选择适合项目的生命周期了。项目团队可以制定出发布条件,并根据驱动因素合理地安排各自的工作。嗯,最后,我们按照当初的要求,成功地交付了项目。

1.5 应对喜欢过多干预项目的出资人

在绝大多数情况下,关于成功标准的谈话不会进行得那么顺利。读者也可以看到,我必须促使克莱德做出选择。类似情况很常见。

有些项目对于功能集合、减少缺陷、时间安排这三者的要求都是最高的,而且优先级相同。 我敢说,读者一定以项目经理或团队成员的身份参与过类似项目。你不能再给团队添加更多人手 了,而且项目的预算不可改变。项目流程必须按照公司的强制要求执行,还得使用公司规定的办 公室和家具,其他一些工作环境问题也让项目难以推进。除非没有技术或时间上的风险,否则没 人能够使这样的项目取得成功。不过还是有一些方法,能够理清这些看来已经没有活动余地的约 束条件。

在1.4节中可以看到项目的驱动因素矩阵。如果出资人愿意排定优先级,那这种方式再好不过了。有些勉强的出资人可能因此而变得更加坚定。项目经理有时要先草拟一个优先级列表,拿给出资人看。比如,几个出资人对于优先级的排序可能无法达成一致意见。要是出资人不愿意拿主意,项目经理就只好自己做决定,然后再给出资人看。她可能会愿意去修正列表,或者直接签

名通过这个列表,自己就不再建新的了。

要想明确项目真正的驱动因素,有两种不同的途径可供选择:预测未来;使用与上下文无关的问题。

1.5.1 预测未来

让出资人设想这样的情况:现在离项目预定交付时间只剩三个星期,却还有功能尚未实现。 (可以着重讨论该出资人需要的一两个功能。)除了没做完的功能之外,还有很多严重的缺陷等着 修复。这样看来,要想在预定的发布日期之前开发完所有的功能并修复所有的缺陷,很明显是不 可能了。这个时候,出资人该怎么办?

如果出资人这么说:"把你的脑袋砍下来给我吧。"那就该考虑"三十六计走为上"了。请看7.7节。

不过出资人很可能会这样说:"把这个该死的项目做完吧。你还需要几个人?"接着,你可以问他:"是先做完所有的功能,还是先修复所有的缺陷?"他一般会说:"这两个功能和这些问题都得解决掉。"再问他:"以项目目前的优先级顺序,是吗?"项目经理要经常帮助出资人搞清楚:哪些约束是真正的约束,哪些是出资人自己一厢情愿的想法。

在对话中使用与上下文无关的问题,有助于识别出项目的驱动因素、约束以及活动余地。

1.5.2 使用与上下文无关的问题识别项目真正的驱动因素

明确了驱动因素之后,项目经理就可以发掘**项目**的需求了。与上下文无关的问题有助于抽取 出这些需求。通过这些比较抽象的问题,可以诱导其他人说出他们对于项目的假设。不妨从下面 这些问题开始。

- □ 项目要怎么样才算成功?
- □ 为什么想得到这样的结果?
- □ 这种解决方案对你来说价值何在?
- □ 这个系统要解决什么样的问题?
- □ 这个系统可能会造成什么样的问题?

要注意,少用"为什么"之类的问题。"为什么"问得越少,对于业务需要反而能了解得越多(而且问问题的人也不会像个令人厌烦的三岁小孩子。)同时,"为什么"这类问题很容易让对方产生戒心。也得注意避免"怎么做"之类的问题,出资人会觉得你在让他们设计系统。

在问问题时,要让人感觉到你真心希望了解这个项目,而不要让别人抱有戒心。这些问题可 以为项目经理和出资人将来的合作打下良好的基础,而不是形成龃龉的关系。

这些问题可以找出系统的价值所在。在向出资人提问时,要营造平等的气氛。在纸上做笔记, 而不是用电脑,这样你和出资人之间就不存在障碍了。将这些问题作为谈话的开端。在刚开始时, 从出资人处收集到的关于项目价值的信息越多,你就能更深入地了解如何设计这个项目。

项目要怎么样才算成功?

——贾斯汀,项目经理

几个月前,我开始管理这个会持续两三年的项目。这两天,老板跑过来跟我说他想加入一 个新的功能。一个我在说:"酷啊,这个新功能一定会很棒的!"而另一个我说:"噢不,我 们介入这个项目已经两个月了。要是我给老板留下可以随意加入新功能的印象,如果不调整我 们团队的工作方式,那我一定会有麻烦的。"于是,我问老板这个问题:"对你来说,项目要 怎么样才算成功?"

使我大吃一惊的是, 老板认为这个项目要能完全随需应变才算成功。他十分确定: 不到最 后一刻,需求不会停止变化——也许要到我们发布前一周才能最后明确。我以前可从来没有管 理过类似的项目! 不过既然现在知道了, 我就能着手筹划应对之策了。

1.6 编写项目章程,共享现有决策

项目章程会明确记录项目的需求和约束,还可以帮助项目经理思考如何进行项目规划。整个 团队和出资人都可以杳看项目章程,以此确保他们对项目有关的决策可以达成一致。

在启动项目时,编写项目章程可以让大家知道应该完成什么目标,以及项目干系人提出的约 束条件。即使不知道完成项目需要的细枝末节,编写章程也有助于发现潜在的问题。项目章程可 以帮助项目经理和团队理解风险、成功标准,而大家也可以藉此考虑组织和操控项目的方式。

如果你在管理一个敏捷项目(使用敏捷生命周期的项目,请查看附录A.4节),项目章程会很 简短,只要包括项目远景(参与项目的人可以做出正确的决策)和发布日期(这样就不会为了给 产品"镀金"而错过项目的结束时间)就够了。

下面是我的项目章程模板。

- □ 远景
- □ 需求
- □目标

- □ 成功标准
- □ ROI估算

项目章程是有意要设计成这么简短的,目的是帮助团队赶紧启动。它不会包含团队对于这个项目"完成"的定义,也不会介绍团队如何组织项目,但是已经足够让大家着手开展工作了。

1.6.1 远景

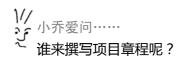
每个项目背后总有一个缘由(或者两个、三个)。发起这个项目的缘由何在?项目的价值何在?要用描述远景的句子来说明项目的价值。越早向大家阐明项目的价值,团队就越有可能告诉你接手这个项目是否明智。也许他们会告诉你,囿于目前的约束、资源和他们的能力,这个项目就不可能完成。要是不能明确阐明项目的远景何在,很可能这个项目就是"不可能完成的任务"了(因为没有远景的项目无法结束)。实用的项目远景对团队来说不可或缺。

1.6.2 需求

某些情况下,项目唯一的需求就是要在某个特定日期到达之时发布某些功能。更多时候,需求掺杂在下面这样的语句之中:"2月20日发布的主要版本中,我们需要这个又棒又炫的功能。"这些才是项目的驱动因素,产品功能列表不是。

1.6.3 目标

项目目标是希望通过项目要达成的目的,不过客户跟出资人可能并不赞同这些目标。(想了解更多关于目标的讨论,请查看2.3节)。



撰写章程,可以帮助团队早点形成凝聚力。要让尽量多的团队成员参与编写章程。 要是还没有为项目分配人员,那就先自己写章程吧。如果不是单枪匹马,就跟大家一起写。在项目启动会议中,把章程跟大家过一遍,确保每个人都知道其中的内容及其缘由。

目标与需求不同。项目并不一定必须交付它的目标。遵循传统的阶段-关卡(phase-gate)式 开发过程或瀑布式开发过程的项目,它的产品里面会有比较严重的技术债务(technical debt,请 查看原书附录B)。通过重新设计解决技术债务、添加更多的自动化测试、设计冒烟测试等等,这 些都是可能的项目目标。

客户有时会提出一些特定的要求:"如果目前的时间安排允许,并且不麻烦的话,我想要一个电子签名。"作为一个注重实效的项目经理,你可以说:"OK,我们会把它加到项目目标里面去。要是雪丽和简有时间的话,她们会去做的。"

1.6.4 成功标准

成功标准是围绕客户能基于完成的产品做什么给出的定义。成功的标准并不涉及缺陷,而只 关注能力。

下面是一些成功标准实例。

- □ 要包括功能1、2、3,这样我们的产品就可以打入目标市场了。
- □ 要提升产品性能,再测出相关数值,这样我们就可以将其与竞争对手的产品进行对比, 并编写新的市场营销材料了。
- □ 客户不需要访问我们的网站,就可以打开安装包、加载软件。
- □ 在第一季度发布。

在你意识到之前,项目就已经开始了

项目通常在正式开工之前就已经启动了。也许有人已经开始了原型化的工作;也许有人已经预想了一些功能;也许管理层已经跟首席架构师探讨了项目在技术上的可行性。这些都是项目的组成部分,即使你不这么认为。

由于项目在你思考之前就已经开始了,所以大可不必因项目章程、项目规划和项目日程的 反复修改而烦恼。作为一个注重实效的项目经理,在项目前期,虽然有些工作已经开始了,你 还是应该张开双臂拥抱项目,接受眼前的一切。在项目中期,你应该不断评估项目进程和风险,从而掌控整个项目。项目收尾阶段,如果你是注重实效的,那就没什么好担心的了。

团队要控制成功标准。项目经理要确保成功标准中不会包含非项目人员才能完成的任务(比如"卖出50000套软件")。项目经理和团队不能控制别人做什么,只能控制自己和团队的行动。要确保成功标准在项目经理的掌控之中。

1.6.5 ROI

我接触过的客户中,很少有人会去度量ROI(即投资回报率)。毕竟,要操纵数字太容易了。 不过,如果管理层要求的话,项目经理可以估算项目的回报,试着计算ROI。在项目完成后,可 以看看是否达到了当初预计的数字。

1.7 理解质量对于项目的意义

要想理解质量对于出资人和客户的意义,必须先要理解项目的关键驱动因素、约束和浮动因素。出资人是为项目提供资金的人,客户是使用产品的人,他们不一定是同一群人——对质量的定义可能有差别。没错,这让项目经理的工作更不好做了。不过,知道了对于项目来说"质量"意味着什么,也就部分理解了"完成"的含义。

温伯格这样说:"质量,就是对于某人的价值。"这个定义也就意味着可以为项目产品添加新功能,并可查看一个功能吸引(或排斥)了多少个(以及什么样的)"某人"。在想到出资人和客户的时候,不妨考虑一下他们希望从项目中得到什么。这样一来,项目经理便可以根据实际情况调整发布时间、项目预算和人员配备;同时,项目经理也可以权衡:是否所有的"某人"都需要某个功能。如果项目经理和团队知道"某人"对于质量的定义,大家就可以朝着这个方向来努力;即使他们改变了主意,团队仍然可以取得成功。

根据项目产品所处的不同市场应用阶段,如图1-2所示,客户对于质量的定义也有所差别。

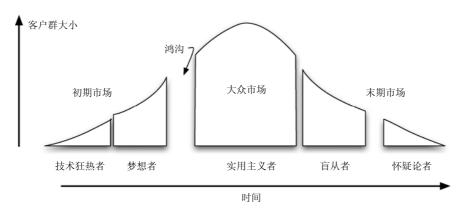


图1-2 摩尔的鸿沟

当产品处于市场应用的早期阶段时,顾客群的规模不大;但是技术狂热者们已经想先睹为快了,所以会面临比较大的发布压力。此时产品不必具有很多功能,而且也不必有很高的稳定性,但是它必须具备自己的独特之处,来吸引更多客户。

在产品进入市场的初期,无论客户群大小,人们都会有各自不同的需求。所有客户都希望马上得到新版本,而且要具备他们自己想要的功能。此时的产品用起来不能太费事,不过由于只有你的产品能够解决客户的问题,所以即使有点儿缺陷,它的销路还是会很不错。

一旦产品(或有替代产品)进入到大众市场之后,实用主义者们就会关心产品的缺陷能否得 到及时修复。如果在之前的版本中累积了技术债务(请查看原书附录B),现在就到了还账的时候 了。鉴于实用主义者拥有强大的购买力,管理层会迫于压力而决定频繁升级产品——即使有些客 户不想安装最新的版本。在加强产品稳定性的同时,每次发布的新版本中多少要加入一些新功能。

保守主义者也会购买你的产品,但往往是迫于某种压力作出的选择。如果产品不能完成承诺 的功能,或者有太多缺陷,保守主义者们会抓住一切机会发起抨击。他们不需要新功能,只希望 承诺的功能好用就行了。此时只要发布缺陷修复补丁,或是发布更加可靠、更高性能、更稳定的 产品即可。

对于你的产品, 盲从者和怀疑论者有可能买, 也有可能不买。处于这个阶段的产品有时会被 称为"现金牛",因为此时产品为公司带来的收入要远远超过需要付出的成本。

尽管在产品生命周期的开始阶段发布的很多版本都会面临不小的压力, 但在产品生命周期的 后续阶级,规模日益增大的客户群对低缺陷率要求的压力会更大。

₩ 铭记在心

- 每个项目启动时都要有章程。
- □ 对项目章程的反复修改要有心理准备。章程不一定完美,它的意义在于帮助整个项目团 队进行规划活动。
- □ 要知道"质量"的意义以及项目的驱动因素。这样随着项目的不断推进,项目经理和团 队才可以作出正确的决策。

规划项目

2

现在你手里有项目章程了。要是团队成员不能跟你一起编写章程,那就在项目启动会议上跟团队成员一起把它过一遍。要是团队成员已经熟悉了章程,那就可以跟他们一起做些有目的的规划和日程安排工作了。

规划和日程安排是两种不同的活动。规划是指制订带有发布条件的项目计划,而日程安排则是对工作项目的有序描述。

2.1 踏上征程

有了项目章程,每个团队成员就可以对自己接下来要干什么做些有明确方向的预先规划——或者,也可能提早知道自己还没有明确的方面。有了项目规划,就可以把团队成员的注意力聚集到预期的项目产出上来。

如果团队的人不多,比如不超过十个人,不妨与他们一起完成项目规划。跟团队成员商量接下来的几天或几周之内要做什么,同时要将项目的产出牢记心间。如果已经收集到一些需求,团队就可以开始原型化的工作,或是展开第一次迭代的开发工作了。要是采取敏捷开发过程,不妨将项目规划看作发布计划。

不过事情不会这么完美,比如人手不够,或者有太多的人参与原型化,甚至你根本不知 道接下来该干什么。此时,可以让一些人或全部团队成员去修复上个版本留下的缺陷。

2.2 使项目足以启动的规划

章程有了,规划是什么?管理层希望知道团队什么时候开发哪些特性。如何测量进度?项目何时完成?

规划不必完美无缺。实际上这也是做不到的。只要这个规划能让项目启动起来,同时能让大家看到成功的希望,就可以了。如果项目面临时间的压力(我接手的项目大多如此),那就要用时间盒来辅助规划活动。即使打算重新规划某个项目,项目经理也不必在一开始担心规划是否完美。

时间盒(timebox)是指特定的时间长度,个人或团队用它来完成某项特定的任务。个人或团队在这段时间内完成的工作量,就是项目接下来的工作的基础。如果有必要,个人或团队可以减少工作范围,以保证在"时间盒"内完成工作。

用时间盒来限制启动规划活动

——史蒂夫,项目组合管理高级总监

一般来说,我们会同时进行2~5个大项目以及15个左右的相关小项目。对于绝大部分项目来说,我们在启动时会先预计到底需要多少人力以及将会耗费多少时间。即使是为一个新项目做初始规划,也要保证耗时不长。

我们最近的一个大项目,预计耗时两年左右,需经历多次发布。我们把制订初始规划的时间盒设定为三天。三天过后,我们有了项目章程、项目规划,并制订出了第一次发布的发布条件,以及前三周的工作日程安排。就是这些,不过已经足够启动项目了。

我们最近的几个小项目(耗时不到半年),制订初始规划的时间盒设定为一天,其中包括安排一个为期两周的工作日程。当项目工作展开后,再根据执行初始规划得到的反馈,指导后续的规划。

有些项目已经完成了某些特定的功能,需要预测这些项目的发展和走向。我们会在项目前几周内进行初步的预测,后续几周内更新这些预测。到了8~12周的时候,对于何时完成哪些功能,我们心里也就有底了。我们没有做到尽善尽美,但是也不会在没有数据的情况下,浪费时间做过多无效规划。做一点规划,收集一些数据,再重新规划。这样做无往而不胜。

即使项目没有面临时间压力,要是为了得到"完美"的规划而花费太多工夫,时间的要求也会使这个项目变得越来越紧迫。

提示: 要根据经验而不是预言来规划项目

经验式规划也就是指不妨先做少量规划,再根据实际过程中收集到的信息反馈来影响未来的规划,这样做具有很高的可行性。预言式规划则是试图预测未来,是很难奏效的,除非你有水晶球。应该尽量在项目中使用经验式规划和日程安排。

也许你听过艾森豪威尔那句名言,"规划毫无用处,但是制订规划必不可少。"^①项目有太多的风险和不确定性,想在一开始就把一切都想好势比登天。从规划开始,每隔几周再重新规划,就像5.6节所述。无论采取何种生命周期管理项目,都得做好反复规划的准备。不要创建完美的规划,只要这个规划在被(很快)替换之前具备可行性就可以了。

提示: 以终为始

开始规划时,要想清楚有多少内容需要写下来。发布条件(请看2.4节)可以使出资人和项目团队集中注意力。如果想减少制订规划的时间,至少要保证规划出发布条件和风险列表。必要时要反复修订规划。如果使用敏捷生命周期来管理项目,就不要做任何预先规划活动了。

2.3 开发项目规划模板

项目经理要用项目规划来整理想法,最好选择能在组织的更多项目中重用的形式。下面是我的项目规划模板:

- □ 产品意图
- □ 历史记录
- □ 发布条件
- □目标
- □ 项目组织
- □ 日程总览
- □ 人员配备(人员曲线)
- □ 建议日程
- □ 风险列表

2.3.1 产品意图

简单描述产品,为什么公司要开发这个产品,它能为公司带来哪些效益,等等。发布版本的价值何在,它是否是后续发布版本(用三四句话)?如果已经在章程中写下了项目的远景,不妨在这里借用一下。

有时候,章程中的远景对于项目来说不够用。或者说,远景是供大的项目工程(一系列或者

① 1957年11月14日,艾森豪威尔在华盛顿特区国防行政准备会议(National Defense Executive Reserve Conference)上的讲话。

一组项目,请看14.1节)使用的。这类情况下,要确保当前项目的意图清晰无误。

多年以前,在TCP/IP协议还没有被整合入操作系统之前,我管理了一个工程,其产出是一个软硬件结合产品,其中包含了许多特性,每个特性都是一个单独的项目。我为工程制定了章程,其中的远景描述如下:"及时发布产品,供某某商业展使用。"有六个项目团队,网络团队是其中一个。他们对项目的意图定义是:"确保TCP/IP在第一次集成之前就能正常运作,并在整个项目中一直发挥作用。"他们的意图很清晰,而且与工程的远景相关,同时又非常明确。

基于项目(或工程)的规模和持续时间,每个项目团队(或子项目团队)都有其自己的意图。 这个意图应与工程章程的远景在大方向上保持一致,但又不完全相同。

2.3.2 历史记录

如果是在管理某产品的后续版本,比如4.2版本后的4.3版本,就要复查之前或相关版本的历史记录。这个历史记录可以说明之前任何已知的技术债务(见附录B)。要复查的数据包括:发布频率、发布后发现的缺陷数、客户报告的问题,以及会影响到项目经理对质量的定义、对项目管理方式的任何内容。

提示: 知道的越少……

对项目情况知道得越少,感到惊讶的几率也就越大。这适用于技术债务、新的架构、开发或测试风险或者制订规划等多个方面。只要在之前的项目中没有积累类似经验,就很容易遇到意外的情况。做一些规划,并做好反复规划的准备,这可以助你一臂之力。

如果客户习惯于每年发布一次产品,并且不会提出更多要求(按照摩尔的说法,他们已经是"大众客户"了,请看图1-2),此时项目经理有多种选择。第一种是拒绝管理层对频繁发布的要求。除此之外,可以使用版本火车或敏捷生命周期来进行更频繁的发布。如果需要打开或进入新的市场,在组织项目时,要考虑使用适合产品频繁发布的方式。

复查已发布版本中发现的缺陷的数量和类型,就可以理解即将使用的代码库中技术债务(请看附录B)的严重程度和类型。

如果客户在某个领域中报告了大量的问题,要看看到底是文档还是代码的问题(或是其他类型的问题)。对即将遇到的技术债务了解得越多,尽早掌控它们的几率也就越大。

2.3.3 发布条件

要详细列举出项目产品的关键可交付物。想识别出它们,不妨问一问:"要是不那么做,我们还能发布产品吗?"要将功能、性能和质量要求都涵盖在内。要想了解更多细节,请查看2.4节。

2.3.4 目标

项目经理在撰写项目章程时,也许已经对目标有所了解了。在准备好制订项目计划时,也许对目标已经了解得更多了。如果在制定章程时不知道任何目标,这时应该好好想想了。

已知的目标也许隶属于以下几类。

- □ **产品目标**也许包括这样一些需求,它们已经被设定好优先级,但是不承诺在当前发布版本中完成。这个列表也许已存在于产品的待办事项中。
- □ 项目目标也许是诸如性能标准之类的目标,对它们的要求会高于一般需求,或者是"在产品交付时,要将未解决缺陷的数目从50个减少到40个"。尤其是在管理一个工程的情形下,每个子项目的目标要特定于该项目所在的领域。项目团队要解决某些特定的技术债务,也许也可以作为项目的目标。
- □ **团队目标**可以是"增加产品的自动化冒烟测试所占的百分比"。团队也许希望改进某个特定功能的性能或可靠性。
- □ 组织目标可以是"减少项目的耗费时间,以提升组织的敏捷性"。^①

如果目标已经在项目章程中详细列出,在项目规划中就不必再写进去了。二者选其一,将项目目标明确出来即可。

2.3.5 项目组织

要明确说明团队在项目中的职责分配,指明项目经理如何使用生命周期组织项目工作,要采纳哪些关键实践,以及是否有决策人可以影响当前项目。如果你知道(或者怀疑)需要一些原型化的工作,要在这里说明。"史蒂夫和詹妮将会为那个功能集合搭建架构原型。比尔会进行同步的测试工作。因为时间紧迫,项目团队会选择可以让我们持续复查代码的技术。"

要说明项目的一般运作方式。比如,在项目启动时加强整个项目团队意识,招聘新人,开发包括代码和文档在内的完整功能,编写所有的代码,同时检查一下(在那个时间)可以记录些什么,诸如此类的事情。

此处的实际发生细节,取决于项目所采取的生命周期模型和团队的结构。要是有一个产品负责人专门负责做出与产品相关的所有决策,可以考虑是否要给他一个任命。如果有多个人同时决定功能特性以及彼此之间的权衡,就应该把这些决策人和他们负责的领域都列出来。

如果使用的是受时间盒限制的迭代,要说清楚每个时间盒的长度是多久。如果使用版本火车

① 感谢Bill Ramos提供该示例。

的方式,要说明每列火车持续多久。

2.3.6 日程总览

应该创建一个日程总览,其中标有主要的里程碑,还要说明人们从这些里程碑处可以得到什么。如果使用迭代或增量式开发,要解释迭代(或增量)的持续时间,并说明在每个迭代(或增量)结束后可以预期得到哪些产出。不要把工作分解结构(WBS)放在这里,要是有的话,可以给出指示,让希望了解更多细节的人去自行查看。无论使用何种生命周期模型,如有beta测试或是早期的客户发布版本,要把这些作为里程碑对外说明。

工作分解结构(Work Breakdown Structure, WBS): 它是任务的组织形式,展示它们之间的依赖、持续时间和所有者等信息。WBS级别越高(在项目中出现得越早),能得到的信息就越少。WBS是会随着项目进度而演变的。

日程总览让我的领导知道项目的实际进展

——特里, 项目经理

目前我正在管理第二个项目。领导们不相信这个项目会比上一个需要更多的客户参与。我编写了带有里程碑的项目总览,向他们说明情况。下面就是它大概的模样。

日期	里程碑
7月1日	项目启动。
7月15日	向露辛达展示Web界面的原型。(待续)
7月30日	向露辛达和厨房职员们展示厨房界面的原型。
8月15日	内部交付Web和厨房界面。
8月30日	向露辛达挑选的5个客户交付早期版本(beta测试)。
9月1日	开始beta测试。
9月30日	结束beta测试。
10月30日	系统上线,包括所有的客户订单和厨房的界面。

当我的领导们看到我在做什么之后,他们就意识到为什么我要提出对时间、人力和客户接触的请求了。

对于复杂的项目来说,如果在开始之间就已经感觉到日程安排上的风险,不妨考虑多准备几个方案,让出资人进行选择。图2-1中提到的组织就是基于矩阵的方式来组建项目团队的。

项目经理要确保自己可以理解项目的价值,而不是只看到不同方案的成本。如果交付的产品不能达到发布条件,其相关的方案也就不能提供足够的价值了。

不同方案	实际时间	架构时间	开发时间	测试时间	文档时间
功能1和功能2	2个月	1人月	6人月	6人月	2人月
功能1、2、3	4个月	2人月	10人月	10人月	3人月
功能1、2、3、4、5、6	12个月	6人月	60人月	50人月	8人月

图2-1 以矩阵方式给出项目团队的不同运作方案

2.3.7 人员配备

很多项目经理不能控制项目团队的人员配备。如果在项目开始的第一天就把所有的人都召集 到位了,那么出现人员变动可别吃惊。如果需要从其他组或是团队中调动人手,要在这里说清楚: 要在何时需要多少、何种类型的人员。请看第7章,以了解具体措施。

2.3.8 建议日程

项目经理要根据理解程度,列出主要的里程碑。如果有反映最初日程安排的甘特图,要在此处加入访问指示。当然,在重新规划后,要记得更新正在使用中的甘特图,以及时反映项目当前状况。将使用中的甘特图与项目的规划分开,这样更容易更新。

我喜欢使用黄色的即时贴来安排日程,所以在我的项目中很少使用完整的甘特图。我也因为说了下面的话而被大家所知:"要想了解当前最新的WBS,去项目房间的墙上看看吧。"(如果没有项目专用房间供张贴甘特图使用,就要使用公共区域,来让大家都能看到日程安排。这样一来,如果日程有问题,项目团队也可以早点提醒项目经理。

波浪式规划方式用得越多,在第一波之后需要加入的细节就越少。

提示: 小心过早细化日程

要反复修订项目日程,随项目进程补充细节。请看第4章,以获得更多信息。如果过早地向 出资人提供了非常详细的日程,他们就会认为项目中几乎没有风险。他们会注意到日程中的项 目结束日期,并以此作为项目真正的结束日期。

21

2.3.9 制订项目风险列表

在项目规划中,至少要将排名前十的风险记录在案。还要经常监控这些风险,并在适当的时机更新这个列表。如果觉得项目目前的风险不到十个,不妨跟项目团队一起坐下来,进行一次头脑风暴。

要尽早开始识别和管理风险。图2-2是我从一个真实项目的启动过程中拿过来的风险列表示例。

风险序号	风险描述	发生概率	严重程度	暴露程度	反应时间	应对计划
为每个风险 排定序号	用一个短语 或一句话为 风险命名	风险发生概 率	如果风险发生,会造成 影响的严重 程度	发生概率乘 以严重程度	应对风险的 时间	处理风险的 计划
1	除非到了项目 后期,否则我们 不知道算法的 速度是否够快	50-50 (中等)	高	(M,H)	7月14日	增加测试人员,与算法开发人员一起进行测试
2	启动超过2分钟	低	高	(L,H)	8月21日	在每次构建 时监控启动 过程

图2-2 风险列表

随着项目的进行,要更新风险列表。不妨使用原书第8章和第9章中的实践,来帮助规避风险。

2.4 制订发布条件

发布条件会告诉你项目中"完成"的含义。有些项目经理会在下面这些状况下发布软件:某个资深经理说该发布了、测试人员"核准"了,或者是人们觉得时机成熟了。问题是大家对于"完成"的定义各不相同。

制订发布条件不是为了责怪哪个组的人,而是要为产品是否可以发布提供一些客观的评判标准。出资人和客户可以用发布条件作出合理的决策,对产品的质量和风险做出判断。

制订发布条件时,要先判断当前项目的关键因素是什么。例如,为什么公司要做这个项目?

为什么客户会买这个产品?

利用发布条件,项目经理可以将整个产品的职责分工贯穿到产品的发布之中。比如,销售人员能够卖出满足这些条件的产品吗?技术支持人员能够为这个产品提供支持吗?培训人员能否制订培训计划并展开相关培训?当项目经理与组织各个职能部门的人共同商讨"成功"的含义时,他们会意识到自己不只是完成了分内的工作,而且也为项目经理指明了成功的方向。

项目经理一旦知道了成功的含义,就可以定义这个项目最重要的因素——发布条件了。

使用下列步骤来制订和使用发布条件。

- (1) 确定当前发布最重要的因素,这样可以将监控发布条件的活动贯穿项目始终。
- (2) 草拟发布条件。
- (3) 让发布条件符合SMART原则:确定的、可测量的、可达成的、相关的和可跟踪的。请看 2.4.3节。
 - (4) 获得项目团队与高层管理人员认可。

2.4.1 确定当前项目最重要的因素

当前项目的关键因素是由组织的需要和客户的需要共同组成的。客户在购买产品时,不会考虑其中有无缺陷或是缺陷数目,而是要看到这个产品解决了困扰客户的某个问题。在制订发布条件时,是应该考虑缺陷或缺陷水平,但是要确保发布条件中不仅仅包括缺陷相关的内容。想想你要为客户解决什么问题——无论是内部客户还是外部客户。

有时,发布日期是最重要的;有时,某个功能或一系列功能决定了项目的成败。通常来说,日程安排、功能特性和低缺陷率共同构成了项目的关键因素。就像1.2节中提到的,这些都取决于客户和他们的期望。

丽塔在一个依赖于现金流转的创业公司工作。公司的资金尚未完全到位,所以非常希望能早日交付产品,这样才能获得足够的现金维持公司生存。在过去的前三次发布中,唯一的发布条件就是发布日期,产品必须要交付给客户,公司才能在法律上确认产生收入。一旦公司成功熬过前几年,并且资金也全部到位后,就会加入其他的发布条件,诸如缺陷数目、测试过程、代码冻结状态(开发人员停止修改发布版本代码的次数)。

丽塔是这样说的:"当我们还处于创业阶段时,必须要保证把头伸出水面,维持生存。最开始的客户非常想要我们的产品,而且愿意一起工作。去年的版本发布后,其效果让我们都觉得非常惊讶。突然之间,客户开始关心缺陷问题了,他们之前从未如此关心过。管理层想知道我负责运转的测试组是什么样的状况,这使我压力倍增。唯一能让我舒心的,就是我已经与整个项目团

队和高级管理层事先确认过:早先设定的发布条件可以满足要求。但是我们没有意识到,当初没有完全弄清楚这个产品的需要。现在我们明白了,不能仅仅使用日期或是缺陷、测试数目来判断版本是否能发布。我们需要从大局出发,这样才能知道应该何时发布软件。

2.4.2 草拟发布条件

事先草拟一些发布条件是很有用的,这样就可以以它们为基础展开讨论。(如果有测试经理参与项目,项目经理可以请他完成或是与他一起草拟发布条件。)制订条件时,如果条件许可,要在上市时间和客户需求、缺陷状况、性能和可靠水平之间达成平衡。没有必要第一次就把所有的条件都订正确,只要能给大家一个讨论的基础就行。

在草拟发布条件时,要在纸面上注明"草案"字样。应该使团队成员和出资人知道这些只是供讨论用的条件草稿,而不是已经做出的承诺。

丽塔在开始时写出了一系列发布条件的草稿,供项目经理和团队成员讨论之用。下面就是丽塔的条件草稿:

- □ 所有代码必须针对所有运行平台编译并构建。
- □ 没有高优先级的bug。
- □ 所有未解决的bug和临时解决方案都要记录在版本发布说明中。
- □ 所有计划好的测试都要运行,要保证通过率至少为98%。
- □ 在最后三周内,未解决缺陷的数目要不断下降。
- □ 在发布之前,功能X要由开发人员完成单元测试,由测试组完成系统测试,由客户A和客户B完成验证。
- □ 准备6月1日发布。
- □ 所有未解决的缺陷都要由跨职能团队进行评估。

不过,当她与项目经理(下面简称PM)讨论后,丽塔意识到,她的初始条件并不完全符合客户或公司的要求。没错,客户很关注缺陷问题,但他们还没达到丽塔所关心的这种程度。实际上,只要几个主要客户对发布版本满意,其余的客户也就没什么大问题了。

刚开始的时候,对于丽塔能够从客户的角度出发看待整个发布,并能为整个发布提出一个照顾到各方平衡的方案,她的PM非常惊讶。他原本认为丽塔会更多地从传统测试经理的角度出发,力图提升质量,降低缺陷率。但是,丽塔从之前在公司做项目积累的经验知道:在对某次发布做决策时,低缺陷率只是考虑因素之一。

丽塔与PM和其他团队成员一起,修订了发布条件,并得到如下列表:

- □ 所有代码必须针对所有运行平台编译并构建。
- □ 没有高优先级的bug。
- □ 所有未解决的bug和临时解决方案都要记录在版本发布说明中。
- □ 所有计划好的测试都要运行,要保证通过率至少为90%。
- □ 在最后三周内,未解决缺陷的数目要不断下降。
- □ 在发布之前,功能X要由开发人员完成单元测试,由测试组完成系统测试,由客户A和客户B完成验证。
- □ 准备6月1日发布。

这些条件没有覆盖丽塔对于发布版本的全部要求,但是它们已经说明了对于当前发布的所有关键因素:发布日期、足够好的软件、一个已经完成测试并由两个客户验证通过的特定功能。测试组计划好的测试只要求通过90%,丽塔对此有些失望。不过在听到其他人的说明之后,她接受了这些发布条件,因为发布日期更重要。对于发布条件中不再要求在临近发布时由跨职能团队评估未解决的缺陷,丽塔也有些担心。不过,产品经理向丽塔保证,他已经跟PM讨论过这个问题,而PM将会把相关意见转达给市场和支持部门。

2.4.3 让发布条件符合 SMART 原则

在草拟发布条件时,要确保团队每个人对其的理解完全相同。我使用SMART原则——确定的(Specific)、可测量的(Measurable)、可达成的(Attainable)、相关的(Relevant)和可跟踪的(Trackable)——来检查条件是否合理、客观。T表示可跟踪性,这可以让整个团队明白:在创建发布条件时,我们要能够在项目的整个生命周期中评估这些条件。

对于当前项目的产品来说,每个条件都应该是确定的。如果一个条件是可测量的,就可以根据这个条件来评估软件。发布条件并不是那种必须要花些力气才能达成的延展性目标(stretch goal),所以要让每个条件都是可以达成的。通过评估产品是否符合客户和管理层的要求,来确保发布条件的相关性。如果条件是可跟踪的,那就可以在项目进行过程中评估条件的状态,而不只是在项目的最后一周才这样做。

"搜索必须在5s内返回第一组结果"是一个客观而且可测量的条件,可以对它展开测试。"所有未解决的缺陷必须由跨职能团队复查"是另外一个客观而且可测量的条件的例子。跨职能团队是否完成对未解决的缺陷的复查,其结果是确定无误的。

"性能要好"就是一个模棱两可的条件。要把它改成客观而且可测量的条件,应该这样描述: "性能场景A(对应于用例A)要在10s内完成。"要给出特定的场景,这样人们就可以引用它,并 为其提供度量方案,团队也可以知道他们是否达成了性能要求。

2.4.4 在发布条件上达成多方共识

得到了合理的发布条件之后,就该取得各方面对它的共识了。如果有人对草拟的条件有负面意见("不,我们做不到"),要找到他们担心的原因。产生发布条件的过程,也是揭示众人对于产品和项目的假定和忧虑的过程。在就发布条件获得大家共识的过程中,可以提下面这些问题:

- □ 在发布日期之前,我们是不是必须满足这个条件?
- □ 要是我们不能在发布日期之前满足这个条件,会发生什么?
- □ 如果不能满足这个条件,我们的产品或公司是不是会因此而承担风险? 人们的安全感是不是会因此被破坏?

通过回答这些问题,整个团队就会更加注意当前发布版本的要求。

可以利用下面这些方式来获得大家的共识:草拟发布条件,针对其展开讨论,并在团队会议上就其达成共识;与整个团队草拟发布条件;与团队中的各个职能经理一起草拟发布条件。有了发布条件草案后,在项目团队会议上将其作为一项讨论议题。我喜欢与整个团队一起讨论并产生发布条件,因为这样一来,整个团队都会对其形成归属感,而不仅仅是我或管理层。不过,要是项目规模很大,或是团队之前从未用过发布条件,而且希望每个成员都能理解工作目标的话,还是先试着在团队会议之前草拟一份发布条件吧。

一旦获得多方共识,就可以用这些条件来监控项目了。

2.5 使用发布条件

到产品发布的时候,发布条件只能有"满足"或"未满足"两种状态。不可能部分满足某项条件——其实就是没有满足。在项目经理与高层管理者讨论软件产品目前的状况时,这种二元的方式是很有用的。如果只是说部分完成了,他们会认为你已经完全搞定了;如果说还没有满足某个条件,他们就会认为你还在努力工作。

在团队会议中,要跟团队一起讨论距离满足发布条件还要多久。整个项目过程中,在进行讨论的同时去看看项目仪表板(见第11章)的状态,整个团队可以评估当前项目的完成度是多少。如果项目有正式的系统测试阶段,可以将发布条件作为测试状态报告的一部分。

发布条件可以作为早期的警告信号,让大家知道团队可能无法准时发布当前版本。曼尼是一个项目经理,他所带领的六个月的项目目前进展到一半。曼尼评估了距离发布日期的进度,担心团队无法按时交付。他打算跟团队一起制订发布条件,让大家更加了解这次发布要达成的目标。

在接下来每周的项目团队会议上,曼尼用发布条件来确认项目在不断取得进展。这样连续做了几周,效果都不错。直到有一周,在评估发布条件时,一个工程师说:"我做不到。我试过很多种方法,看来还是无法在交付日期之前满足这个条件。"曼尼说:"那好,我会去跟上面说说看能做些什么。在去之前,还有人觉得哪些条件不能满足吗?"另一个工程师说:"在发布之前,我无法让性能做到如此优秀。咱们当初讨论发布条件的时候,我觉得是没问题的,但现在我知道那是无法做到的。"

使用发布条件,曼尼可以得到项目进度的一些早期数据。对于这个团队来说,在进行到三分之二的时候知道他们无法达成发布条件,总好过在最后一刻才发现问题。项目经理在了解到项目的真实状况后,可以就此与管理层沟通,看看在哪些方面进行折中,可以产生最现实的结果。在这个例子中,为了让产品满足发布条件,曼尼可以就发布日期进行重新谈判。

理想状况下,项目经理可以随着项目进度评估发布条件,并在项目预期结束时满足这些条件。 不过人生不如意十之八九,项目也是如此,发布条件不可能总是能够达成的。发生类似情况时, 要坦诚面对现实。

在必要时变更发布条件

使用发布条件,可以让项目关于"完成"的定义保持稳定。不过,在下列情况发生时,项目经理要考虑改变发布条件:进一步了解了这个项目中"完成"的含义时;认识到无法在预定发布日期前满足所有发布条件时。

如果进一步了解了"完成"的含义,不妨问问自己之前关于发布条件的问题:我们是不是必须要满足发布条件?要是没有满足发布条件,会发生什么?请看2.4节。

假设项目经理所在的项目无法满足发布条件。首先,要跟团队确认为什么无法满足条件。接下来,向管理层解释无法满足条件的原因。促使管理层向项目团队解释目前的状况,就像这样: "我们曾经认为这些条件很重要,但是现在知道发布日期是更重要的。我们将会准时发布产品,即使不能满足所有的发布条件。"如果事实如此,项目经理可以让他们补充: "我们会找出这次不能达成发布条件的原因,并在下个项目时注意不再发生同样的问题。"如果不向团队解释,他们会觉得自己正在玩日程排定游戏。

₩ 铭记在心

- □ 项目规划是在不断进行的,这只是开始。
- □ 为项目团队、出资人和项目经理自己制订发布条件,以明确定义"完成"的含义。
- □ 项目规划不必完美无瑕,但是它必须存在。

识别和避免日程安排游戏

人使项目经理自己努力做好估算、规划和日程安排工作,你遇到的出资人、管理层和团队成员还是有可能视日程安排为儿戏。项目经理要把这些人带回现实,不过首先要学会识别这些日程安排游戏。

所有的出资人和管理层都会逼你在日程安排上做出一些让步。即使你制定的项目日程已经相当合理了,他们还是会玩这样的游戏。不过他们抗拒的方式很容易识别,很少脱离几种固定的模式。项目经理只要能够识别出他们所玩的游戏,就可以更容易地掌控项目,得到理所应当的产出。

3.1 给我一块石头

克里夫与团队一起,用一周时间制订出了项目日程。他们完成了"哈德逊湾式启动"(见4.2.4节),并且确定已经识别出了主要的技术风险。他将风险和日程安排告诉了他的上司诺姆。"你就不能再早点完成项目了吗?"诺姆的一句话将克里夫送回了团队,步履蹒跚。

克里夫与团队又花了一周时间修改时间表,得到另外一个日期。他走进诺姆的办公室,说道:"如果你能在这里和这里为我提供更多的人手",他指着几个里程碑,"我就能用一个月时间完成项目。"诺姆皱着眉头说道:"还不够好。我需要这个项目早点儿结束。"克里夫叹了口气,又回到项目团队中去了。

又过了一周,克里夫拿着另一个日程来找诺姆,"好吧,这就是我们力所能及的结果了。"克里夫说。

诺姆几乎连看都没看,就说道:"但是还是不够好。"

克里夫暴怒道:"你到底想要什么?"

"给我一块石头"(见图3-1),这就是诺姆玩的游戏。不管你制订出什么样的日程,你的出资

人总是希望项目能更早完成。你只会发现:出资人不会认同你提出来的每一个截止日期——你的日期总是离他们的期望值很遥远。



图3-1 给我一块石头

当"他们"希望项目能更快交付,但是不告诉你何时需要或为什么的时候,就会玩"给我一块石头"游戏。如果他们告诉你期望截止日期,你就能告诉他们能完成哪些工作。如果他们告诉你原因,你和团队也许就能想出一些创造性的解决方案来满足他们的要求。

讲求实效的项目经理自有应对之策,其中就有克里夫采取的谈判策略。一旦谈判失败,或者 看起来永远难以成功,不妨试试下面的方式。

- □ 在试图取更多石头之前,先提几个问题: 你喜欢短的日程,还是长的日程? 是要更多的人,还是更少的人? 要是少实现几个功能会如何? 先知道什么是最重要的,这会帮项目经理产生更加合理的解决方案,多少也能为你的谈判做些准备。
- □ 找出是什么原因促成了他们期望的截止日期。探寻出这个项目的战略原因,并搞清楚"成功"的真正含义。

- □ 要让出资人明白你做出的选择以及背后的原因。也许出资人会有更容易操作、更快实现的好主意。
- □ 为你提供的日期说明信心范围。很可能管理层不明白你的估算意味着什么,而且你也有 可能不理解他们所要的东西。
- □ 在提供日期时要说明发布条件,这样你就可以提一些问题,了解他们对于发布版本的质量和功能的要求。我们可以让这个功能的性能极其出色,但是就得先放下那个功能,这样做可以吗?用户们可以接受带有更多缺陷的产品吗?

在一个组织中,"给我一块石头"的游戏会反复发生。很多时候,每个项目都会发生这种状况。如果你总是碰到"给我一块石头"的问题,考虑采用下面的实践。

- □ 制订排好优先级的产品代办事项列表。。
- □ 逐个实现功能。如果能够让出资人了解更多的项目进程,他们就不会那么纠缠截止日期 了。
- □ 使用短小的时间盒(持续时间少于四周),这样出资人就可以看清项目进程。如果你每隔 几周就能展示出项目的有价值的进展,截止日期就没那么重要了。你可以开始讨论何时 实现哪个功能,他们对质量的要求又是什么。

3.2 "希望"是我们最重要的策略

几年前,一位资深经理打电话给我,说道:"我们有个项目出问题了。在启动的时候,我们充满了希望,但是现在看来已经不可能了。"我问了几个问题,发现他们之前从来没有做过类似的项目。相对以前,这个项目的规模更大,使用新的开发语言,基于新的平台,而且日程安排更短。

整个公司的未来都押在这个项目的成功上,问题是它比以前做过的项目都要复杂,而且要求也更高。他们唯一的策略就是"希望"(见图3-2)。

他们没有安排任何关于项目所在领域、使用的开发语言或新操作系统的培训。他们对这个项目希望达到的时间要求,也是以前从未做到的。

仅有希望,不足以交付一个成功的项目。®

讲求实效的项目经理会按如下方式做。

① 我是从以斯贴·德比(Esther Derby)那里听到这个游戏的。



图3-2 "希望"是我们最重要的策略

- □ 识别风险并记录下来。风险可能来自于技术(新的开发语言、新的平台)、日程安排(时间过短、人太少),很多时候两者皆有。
- □ 不到万不得已,不要选择瀑布式生命周期。为什么?因为你没有任何数据足以成功支持 瀑布式需要的前期计划过程。要是你从未做过类似的项目,用迭代进行原型化,或者通 过迭代开发几个功能,看看会是什么情况。
- □ 可以用"哈德逊湾式启动"看看是不是能做出些什么东西。要使用即将使用的新的开发语言、操作系统、数据库以及类似新技术,这样做效果尤其好。"哈德逊湾式启动"能够让团队了解要做的东西,而且可以揭示一些目前尚未发现的风险。
- □ 确保大家具备相关的技术能力,还有解决问题必备的领域知识 [Rot04b]。有必要的话可以进行培训。让大家学习项目中要用的开发语言,这些投入比起白白浪费时间所付出的成本要低。
- □ 考虑所有工作都采取迭代的方式进行,特别是项目规划和日程安排。
- □ 由于缺少经验和专业知识,可以寻求相关的帮助和信息。跟团队成员一起商量如何能让 别人了解他们的工作进展。
- □ 制订里程碑条件(里程碑也可以是迭代的)。在管理层复审会议上审查这些条件。即使管理层或出资人不愿意做复审,项目经理也可以主导这些会议。如果不知道怎么样才能让

项目正常运转,可以按里程碑周期性审查项目进度。

不要指望仅凭希望就能得到好的结果。

作为项目经理,你的工作就是要计划、再计划,并努力工作,以得到最好的产出。以下这些 实践可以帮你达到目的。

- □ 使用有时间盒限制的迭代,这样所有的人都可以看到项目的进度。
- □ 使用速度图表展示项目进度。要让大家都能很明白地看到进度(或匮乏的资源)。这样一来,特别是在需要帮助的时候,这些数据可以拿来使用。

3.3 拒绝女王

有些老板就是不愿意面对现实。你告诉他们:"我们无法达到你在时间上的要求。"他们就好像根本没听见,看着你,然后告诉你:"我相信只要你上心,就一定能按时搞定。"你坐在那里哑口无言,他们却高高兴兴地走了,好像项目在他们给出的日期前一定可以交付。出现这样的状况,你就是碰见"拒绝女王"^①了(见图3-3)。



图3-3 拒绝女王

① 我第一次听说"拒绝女王",是从本森·马古里斯(Benson Margulies)那里。

有不少原因会引发"拒绝"。我所见过最常见的原因,就像上面提到的经理,他希望鼓励项目团队。有时,人们拒绝是因为他们害怕项目无法在截止日期之前完成,他们不会管你说什么,这就是鸵鸟效应。有时,公司高层相信:如果设定模糊或是不可能的日期,项目团队就可以早于他们自己预想的时间完成项目。

可以用下面的方式应对"拒绝"。

- □ 找出你的经理表示拒绝的原因。尝试用一些与上下文无关的问题,以此来理解项目背后的原因。比如,你可以问:"对这个项目来说,要怎么样才算成功?"
- □ 写下项目的风险及其潜在影响。用高、中、低来讨论严重程度和暴露程度,不要用数字。 不拿你制订的日程当回事的人,对这些风险数字也不会认真。
- □ 展示你所能做到的事情,并测量团队在项目中的实际开发速度。没错,这里用迭代是非常方便的,而且速度图表可以告诉别人项目的真实现状。
- □ 保证参与项目工作的每个人都有相关领域的专业知识。

如果管理层认为"拒绝"是鼓励团队的方式,建议他采取其他的鼓励技巧。通常,这意味着让他去干点儿其他对团队有益的事情(比如通过谈判缩小需求的范围),或者将他的注意力吸引到别的项目上去。管理层可能认为拒绝现存问题或潜在问题可以鼓励团队,其实这反而会对团队形成障碍。将他们的注意力转移到其他项目上是一种很有用的技巧,可以让他们不再总盯着你的项目。

只要项目经理能够接受现实,"拒绝女王"就不一定能造成灾难。

有个团队试图说服项目经理接受项目的现实。失败几次后,他们放弃了,并且决定自己组织起来,忽略项目经理的存在。他们不再参加项目团队会议,并且将项目经理的话抛在脑后。他们开发了项目的部分功能,并且产生了一些数据(不过不是速度图表)。几个月之后,大家都看出来项目经理根本不了解目前的状况,此人也因此被炒了鱿鱼。可这时,项目团队也已经损失了很多时间,他们很难再控制什么时候交付哪些功能了。

现实总是会在某个时候跟"拒绝"面对面,这不可避免,而且这就是"拒绝女王"为什么只是规划游戏,而不会总是造成灾难的原因。当管理层看到了真正的现实之后,项目经理要确保自己有部分可以工作的产品、可以展示的数据,让管理层可以跟你讨论接下来能够做些什么。此时也是讨论"有哪些事情可以不做"的好时机,这样你就可以完成当前的项目,并且为后续项目做更好的规划。

如果项目经理总是遇到"拒绝女王",那么在进行管理的时候,可以把下面的方法集成进去,大家也就可以看到实际进展了。

- □ 使用有时间盒限制的迭代,这样所有的人都可以看到项目的进度。
- □ 制订排定优先级的产品待办事项。团队可以按照功能的业务价值,从高到低逐个实现。 等到出资人如梦初醒,终于意识到项目无法按他给出的日期交付时,团队也已经开发完成了若干高优先级的功能,这样项目经理也会得到一些有价值的东西。

3.4 把灰扫到地毯下面

几年前,我接到一个项目组的求助电话。等我参与的时候,项目正处于发布周期的中期。我 跟团队成员一起识别出他们可以交付什么、应该交付什么、还有哪些应该推迟。项目团队后来也 按可交付物列表完成了交付。

发布之后,我建议团队召开回顾会议,看看有哪些工作是在下一次可以改进的。副总裁却认 为没有人能从回顾中学到东西。

他忘了我为什么要来帮忙,光注意到了大家的成功。他说:"不过大家这次干得很不错,满足了我们对这个版本的所有要求。"

他的话把所有的问题一扫而光——特别是优先级的变更——灰都被扫到地毯下面了[®]。请看图3-4。团队里可没人觉得自己做得有多好。副总裁的话不再可靠了。项目团队成员感到疲惫不堪。这个版本一开始要求的有些功能不做也是可以的,如果项目团队一开始就能知道这一点,开发人员就可以集中开发最需要的功能了。



图3-4 把灰扫到地毯下面

① 我第一次是从伊丽莎白·韩德瑞克(Elisabeth Hendrickson)那里听到"把灰扫到地毯下面"的说法。

下面这些主意可以避免这个游戏。

- □ 把某个版本需要的功能先按优先级进行排序,再实现。(排序意味着1、2、3、4、5、6, 诸如此类。)
- □ 逐个实现各功能。如果按照架构进行开发,就会形成很多部分完成的功能,没有哪个是 完全开发完成的。而且架构会随着项目进展演化,很多管理层也认识不到这一点。
- □ 制订发布条件,项目经理在项目开始时就可以讨论当前版本需要的东西。参见原书2.3节。
- □ 如果总是遇到"把灰扫到地毯下面"的问题,可以试试下面的方法。
- □ 使用产品待办事项列表,功能按业务价值进行排序。这样,你就总是在完成最有价值(也就是最重要)的工作。
- □ 使用有时间盒限制的迭代,并逐个实现各功能。项目经理和团队成员可以看到进展,并 且总是最先提供最有价值的功能。

要想利用上述规避策略,就得在项目开始时与项目干系人对话。这些对话很难进行。不过其有益之处在于:如果没有交付任何东西,没人会假装项目是成功的。相反,团队可以把精力放在为了成功交付必须要做的事情上,而且只做这些。

3.5 幸福日期

我遇到一些组织,他们有个不成文的规定: 绝不讨论项目日程。这种情况我见得太多了。管理层想要一个日期,项目团队就会说: "当然,没问题,就圣诞节吧!"但他们不会明说是哪个圣诞节。

最后,当某个圣诞节到来之时,已经过了不少日子了,人们开始讨论日程安排。而这时项目已经错过不少里程碑的交付日期,甚至有可能几个早先期望的项目结束日期都已经过去了。

我曾跟这样一个项目团队共事,他们在5年之内没有达成项目日程中任何一个里程碑或其他 任何要求。他们总是在反复制定优化的项目日程,却没有信心范围。最后,在资深管理层变动之 后,整个团队被叫去参加高层管理会议,解释日程为什么会如此糟糕。

有一个经理说:"你们看,我想知道什么时候才能有所成果。咱们找个出发点吧。"此时,项目团队就知道他们必须要做出改变了。

我必须承认,长久以来,我很难理解人们怎么会陷入到这种日程安排游戏中。不过,我确实见过善于口舌的管理层,他们或威逼,或利诱,或用权力来"说服"项目经理以及团队,让他们相信可以满足管理层对"幸福日期"(见图3-5)的要求。这种所谓的"说服力"加上逃避困难话

题的文化,足以让项目经理适宜进行"梦想时间"或"幸福日期"日程安排游戏。®



图3-5 幸福日期

"幸福日期"与"拒绝女王"有点儿联系,但是又不完全相同。有了"幸福日期",组织中有些人(项目团队或项目经理)就可以让另外一些人(项目干系人)感到心安。从某种意义上来说,大家都不认为日程有讨论的必要。项目团队想抚慰项目干系人,项目干系人也愿意得到安抚。而在"拒绝女王"游戏中,项目干系人也希望得到安抚,可项目团队会坚持告诉他们现实。

要阻止这个游戏的发生,你必须与组织一起在项目层面上工作。对于项目,可以采取下列措施。

- □ 说明日程安排范围,特别是在没有使用迭代生命周期的情况下。
- □ 使用迭代式生命周期,并说明将会通过信心范围实现哪些功能。("到下个月为止,我们可以完成这十个功能,也许还能再多完成三个。我们会在本月底之前告诉你。")
- □ 使用敏捷生命周期和经过排序的待办事项列表。
- □ 即便采取按阶段交付的生命周期,也要使用短小的时间盒,这有助于人们不断取得进展, 而且要让大家了解这些进展。
- □ 不要仅仅以里程碑日期作为项目的衡量标准。要想了解项目的真实状况,如果只使用单一的衡量标准无异于饮鸩止渴。可使用速度图表,让每个人都能了解进度。

不过,上述只是项目经理仅凭一己之力所能做到的。这个游戏揭示了组织的不一致性——大家只想彼此安慰,避免冲突。建设性的讨论(又名建设性的冲突)可以让组织更坚强。避免冲突

① 我是从蒂姆·李斯特(Tim Lister)那里第一次听到"幸福日期"这个词的。

和必要的讨论只会让组织变得更孱弱。

3.6 屁股着火

一天,一封来自公司大人物的紧急邮件把你招呼到公司。邮件里说:"别弄那个项目了,赶 紧开始做这个吧!"

可以确定,如果此事发生一次,它就还将多次重演。项目经理和团队都会周旋于几个项目之 中,或是在两个项目之间不停切换。不管是哪种状况,项目经理都陷入多项目、多任务切换的泥 潭中,而且整个团队亦是如此。项目经理知道自己无法取得任何成果,所有项目的紧急程度却在 不断攀升、攀升、攀升……

当管理层害怕或无法一次将精力集中在一件事情之上时,就会发生屁股着火(见图3-6)的 状况。有以下几种形成原因: 技术人员过去就曾延迟交付, 公司没有战略规划, 或者公司的战略 规划没有分解成足够详细的具体工作规划。 ①



图3-6 屁股着火

项目经理可以采取下面这些行动。

① 蒂姆·李斯特在与伊丽莎白·韩德瑞克谈话时,命名该游戏了。

- □ 按小时间盒迭代进行规划,在每个迭代边界都开始新的工作。要达到该目的,迭代必须 足够短,比如一周或两周,这样才有机会开始新工作。
- □ 如果无法管理迭代,就按功能逐个实现,使用按阶段式的交付。
- □ 将采取这种策略的成本告诉管理层,让他们选择是要解决时间上的危机,还是要付出经过计算的额外成本。了解如何帮助管理层计算成本和收益。
- □ 策略确定后,帮助公司检查相应的具体措施。可以这样问:"根据该策略,我们会得到这样的结果。这真的是我们想要的吗?"
- □ 项目经理可以调整估算方法,让团队更容易达到最初的估算日期。如果团队无法满足估算日期,管理层可能会觉得别无选择,只能让他们去做别的事情了。保证大家都用"小石子"式的方式,而且尝试使用持续集成,这样团队就可能完成一些工作。
- □ 有时,如果客户觉得还不需要目前的产品,也会发生"屁股着火"的状况。管理层觉得团队没有完成这个产品的必要,他们想让所有的人或者绝大部分人去参与别的项目。既然如此,要让大家以短期迭代方式工作,而且在项目启动时就要知道是否存在会导致项目推迟的原因。还要确保管理层已经制订出了项目组合方案,而且目前也在管理该组合方案。请参见原书第16章。

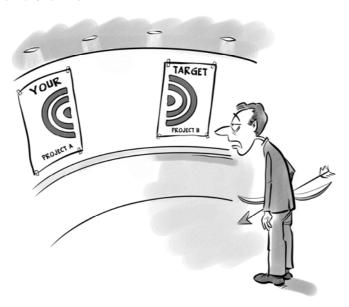


图3-7 分散注意力

"屁股着火"会浪费所有人的时间。不过,管理层有时就是无法改变他们的管理风格,或者就是不相信同时进行多个项目会浪费时间。如果你陷入此类状况,不妨考虑如何创建单一项目的工作环境,这样你和团队都可以成功取得进展。

3.7 分散注意力

38

我曾遇到这样一个经理,他的原话如下:"我希望你能在项目A上花50%的时间,在项目B上花30%的时间,在项目C上花20%的时间。在空闲的时候,你能看看给老大的报告吗?"我答道:"什么空闲时间?"

"分散注意力"就是与多项目、多任务相关的游戏。如果管理层无法把精力投入到一个项目或是工程的策略之中,就会引发此游戏。他们不会对每个项目说"是"或"否",也不会问"什么时候?",而是对所有的项目点头称是。

项目经理可以采取下面这些行动。

- □ 与工程团队一起尝试原书16.7.1节中的解决方案,特别是有助于工程团队或管理层团队判定先做什么、后做什么的方法。
- □ 如果管理层坚持要你和你的团队同时进行多个项目,可为每个项目设定一周的迭代,并确保在每个迭代结束时有可发布的产品。我使用一周的迭代,人们就能在这一周内将注意力集中在一个项目上。一周结束时,到了周末,人们就可以改变关注点了,大家可以不再考虑眼前的项目,而是开始思考下一个项目。
- □ 如果无法管理迭代,那就按功能逐个实现吧,使用按阶段交付方式。要保证每个项目都 有发布条件,这样就能完成每个项目最低限度的工作量。
- □ 就像在"屁股着火"中提到的一样,将这样做的成本告诉管理层,让他们知道要在时间 上的危机和付出额外的成本之间进行权衡。参见原书16.7.1节,以了解如何帮助管理层计 算成本和收益。
- □ 确保已经对需求进行过优先级的排定,而且可以赶紧完成一些工作。"分散注意力"游戏的发生,有时是因为一些项目干系人不相信团队能够按时交付。他们想让项目经理和团队同时做几个项目的工作,这样能让他们更快看到成果,并由此产生安全感。他们错了,却没有认识到这一点。

如果项目经理所在的组织经常"分散注意力",那就准备采用一周一次的迭代吧。没错,这样做很困难。你必须跟团队成员一起,将需求打散成足够小的部分,他们就可以在一周内取得工作进展。如果不能帮助团队转向在短期内开发小批量任务的方式,你们就永远无法取得成果。正如原书15.3.2节的文本框中所提到的,团队会经常产生延迟。作为项目经理,你有责任帮助大家完成工作,同时帮助管理层不要陷入"分散注意力"的泥潭。

集中注意力意味着只在一点上集中

我曾为一家公司工作,在公司外开完了几天的策略会议之后,大家决定:我们要同时"把

注意力放在五个方向上"。

这不是集中注意力。

集中注意力意味着将所有的精力都放在一件事情上。五个战略方向实在是太多了。

然而,对于公司来说,为了照顾到所有可能的客户,而把自己的精力分散开,去开发尽可能多的产品。此类状况太常见了。如果你所在的工作环境就是这种状况,赶紧采用短期迭代吧,越快越好。这样你就可以告诉管理层:"好吧,我们下周做那件事情,因为再有两天我们就可以完成这项工作了。"要想取得更好的效果,将迭代的开始和结束都放在周中,比如周三。这样的话,如果管理层在周末又突发奇想,他们也许还会给你几天时间完成现有的工作。

对于你保持注意力集中的努力,你的团队成员会感激不尽。

3.8 日程等于承诺

我们都知道,日程安排只是估计而已,不过是大概猜测罢了。项目日程是对于团队何时到达哪个里程碑、何时完成项目的最佳推测。日程安排并不是预言,仅是猜测而已。但是有些项目经理的出资人会将这个猜测视为承诺(见图3-8)。



图3-8 日程等于承诺

如果面对上述困境,可以问问如下两个问题:

- □ 你是否关心团队交付的东西?
- □ 你是否关心产品的质量如何?

要想让有关项目日程的谈话富有成效,就得讨论项目后面的驱动因素、约束和浮动因素;应该商量在截止日期之前,至少有哪些功能是必须要交付的,这些功能应该达到什么样的质量标准。如果相关人员还没有准备好讨论项目日程、功能集合和缺陷水平,那么任何关于日程应该作为承诺的讨论都显得为时过早。

当资深管理层希望得到承诺时,我喜欢用信心水平与他们沟通:"我有90%的信心在8月1日发布。如果允许在10月1日交付,我有100%的信心。"我会告诉他们这两个信心日期之间我们必须要做的事情,也因此而深入理解了将日程作为承诺对我来说意味着什么。

我还喜欢使用"交付日期渐进法(date-for-a-date)"。"我可以告诉你,下半年发布没问题。 现在我可以把时间限制在某个季度(给出一个日期),然后可以限制在某个月(另一个日期),接 下来(给出另一个日期)就是我们要发布最终版本的日期。"

不过我最喜欢的技巧还是使用有时间盒限制的迭代 (持续两到四周,不能更长了),同时使用按优先级排好顺序的待办事项列表。这样,就可以应对任何时间的发布要求。项目经理明白,每个迭代的成果都可以投入正常使用。项目经理也知道,最重要的需求已经先实现了。如果管理层想进行产品发布,没有问题,因为产品已经整装待发。开发人员不必再做出任何承诺,反而是项目经理需要从提供需求的人那里得到承诺,因为他们要说明哪个需求在何时需要。

如果有人要项目经理承诺一个日期,那就考虑如何组织项目吧。试试用迭代,或者尝试用"逐步逼近的日期",要不就用带有日期估算的信心水平吧。但是不要只承诺一个日期,这简直就等于是为墨菲敞开大门了[©]。你可以承诺一个日期,但是总会发生一些意外,让你的承诺变为水月镜花。

3.9 到了之后,我们会知道身处何方

有一个副总声称自己有"多动症"(Attention Deficit Disorder)。他不会要求不同的日期,实际上,他在不停改变项目的目标。一开始,目标是一个特定的功能集合。而从开发速度图表中可以看到,团队无法按时交付。这位副总就改变了目标,要提升某些功能的性能。但是性能提升在技术实现上很有难度。几周之后,他又决定把重点放在提高可靠性上了。

让我觉得有意思的是: 他在做项目经理时没有这个问题, 一点儿都没有。那时他很明确每个

① 作者此处是指墨菲定律(Murphy's Law)会发生作用。——译者注

项目都有一个确定的目标,而且他会尽量注意不让高层改变项目的目标。可当他自己处在高层之后,却不能让项目有明确的目标,这些项目也就很难完成了。

如果高层变更项目的目标,或是对于项目应交付的产品有更好的主意,或是对于项目脱离正常轨道有了更好的想法,就会发生这个日程规划游戏。我曾见过有些不靠谱的项目经理,他们就像高层所做的一样,导致项目脱轨,"哎,你看,那样做是不是看起来更好啊?咱们就那么做吧。"见图3-9。

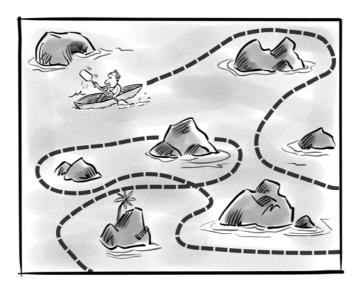


图3-9 到了之后,我们会知道身处何方

该游戏与"给我一块石头"不同。在那个游戏中,项目的目标不会随时间而改变,出资人想要的是更短的日程。在"到了之后,我们会知道身处何方"这个游戏中,出资人不想改变日期,而是希望改变项目目标。

当管理层无法或不想决定产品的方向时,这个游戏就发生了。

这就像是一个男孩总在等下一个最漂亮的女孩。他跟这个漂亮的女孩儿约会一段时间,如果 另外一个美丽的姑娘出现了,他会抛下这一个直奔新的目标。

无论如何称呼这个游戏,其效果都是一样的。项目团队无法将全部精力投放到可以交付的一个产品上。项目总是在变换目标。最近一个我以咨询顾问的身份接触的类似项目中,一个高层管理人安慰我说:"等我们到了之后,就知道我们在哪里了。"

项目经理可以考虑如下建议。

□ 确定已经有书面的项目愿景、项目目标和发布条件。就愿景、目标和发布条件得到大家

共识。项目经理知道未来的方向、还要做哪些工作、做到何时就算成功。组织的其他人 也要对这些了如指掌。

- □ 如果管理层不愿意定义愿景,那项目经理就要担起这个责任。完成后把它公布出来,并 坚守这个目标。如果做不到,就赶紧结掉这个项目,再用新的目标启动新项目。
- □ 如果项目会持续很长时间,就用迭代来组织项目。每次迭代完成后评估项目进度。如果已经达成了足够多的目标,就结束这个项目,再启动新项目。要用短的(不超过四周) 迭代。

有些情况下,上述措施都不能发挥效用,因为有些高层会干扰项目经理的工作,他们绕过项目经理,向团队分配其他任务。如果类似情况就在你身边出现,要跟高层沟通,告诉他们你能给他们带来哪些好处。如果他们不听,或者无法停止自己的行为,要记得自己并没有签卖身契。请参见原书7.7节。

项目需要清晰明确的目标,整个项目团队要把精力都放在这些目标上。不要允许别人让你的项目失去方向。否则你就会失去目标,到最后只能发现自己陷入一片迷茫!

3.10 日程安排工具总是对的,又称为梦幻时间日程

巴尼是一个项目经理,组织的高层只知道瀑布式生命周期。他们觉得迭代式的做法就是浪费时间。他们希望在项目的第一周就看到甘特图,这样项目经理就可以按照甘特图管理,一切都能按部就班进行。如此一来,无可避免的是,要是巴尼报告说项目没有按计划进行,有些高层就会这么说:"哎,按照日程安排,你的进度应该到这儿。没能按照计划进行,你是怎么回事啊?"

决策层对于项目的了解并不深入,他们不知道,人们在项目中是根据经验来思考和行动的。 他们相信,关键路径会永远保持不变,任务安排顺序也一直大体相同。

发生如此状况,原因在于:一直以来,决策层看到的报表是由已经完成的工作、销售数字或其他数据构成的,这些数字反应的是在过去发生的事情。然而,项目日程是对于工作未来进展的猜测。该日程游戏也被叫做"梦幻时间日程",参见图3-10。[©]

漂亮的甘特图会掩盖这样一个事实: 日程安排只是猜测。甘特图会让人们安心于日程安排, 从而忽略对现实的检查。

如果项目经理面对这样的管理层, 可以参考下面的建议。

① 艾丝特·德比和我在一次谈话中提出了这个命名。



图3-10 梦幻时间日程

- □ 制订波浪式规划。你只需规划出前几周的详细工作,还有主要的里程碑。过了头几周,要是你还不能提供详细的日程,人们就会觉得你无法预测未来。可以再制订一个新的日程,其中带有已经完成的任务、下一个波浪的工作、每个月更新后的里程碑。这样就可以告诉别人做完了哪些工作,而且不必束缚于一个无法实现的庞大计划之中。
- □ 使用低技术含量的日程安排技术,比如黄色即时贴式日程安排或是墙上的卡片。还可以 邀请决策层复查项目日程。
- □ 提供带有信心水平的估算,而不是用甘特图。
- □ 使用有时间盒限制的迭代,而且每次只规划一个迭代能做的工作。测量团队的开发速度。 过了三个迭代之后,项目经理也许可以知道足够多有关速度的数据,这就可以预测项目 剩余的日程了。

人们之所以坚信日程安排工具的正确性,是因为它假定估计出来的日程可以很准确。而问题在于很少有日程是准确的。很多可以很精确,比如"我们会在周三下午3:32发布产品",但是却不准确。因为日程只是估算而已,让它看起来很漂亮,并不能改变这个现实,而且日程总会有误差。

发生这个游戏,日程安排工具并没有错,关键是使用工具的人对其过于相信。作为项目经理,你必须选定最有效的技巧安排日程,而且要跟别人说明这个日程安排。如果确实有作用的话,用日程安排工具也是可以的。但是不要只因为它能做出漂亮的图表,就对其坚信不疑。

3.11 我们必须拥有这个功能,否则就完蛋了

老板给项目经理曼尼打电话说道:"曼尼,我们得聊聊你的项目了。"曼尼答道:"当然,有什么问题?""嗯,如果这次不加入这个功能,我们就完蛋了。大客户们不会买这个版本的。"曼尼叹了口气,说道:"我跟大家说一下吧,回头告诉你结果。"

曼尼转头跟项目团队解释了这个情况,大家同意在当前版本中加入这个额外的新功能。不过 大家已经对按日程及时交付不抱希望了,顺便说一句,项目日程从来都没有改过。

在这个例子中,每个人都希望项目成功:老板、项目经理、项目团队。可他们没有权衡这样做给项目带来的影响,从而使得项目完全失去了按日程交付的希望。整个项目(和团队)就这么完蛋了。^①参见图3-11。



图3-11 我们必须拥有这个功能,否则就完蛋了

如果管理层和项目团队都是出于好意,项目经理可以尝试下面这些方法。

- □ 要求改变功能集合。你可以问这样的问题: "有哪些功能是在这个版本里面不需要的?我 看能不能想办法把你说的这个功能加进去。"
- □ 要求更多的时间。"如果你愿意推迟交付时间,我们可以加入这个功能。"
- □ 要求更多资金。"我知道你为什么需要这个功能,可与其他功能相比,要多花两周才能做

① 请参见: http://www.stickyminds.com/s.asp?F=S11829_COL_2。

完它。我们不能随便拿出来一个功能,再把这个功能放进去,还得再考虑整个版本。你想让我现在动手做吗?团队做估算、重新规划版本,要占用一些时间。咱们还是搞清楚这到底是不是你想要的吧。"

要想阻止"我们必须拥有这个功能",有个办法。如果是按照功能逐个实现,而且按期提供可供发布的产品,项目经理就可以和管理层重新安排下个版本要实现功能的优先级。使用不超过4周的迭代,是阻止该游戏的最佳方法。用4周的时间盒,人们最久只需等待8周就可以得到新功能[©]。

要是每季度发布一次,比如使用"发布列车"(release train)方式,想得到新功能,最长的等待时间是6个月。这对很多管理层人士来说都太长了。而且,如果每隔6~9个月才发布一次,那别人就要等上一年甚至更久了。

如果你曾在组织内部遇见过"我们必须拥有这个功能",使用带有时间盒限制的迭代吧,这样你就可以管控这些要求了。或者考虑使用"发布列车"。参见原书16.6.1节,以了解如何使用待办事项列表来管理对更多功能的要求,以避免这个游戏。

我们假设你已经有了一个还算靠谱的项目日程。可是总会有出乎意料之外的事情发生,不是每个人都能完成之前承诺的工作。在前一节中,你已经见到了出资人和管理层会玩的游戏。而我见过有些团队成员会玩这个日程游戏。再次强调,项目经理的职责是要让团队成员认识现实。

3.12 我们不能说"不"

作为项目经理,管理层想让你再向当前版本中塞进来一个功能。他们在玩"我们必须拥有这个功能"的游戏。作为一个负责任的项目经理,你告诉了团队成员新的功能要求。你已经跟管理 层说团队会评估请求,而且会告诉他们新的发布日期,或是加入新功能带来的成本。

可是当你开始跟团队成员讨论这个功能及其带来的影响时,大家都不愿意说"不"、"还不行",或是"这样做会带来新的成本"。大家两眼一闭,就把新的工作接下来了,而不去讨论时间和资金上的成本,还有对当前工作的影响(见图3-12)。团队这么做,有时是处于内疚,有时是因为没有估计到额外的工作真正需要多少时间。^②

如果项目经理管理的团队不愿意说"不", 你就得帮他们学会表达不同意见。不过只对高层

① 假如新功能在一个为期四周迭代的刚开始阶段提出,那么这些功能只有在下个迭代中才能被安排开发,因为一个 迭代开始之后,其中要开发的功能是不能发生变化的。这样,要得到完成的新功能就需要两个迭代,也就是8周。

⁻⁻⁻译者注

② 请参见: http://www.stickyminds.com/s.asp?F=S11829_COL_2。

(或市场部,或是其他希望加入新功能的人)说"不"还不够。项目经理可以对任何事情表达不 同意见。如果人们试图处理额外的工作,为了帮助他们管理这些额外的工作,可以考虑下列方式。

- □ 询问团队成员,看他们能否针对添加额外的功能制订计划。可以使用黄色即时贴式日程 安排和相对大小方式,看看如何能做得到。如果可以提供一个大家都满意的计划,项目 经理就要尽量实现这个计划了。
- □ 项目团队成员有时会说:"我们会加入这个功能,并且加班完成。"如果他们想加班,要 建议他们用时间盒限制加班时间,并衡量加班的结果。项目经理可以说:"好吧,我们把 工作按为时一周的迭代分开。可以加班一周,再看看我们的工作效率怎么样,疲劳程度 又如何。过了第一周之后,我们再以正常的方式工作,再衡量工作效率。然后我们可以 对比这两个工作效率,看看有没有引发诸如额外的变更或是缺陷的新问题。如果觉得加 班没有效果,我们还可以按正常时间工作,到时候再看看是不是还能有其他方式做得更 好。"
- □ 项目经理有时可以加入额外的人力来做更多工作。(不一定总好使。) 如果组织中有人具备 领域专业知识,可以融入到团队之中,而且团队也希望有这些人加入,项目经理就可以把 这些人加入到团队中。不要加入对产品或团队不了解的人——想想布鲁克斯(Brooks)[®] 法则,参见原书7.5节。

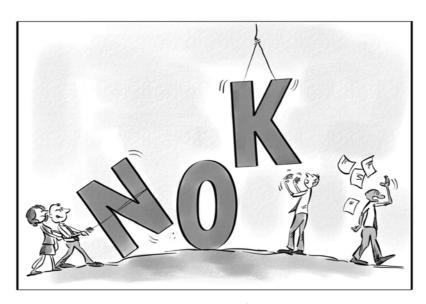


图3-12 我们不能说"不"

① 弗雷德里克·布鲁克斯 (Frederick P.B Rooks, Jr) 在《人月神话》中提出一个重要的法则: 向进度落后的项目中 增加人手,只会使进度更加落后。——译者注

如果上面这些方法都不起作用,项目经理就要帮助团队说"不"了。你可以用数据来抵消团队的内疚或是反驳完成公司要求的渴望。速度图表和迭代内容图表在这里特别有用。如果项目经理不能帮助团队学会说"不",那大家就踏上死亡征途了。没有人希望这样。

3.13 日程小鸡

在逐个报告进度的项目进度会议上最容易发生"日程小鸡(Schedule Chicken)"的情况。[®] 项目经理会问大家的进展情况如何。每个人都说自己在按日程安排进行,而实际情况却是没有一个人做到。每个人都在等别人眨眼睛并承认自己没能跟上进度。而且除非已经太晚了,没有人会承认自己错过了项目进度安排。参见图3-13。



图3-13 日程小鸡(树干上的文字是"日程")

讲求实效的项目经理会有如下选择。

- □ 避免逐个进行的进度报告会议。可以采取原书10.5节中的方法来判断项目真正的进展状态。
- □ 将任务分解成更小的部分,这样每个人每天都能交付一小块任务,最多两块任务。
- □ 按功能逐个实现。如果能有更多的人把精力集中在一小块可交付的功能上,就可以尽快完成这个功能,而且人们也更容易看到自己所取得的进展。在某些情况下,人们玩"日程小鸡"的游戏,是因为开始得晚(见原书8.10节中的提示)。定期看到产品的各个部分集成到一起,可以让大家都看到项目的进度,从而能够帮助他们知道何时没有取得进展。
- □ 考虑使用迭代吧,特别是使用敏捷生命周期。用了短期迭代,就不必再坐下来开每周的

① 我第一次讨论"日程小鸡",是跟戴夫·史密斯(Dave Smith)和杰瑞·温伯格(Jerry Weinberg)。

小组项目进度会议了(而且这个会议绝不应该开)。有了每日站立会议,人们就不能掩饰 自己的真实进度了。

3.14 90%完成状态

许许多多的知识工作者,特别是技术人员,都未学过如何估算。就算是尝试过估算的人,他 们也过于乐观了,总是会过低估计一项任务需要的工作量。要么得不到任何对估算的反馈,所以 他们不会知道自己的估算是不准确的;要么他们预测不到一个任务中会牵涉多少子任务,比如准 备测试环境或是签入代码这样的任务。在任何情况下,当有团队成员以为自己完成了90%的任务, 而实际上还有90%工作尚未完成的时候,"90%完成状态"就发生了(见图3-14)。



图3-14 90%完成状态

我在职业生涯的早期,曾是"90%完成状态"的受害者。当时我在写一个数据库对话工具, 要把一种数据库格式转换成另一种。我认为其中的数据是干净的,可事实恰恰相反。我以为我很 清楚每个字段的格式,其实我不知道。我以为我对需求了解得很清楚,可随着工作的进行,每次 处理数据库中的一条记录,我就会遇到更多的特殊情况,每一种都对需求有所变更。

最后我还是振作起来了。我开发了一系列测试用例,改变代码后我就会运行这些测试用例, 慢慢地,整个开发工作开始取得进展。(要想知道更为现代的方式,可以阅读有关行为驱动开发 的资料。[®])上司问为什么进度这么慢,我向他展示了当时的进度,并解释了我们在一开始并不了解的种种特殊情况。

作为项目经理, 你应该指导大家, 以消除"90%完成状态"。

- □ 帮助大家定义出自己的"小石子"。你可以跟对方坐在一起,然后提问:"要完成这项任务,你需要多久?这周要处理的细分任务都有哪些?
- □ 要让大家把自己的工作进度展示给你。这可能会揭示出他们代码中潜藏的问题(正如我的例子)、风险的列表、测试用例的列表,他们也可以在添加了一些代码之后,告诉你这些代码的临时意图。
- □ 教给大家如何跟踪自己的估算,让他们了解自己一开始的估算准确度是怎么样的。

有些情况下,人们进入"90%完成状态"是因为他们的实现工作跨越了整个系统架构。如果项目经理让他们按功能逐个实现,同时以短期迭代方式开发,他们就会开始以更小的粒度对工作进行估算和实现。他们的估算会越来越准确,而且也能提高工作的完成度。

3.15 我们马上会变得更快

假定项目经理正在管理一个敏捷项目,或是按阶段交付的项目,或是其他生命周期类型的项目,总之这个项目可以用增量式的方式来构建系统。项目经理一直在测量团队的开发速度(或是实现的功能),可是进展速度没有达到预期效果。由于某些原因,团队成员对于如期交付很乐观(见图3-15)。



图3-15 我们马上会变得更快

① 请查看http://behaviour-driven.org/。

讲求实效的项目经理不愿意发出悲观论调或是冷嘲热讽,他希望让大家认识到现实,帮助团队成员看清真正的进度。说到底,也许他们还是可以赶上进度的。可以采取下面这些措施来管理信马由缰的乐观情绪。

- □ 与项目团队成员讨论进展速度。收集数据: 是什么样的所见所闻, 让他们觉得接下来可以比之前工作效率更高?
- □ 测量估算质量因子。有些因素会让人们觉得自己一直在按进度计划开发,甚至超越了进度计划。要特别注意这些因素。
- □ 测量团队所做的每一件事情。确定每个人都全力投入到项目中,而且都为了按规定日期 交付而开发必要的任务。如果有人在为其他项目工作,或是承担某个后续版本开发的任 务,要马上中止这种状况。
- □ 如果项目中涉及硬件组件,要测量其挣值,看看它是否能按进度计划进行。如果做不到 (现有挣值低于应有挣值),重新规划整个项目吧。
- □ 在测量开发速度时,在团队完成第三个迭代之前,对于项目出现的任何整体速度数值都不要感到惊讶。因为只有到那个时候,团队的速度才有可能稳定下来。

3.16 令人恍惚的日程

下星期二,来自总部的大人物就要来视察了。或者你可能要在十个星期后做上市演示。不管怎么样,你要面对一个不可改变的日期、一系列充满野心或是不可能实现的功能集合,而且这些功能必须要在那个日期之前完成。无论是否已经测量过团队的开发速度,你知道团队是无法在截止日期来临之时完成所有的功能了。看起来,团队面对着日程已经处于恍惚的状态了。有时,这就是团队对于"幸福日期"的回应。参见图3-16。

首先,要创建项目的仪表板(dashboard),再测量进度,从测量开发速度开始。然后考虑采取如下行动。

- □ 如果还没有用迭代式开发,赶紧把剩下的项目分解成迭代吧,迭代时间越短越好。如果在不可改变的截止日期前有十个星期的时间,项目最少要分解成5个迭代,最好是10个。每周一个迭代,这可以帮你不断调整工作的优先级。迭代的目标是要完成某个功能或一组功能,这包括功能的开发、文档、测试,以及其他产品对于"完成"的要求。如果上市演示需要在线帮助,除非一个功能具备了在线帮助,否则它就不算完成。
- □ 在一个迭代中要集中精力,每日站立会议对此会有所帮助。项目团队的目标必须放在要完成的工作上。当完成足够多的细分任务之后,就可以得到一个完整的功能或是产品了。 不要让团队中的人们受其他项目、未来工作或是技术债务的干扰,除非技术债务使得当

前迭代的工作无法完成。

□ 如果还没有准备好,就按功能逐个实现吧。这样可能会产生不完整的系统架构,不过不 必担心。如果一个功能需要架构提供某些方面的支持,团队会实现的。如果功能上不需 要,那就没有人会用它。



图3-16 令人恍惚的日程

通过使用"游击队式敏捷(guerilla-agile)"^①,可以消除很多上述的日程安排游戏。如果能够以不超过四周的时间盒来组织项目,按功能逐个实现,随进度集成,并测量进展速度,项目经理就可以中止这些游戏。如果总是能以这样的方式管理,那就可以完全避免绝大多数的日程安排游戏。

₩ 铭记在心

- □ 日程安排游戏总是会发生的。项目经理的工作就是要识别它们,然后管理项目,让项目 仍然可以取得成功。
- □ 绝大多数情况下,人们玩这些游戏都不是出于恶意。
- □ 即使没有恶意,日程安排游戏还是会拖项目后腿,使其停滞不前。

① guerilla agile是作者Johanna Rothman在Agile 2007大会上的演讲题目,指的是非正式的敏捷实施方式,演讲提纲下载地址: www.agile2007.org/downloads/handouts/Rothman_383.pdf。——译者注

掌控项目

4

丁 目经理已经编写了项目章程,也有了相应的规划。团队知道"完成"的含义,也有了 项目日程。大家的工作正常进行着。项目现在正式进入了中期,作为项目经理,你的 职责就是掌控项目,使其到达成功的彼岸。

如果还没有制订出项目的仪表板,现在赶紧动手。定性和定量的度量方式都需要,这样才能 反映项目的真实状态。一旦真实状态清楚了,项目经理就可以决定如何把握项目的方向了。

掌控项目意味着要寻找风险和管理风险。通过组织项目来发现它的节奏,这是一种发现风险的方式。任何扰乱节奏的事情都是已经出现的风险,任何可能扰乱节奏的事情就是潜在的风险。通过积极地管理风险,项目经理可以保证项目不会偏离正确的轨道。

4.1 掌控项目的节奏

节奏,每个项目生而有之。有些项目节奏混乱,进展缓慢。有些项目就像坐上了火箭,倏忽间,团队就完成了更多工作。所有的项目都有节奏,不过它会随时间而变。观察节奏是项目经理的职责,你还要能看出来,那些实践是否能够帮助项目建立并维持合理的节奏,保证项目取得成功。

项目的生命周期越接近顺序式生命周期,项目的节奏就越有可能随阶段发生变化。在早期阶段,项目看起来就像遭受了很多痛苦,项目经理也不知道需求何时冻结、设计何时稳定。有些项目会"神奇地"找到出口,在进入实施阶段后,节奏逐渐显现出来,而且日趋稳定,因为该做的决策已经做了,而且得以实行。假定没有人手方面的问题,项目就可以顺利完成。如果项目总是混乱不堪,项目经理就会在整个生命周期中都找不到方向,也难以发现令人舒服的节奏。

在敏捷生命周期中,迭代规划阶段也许会察觉到混乱的状况。不过迭代规划只会持续几个小时,团队在迭代开始后可以找到像鼓点一样的节奏。(如果你曾见过敏捷生命周期找不到节奏的

状况,很可能是因为团队没有遵循敏捷的价值观,或是错误使用了敏捷实践。)

下面这些打断项目正常节奏的问题, 是我亲眼所见的。

- □ 不知道先要完成哪些需求。
- □ 允许项目的需求收集阶段持续过长时间。
- □ 允许GUI随时发生变化,而GUI相关人员在项目中不知该如何是好。
- □ 没有架构的整体描述,不知道各个部分如何构成。
- □ 无法及时向项目中加入必要的人员, 使得他们很难取得成功。

如果见到项目在努力寻找稳定的节奏,可以考虑使用下面的管理实践,也可以参见原书16.5 节和第9章,从中得到更多建议。

4.2 举行中途回顾

寻找估算全部落空的原因

——史坦利,被人围攻的项目经理

我被难住了。不管针对什么,我们的估算总是落空,这个难题一直在困扰着我们。我们无法达成时间盒的要求。无论估算多么小,我们都无法达成。我以为这是个系统性的问题,所以决定在项目结束之前召开一次回顾会议,回过头看看过去几周的工作。

在这个中途回顾中,好几个人解释说: 总是有市场人员和销售人员过来让他们"再新增一个功能,就一个"。我根本不用开口,因为技术带头人斯特拉说道: "要是我们在接受这些变更时不把它们加入到待办事项列表中,我们就永远无法完成之前承诺的工作。我们必须让史坦利看到这些新的需求。"

在回顾会议中做之后的行动规划时,团队想出一些方法让我接收到新的需求,包括让总是 在叫着"再新增一个功能"的人到我的办公室。而我也意识到,我必须更频繁地发布产品待办 事项,而不能光让人们去主动找到它。

如果没有这个中途回顾会议,我可能永远不知道为什么我们的估算总是会出问题。噢,也 许到了项目后期会发现,可那时整个项目已经变成一场灾难了。按照目前的状况,我可以更加 积极地管理项目后面的进展了。

要想了解项目中发生了什么,同时为未来的项目做规划,回顾活动是一种非常出色的方式。在项目结束时举行回顾挺好的,而且项目经理也可以把经验教训用到下个项目中。不过你也可以在项目的任何时候举办回顾,这样就可以知道明天是不是能换一种更好的工作方式。

敏捷式生命周期项目需要在每个迭代结束时举行回顾。顺序式、迭代式和增量式生命周期项目也要做回顾,不过是在达成主要里程碑之后。如果两个里程碑之间的时长超过一个月,项目经理可以举行一次中途问顾,以了解工作进展情况。

我推荐德比和拉尔森的书,读者从中可以了解许多种举行中途回顾的操作方法。

中途回顾物有所值

项目经理皮特负责一个长达18个月的阶段-关卡式项目。原计划打算用6周收集一些需求,以开始架构原型化的工作。不过6周后,他们发现收集的需求根本不足以用来做原型化。皮特决定不再收集需求了(需求分析师要求说:"4周,再多给我们4周。"),他要举办一次中途回顾。

在回顾会议中,团队知道了为什么总是收集不到足够的需求。而且,他们也了解到:如果继续之前的工作方式,再给4周他们也收集不到"足够"的需求,至少还需要12周。项目开始6周后的这次回顾,避免了可能出现的3个月的延迟,而且也让项目团队相信:选择并坚持不同的生命周期模型势在必行。

4.3 为需求排序

我们有"超高"优先级的需求!

——帕翠莎, 出类拔萃的业务分析师

公司来了一个新的产品经理汤米,他对工作充满热情。他讨厌对客户说"不",承诺客户 所有的功能都会在下个版本中提供。我总是搞不懂他的想法,他对这些需求总是只说一半,然 后就再也没时间跟我细谈了。

压垮骆驼的最后一根稻草上周出现了。他决定我们的需求优先级不再划分为高、中、低三个等级,而是应该包括超高、至关重要、高、中、低这五个等级。

我跟他解释说我已经以最快的速度工作了。只要这些需求能够以数字的方式排序,我很愿意以任何顺序处理任何一个需求。"告诉我哪个需求是第1位的,我会先从它入手,然后赶紧交到开发人员手中,之后处理第2位的需求。"

他兴奋异常: "好主意! 你看, 所有的这些都是第1位的!"

"汤米,也许我没说清楚。你只能选择一个第1位的需求。只有一个。"

"可这样一来,我会让一些客户失望的。"

"我很遗憾,你向太多的客户做出了太多的承诺。但是我们只能有一个第1位的需求,一个 第2位的需求。我相信只要你仔细想想,一定能找出来。" 果然,他做到了。他把所有的需求都排了顺序。接下来的一周,他想重新排序,不过我们已经开始处理第一部分需求了,所以他就没有打扰我们。这也是件好事,否则我就要干掉他了。(这句掐了,别播!)

幸运的话,项目经理可以把所有的需求都收集到一个地方,而且也知道先实现哪个,后实现哪个。不过据我所知,大部分项目经理面对的需求都散落在需求文档和缺陷跟踪系统之中,而且 把它们按照高、中、低优先级进行排列。

接下来我要说的可能会让人觉得很悲观,可这是我在多个项目中亲眼所见的。项目中有太多的高优先级需求,以至于中优先级的需求根本无暇顾及,更甭提优先级低的需求了。这一大堆的高优先级需求也根本无法区分先做哪个。

到了下一个项目,提供需求的人知道你难于完成所有的高优先级需求,所以他们就想出来一个比高更高的优先级名称:至关重要。于是,很多需求的优先级就变成至关重要了。你可能还是要面对一些优先级为高和中的需求。可是现在,你连完成所有"至关重要"的需求的时间都没有了。怎么办?

给需求排序吧。从1开始,为每个需求分配唯一的数字。这样人们就能了解团队会什么时候 处理哪些需求了。即使不按照功能逐个实现,你也可以按照排序的顺序来实现功能。

项目经理可以通过下列方法对需求排序。

两相对比。你可以把两个需求放在一起互相比较,然后问"这个需求的优先级比那个是高还是低?"判断出哪个高之后,它就成为第1位的了,相对较低的需求就是第2位的。

两相对比不太好操作,很可能会遇到判定项目驱动因素时同样的问题。项目经理要以温柔而坚决的方式进行,说明只能有一个第1位优先级的需求,也只能有一个第37位的需求。

按评判标准排定优先级。如果"两相对比"的方法没有进展,可能大家需要把需求的价值和 优先级都明确表达出来,而这些一般是隐藏在他们平时的决策背后的。明确表述需求的价值,大 家就可以根据这些值进行权衡,做出判断。

要想产生客观的条件,可以采用下列步骤:通过"头脑风暴"产生一系列可能的条件、将相似的条件分为一组、为每组分配一个分类名称。类别的名称就是大家用来评估的条件。我过去用过下面这些条件:

- □ 功能对架构的影响。
- 估算的实现时间。

- □ 该功能对于特别重要的客户的重要性。
- □ 特定人员实现或测试该功能的可行性。

为每个条件分配相对权重。负责主持头脑风暴的人可以问下面的问题:这个列表中,哪一个优先级最高?列表中是不是还有跟这一条优先级同样高的条目?这个列表中,哪一个优先级最低?列表中是不是还有跟这一条优先级同样低的条目?

知道了相对的顺序之后,就可以分配权重值了。优先级最低的条目权重值为1。每个优先级更高的条目,权重值为前一个的两倍。如果每种条件的优先级都是独一无二的,就会得到1、2、4、8这样的权重值序列。(你也可以定义高、中、低这样的优先级,而且每个优先级都可以有多种条件。所有低优先级的条件权重定义为1,所有中优先级的条件权重定义为2,所有高优先级的条件权重定义为4。)然后对应每种条件,为每个需求赋一个0~10的值,10说明是最重要的。请参见图4-1。

评判标准	权重	方法1		方法2	
		得分	权重得分	得分	权重得分
功能对架构的影响	1	10	10	3	3
估算实现时间	4	3	12	10	40
该功能对于特别重要的 客户的重要性	8	1	8	10	80
特定人员实现或测试 该功能的可行性	2	10	20	7	14
			66		137

图4-1 评判标准评估

制订并维护不断变化的需求日志:产品待办事项列表。如果有满坑满谷的需求,要想对它们都进行排序是非常困难的,我就遇到过这样的问题。要是有可能,只要一听到需求就应该对它们排序。可以将需求记录在电子表格或日志之中。如果没有使用敏捷项目生命周期,可以维护一个季度日志,记录这个季度要做什么,以及你认为接下来的三个或四个季度内要做什么。参见原书16.6.1节,特别是那张季度待办事项列表的图。

4.4 用时间盒限定需求相关的工作

需求来了一茬又一茬,一茬又一茬……

——朗达, CIO

我的工作是:确保我们的项目可以完成,从而体现业务的价值。我们曾遇到一个拖了很长时间的重要项目,它原本预计6个月完成。项目开始2个月后,项目经理山姆跑过来跟我说: "我真快要把头发都揪掉了。需求总是来个没完,市场部的家伙们不停地添加新需求,而且也不说清楚现有的需求,我甚至都不知道该怎么启动项目。"

我知道怎么处理这个问题。我给市场部副总打了电话,跟他说他的人还有一周的时间可以用来定义需求。只要是定义完整的需求,我们就会做。如果只定义了一部分,我们就不管。就这么简单。

他试图搪塞我,告诉我他的人都不在办公室。于是我提议,如果一周不行的话,那干脆今 天就停止需求收集活动。"别,我们会搞定的。"

我们没有开发他们要求的全部功能,但是确实做完了他们需要的功能。

如果项目中必须有明确的需求阶段,就用时间盒限制它。否则,需求分析和定义过程就有可能占用整个项目的时间。我曾参加过这样一个组织的工作,他们试图将项目的持续时间从18个月降低到6个月,但是需求阶段就用了4个月。在剩下的2个月里,他们没法做任何有用的事情。

在了解了一些情况之后,我知道了原因所在。市场人员过于忙碌,以至于没有时间与系统分析人员讨论,所以他们会推迟需求讨论会议。系统分析人员解释说他们还没有完成需求,因此希望从项目经理那里得到更多时间。这样一来,市场人员也就不急于解释清楚他们对需求的真实要求了。

项目经理决定把最初的需求获取工作限定为两周。市场人员因此满腹牢骚,在两周结束时,他们发出了好几份备忘录,说明为什么这个版本对于客户如此重要。不过开发人员们已经拿到了最重要的需求,可以开始开发产品了。三周之后,他们有了符合最初需求的产品原型,市场人员也已开始与系统分析人员合作定义后续的需求。

接下来,他们可以采用四周的迭代了。这样市场人员就可以用足够长的时间来定义任何给定的需求。不过项目团队知道,这些需求是要在下个迭代中实现的。

即使不适用迭代式开发,也要用时间盒限制初始的需求收集和分析活动。我是用下面这些方式完成的。

用时间盒限制初始的需求相关活动,并持续收集更多的需求。项目经理可以选择一个合理的短时间段(不超过项目总体持续时间的10%),要求收集到最重要的需求。风险在于,你可能得到了对客户来说是最重要的需求,但这些需求不一定能够决定最终的系统架构。

用时间盒限制所有的需求定义活动。这么做对于项目经理来说有点儿危险,可能会招惹来"你是一个没有团队精神的人"之类的废话。不过你可以这么解释:"我已经尽我所能,交付你们需要的产品。如果我允许需求收集工作还像上个项目那样耗费那么长时间,项目就不能及时交付了。如果你们觉得发布日期不重要,我们可以不这么干。可上一次,你们对于发布日期可是相当重视的啊。"

我在使用时间盒处理所有需求相关的工作时,项目团队会与需求相关人员一同收集、整理和细化需求。到时间盒结束时,团队实现了已经完成的需求,而且只实现已完成的需求。这种方式最适用于短期项目,而且是必须使用顺序式生命周期的项目。

提示: 用一分为二的方式减少迭代持续时间

如果迭代效果不好,不妨试试把它一分为二。如果迭代持续6周,就改用3周。如果持续4周,就改用2周。如果是2周的迭代,就改用1周。更短的迭代有助于项目经理更快收集到反馈,洞悉团队在时间盒里的实际工作状况。

一旦知道人们在迭代中的具体工作情况,项目经理就能知道问题的出处了。是因为估算不准确,还是因为同时处理多个任务?也许是其他原因。项目经理就可以着手移除障碍。长期的 迭代会掩盖问题。

4.5 将迭代限制在 4 周或是更少的时间内

我们完不成迭代的工作

——托弗,项目经理

这6个月来,我们一直试着用4周的时间盒推进工作,可我们怎么都完不成在迭代开始时估算可以完成的任务。

第一个迭代结束时,我们还剩几天的工作没完成,所以就把迭代扩展到了5周。下个迭代中的工作也相应增多,我们不得不把迭代增加到6周。后来就变成了8周一个迭代,而我们仍然无法完成为迭代规划的工作。

最后,我们决定采用一分为二的方式,也就是使用2周的迭代,看看能不能提高估算的准确性。结果发现,我们的估算没问题。可除了开发工作之外,我们还要做支持的工作,所以在一个

迭代中, 我们不能像预想的那样投入足够时间进行开发。我们的任务估算没有问题, 但是对于 工作时间的估算却落空了。

要是不采用更短的迭代,我们恐怕永远都不会发现是这么一回事。

也许你已经开始使用迭代了,可迭代持续时间是8周,甚至更多。现在你就该遇到其他问题 了。在一个迭代周期中,测试人员无法完成他们的工作,要么就是开发人员总是过多估算自己可 以完成的工作。或者,项目经理使用了螺旋式生命周期,但却没有按照完成一个完整功能的方式 进行规划。

迭代时间越长,项目的节奏就越难保持。在顺序式生命周期中,项目经理可能已经见过类似问题,因为整个项目就像是一个迭代。如果很难保持项目的节奏,那就用更短的迭代吧,直到能够保持节奏为止。

这看起来好像违背直觉。"如果我使用6周的迭代都有问题,那4周的迭代又怎能帮到我呢?"答案在于:更频繁的反馈。迭代周期越短,要是项目可以因此达成特定的里程碑(理想状况下就是可发布的软件),那就很容易知道应该如何启动和结束一个迭代。"哈德逊湾式启动"(见4.2.4节)能够收到成效就是这个原因。如果不能保持项目的节奏,人们就很难进行准确的估算,他们会试着一次完成太多的工作,或者同时从事多个项目的工作,或者就不知道先从哪些工作入手,也许还会有其他特定于项目的原因。短时间盒会让问题暴露得更加明显,项目经理就可以着手解决。

4.6 使用波浪式的规划和日程安排

每次规划少量的工作,这让我更具灵活性

——唐纳德,项目经理

我们受困于顺序式生命周期。我们有严格的阶段关卡要求,而且管理层要审核每个关卡的 完成状况。不过在各阶段关卡之间,我们使用时间盒,并对原型进行迭代,这样就能知道估算 的质量如何。

现在我们处于设计阶段,而且大家认识到:我们无法让特定组件的性能符合要求。这是不可能的:它违反了物理规则。现在就得重新规划我们的工作了,发布的产品也会与规划的有些许不同。

我不担心日程安排。因为每次只规划4周的详细工作。我必须让大家知道接下来几周我对项目的规划,不过重新规划并不费事。我担心的是产品,不过那是另外一个问题了。

在项目中的某个点,某些风险的发生会改变整个项目的形势。软件项目会以不可预见的方式

显现出本来面目。各个任务的完成时间也会发生变化。如果项目经理准备好用迭代方式反复规划,就可以提升项目日程安排的准确性。如果项目经理没有足够的产品相关知识,或是没有足够的历史数据来安排准确的项目日程,或是由于项目过大很难准确规划,那使用迭代式的规划和日程安排,就有助于项目应对技术风险和日程安排风险。

重新规划顺序式生命周期项目的时间安排。要先定义出顺序式生命周期的阶段或主要里程碑。确保团队明白选择的里程碑特定意义,要不就赶紧定义出里程碑的含义。比如,如果有个叫"需求冻结"的里程碑,就要明确说明它在项目中的含义何在。里程碑定义完成后,要为每个里程碑设定验收条件。这样一来,项目经理就可以知道里程碑何时算完成了。不要试图规划所有的工作。只规划将来3~4周的详细工作,这样每个人都知道接下来要做什么,后续几周要完成哪些任务。(如果你知道某些功能是要在项目的特定阶段完成的,比如演示、参加商业展览或是beta版发布,就得把这些特定阶段也定为里程碑。)

项目经理要做的就是监控项目进度,牢记里程碑验收条件。里程碑验收条件是为了达成特定 里程碑必须要完成的任务。举例来说,我参加过一个团队,在顺序式生命周期中,他们将"需求 冻结"作为里程碑,将"foo功能的需求完成并经过审核"作为验收条件。在顺序式生命周期中, 人们很难达成"冻结"或是"完成"这样的条件,因为很难说早期的阶段是不是真地完成了,除 非已经拿到了可以工作的代码,而且能够用它来表示需求或设计。

团队一完成手上的任务,项目经理就可以向当前的详细规划中添加新的任务了。假定项目经理制订的是4周的波浪式规划,如果目前处于第3周,那就可以规划第7周的工作了。想通过迭代的方式来规划顺序式生命周期项目的工作比较困难,因为进度难以评估,除非已经接近某个里程碑了。这就是为什么需要里程碑验收条件的原因。

使用波浪式规划的顺序式生命周期有个问题。由于项目分成各个阶段(需求、分析、编码,诸如此类),在项目开始时,人们很容易倾向于规划尽量详细的工作。不要规划所有的细节,只为接下来的几周规划详细工作就可以了。

项目完成一个前面的阶段后,就可以为后面的阶段准备更多细节了。随着项目进展,看看是 否存在技术债务的状况。如果后续阶段所费时间超出预期,这就是存在技术债务的警讯。技术债 务越多,后面阶段所耗费的时间越长。

如果顺利进入编码和测试阶段,项目经理就可以规划最后的测试和项目临近结束的活动了。 比如早期版本发布、beta发布,还可以有其他在最终发布之前的活动。

项目经理已经使用波浪式规划来构建项目日程,那么在顺序式生命周期中,项目经理要创建里程碑,在完成一个阶段后要重新规划这些里程碑。要将重新规划活动放入最开始的项目日

程安排中。利用目前对项目的了解(特别是在项目中期回顾得到的信息),更新项目的规划和日程安排。

这样一来,团队就能知道项目是受控的,团队和项目经理可以不断评估并管理与日程安排相关的风险。

提示: 将重新规划活动放到项目日程安排中

要是没有使用敏捷生命周期组织项目,还是将重新规划活动明确标识出来吧。还要经常重新规划,这样项目日程就不会在众人领会它之前脱离现实了。

为迭代式、增量式和敏捷生命周期进行重新规划。由于这几种生命周期模型都包含迭代或增量式活动(或者两者皆备),相比顺序式生命周期,要进行重新规划就简单多了。

对于每次迭代不会将完成的代码交付到代码库中的迭代式生命周期,可以使用与顺序式生命周期相同的方式进行重新规划。可能会有不同的里程碑,比如"探索原型1并发布结果",不过背后的理念都是一样的。

在项目启动时,增量式生命周期进行重新规划类似于顺序式生命周期,因为一开始时也是倾向于分阶段的。一旦增量开始后,项目经理会发现,要制定波浪式的日程安排很容易,因为不必等到一个阶段结束。你所期待的是将完成的(即经过开发和测试的)代码签入到代码库中。

敏捷生命周期会自然而然地使用波浪式规划,因为每个迭代都是一个时间盒。项目经理规划 的工作只需启动迭代即可,要在迭代中监控进度,而且确保迭代完成时提交完成的工作。



是不是迭代式日程安排不可避免?

不是每个人、每个项目都适用迭代式日程安排。它最适合用在下列这些情形下:

- □ 当你知道要做什么,却不知道应该怎么做的时候。
- □ 当你面临时间压力,而且希望利用项目现有进展的时候。

这样以来,研究型项目或是项目的研究阶段就不在适应范围之中了。要在研究型项目中使用迭代式日程安排,就得制订出每个时间盒结束时需要提出的问题。然后就可以用迭代式日程安排来重新规划下一个时间盒的工作了。项目产出的不是产品,而是对于一些问题的答案——或是更多问题,这是一种不同的交付产品。

4.7 创建跨职能团队

各自为战使我们一叶障目,不见泰山

——布莱恩, 开发经理

我是一名开发经理,负责一个大型的事务处理产品。我们有一个GUI的前端,一堆中间件,后端是为数众多的数据库。

所有的开发人员在一起工作,设计和开发产品。我负责管理项目。然后我们就会把开发的 程序交给测试人员。测试经理南希会负责发布产品。

完成开发后,我们会开始下一个项目的工作。同时我们会根据测试人员对之前项目返回的测试结果进行修复。这对当前项目的开发工作形成很多干扰。而且,客户开始使用产品后也会发出抱怨,他们会发现开发人员和测试人员都没有找到的问题。

为了以后的项目考虑,南希和我坐下来谈了谈。我们该做些什么,才能保证完成现在的项目,并且不受所有这些干扰的影响呢?我们决定将测试人员加入到开发团队中。南希承担项目经理的职责。对于跟踪细节,她比我在行。

结果真令人惊喜!开发人员喜欢跟测试人员一起工作,而测试人员也十分愿意跟开发人员一起工作。在设计时,测试人员能够发现开发人员想不到的问题。而有些开发人员也喜欢炫耀他们精巧的想法,帮助测试人员测试。后来,等到产品上线后,几乎没有任何来自客户的抱怨,我们也可以把全部的时间都用来应对新产品了。

我们不再把工作在各人之间传来传去了,大家不再各自为战,我们现在以跨职能团队的形式工作。

在第7章中,我建议项目经理应该在自己的团队中配备几种不同角色的人员。团队的构成应该是跨职能的。跨职能团队有以下好处。

- □ 跨职能团队的工作效率更高。单一职能团队可以更快地完成各人负责的部分,但是没有人可以复查或验证他们的工作质量。整个产品也不能早日完成。真正能让项目团队得到认可的,不是因为完成了需求,而是因为实现了全部功能。
- □ 跨职能团队具有多样性。测试人员试图从可测试性方面考虑问题,技术文案跟开发人员 和测试人员一起确认如何表示项目产品的工作方式,分析人员不断细化需求,并有可能 参与构建验收测试。

4.8 根据项目的风险选择生命周期模型

标准流程对我们来说不管用!

——辛西娅,项目经理

我们属于一个大公司,称我们为研究所,也算是八九不离十。公司有一个项目管理办公室 (PMO),他们会定义所有的项目管理流程。

几年前,他们定义出了"标准的"项目流程。一个标准的项目是很庞大的,通常超过100个人,而且会持续数年之久。当然,没有哪个项目经理能够按照PMO定义的方式来成功管理一个标准的项目。

我们决定通过不同的方式来管理项目。我们很聪明,可以产生PMO需要的文档和度量数据,而且仍然使用我们自己的方式来管理项目,以取得成功。我们的项目规模大小不同,时间长短不同,有些项目的技术风险很高,有些基本上没有任何技术风险。而且,当我们必须要管理成本的时候,我们也能做得到。

现在,即使是PMO也说: "根据你们的风险选择生命周期模型吧。选择哪些实践,也要考虑 到你们的团队和所选生命周期模型的现实情况。"我们一直没有失败,部分原因在于不再去碰那 些无比庞大的项目。我们从小起步,保持小规模,使用迭代、时间盒和增量式开发,因为它们都 很有效。而且我们与团队一起,基于实际情况,选择符合我们要求、能够为我们所用的实践。

对我帮助最大的,就是认识到:我们不一定非得用顺序式生命周期来管理所有的项目。我 有其他选择!一旦选定了生命周期模型,再考虑用哪些实践就很容易了。

在原书第3章中,可以看到各种生命周期模型能够明确展现和解决的风险。选用何种生命周期模型,是项目经理必须先决定的几件事之一,而且对如何组织项目会有长时间的影响。要三思而后行。如果不能确定,不妨先使用敏捷生命周期,因为它能尽早带给项目最大的灵活度。

4.9 保持合理的工作时间

每天晚上都在公司吃晚餐毫无道理,对任何人都是如此

——贾斯汀、开发总监

我们大概有150人投入了下一个版本的开发,当然,这个版本对公司的发展至关重要。整个进度陷入了停滞阶段。我的老板——公司副总说现在到了每个人都得在公司吃晚餐的时候了。

计划原本是这样的:每晚7点,有一名总监会负责安排所有参与该项目人员的晚饭。我们聚在一起,共进晚餐,之后大家会继续工作,努力多干些活儿。

头一周看起来还不错。接下来,有些人就得上午10点或11点才能到公司了。有一个人甚至下午才能到。而且,不少人吃完晚餐后就离开了。我整天在办公室里面转悠,看到有人在整理私人账目,有人在给伴侣或是父母打电话。还有一位,正在帮自己的孩子安排周末聚会时间。

最后,我们停止了这样的做法,恢复到每个人每周工作40小时。突然间,代码和测试的质量开始提升了!我们也能完成更多的工作。

项目经理在面临很多问题的时候,会很容易产生让团队一起加班工作的想法。可是如果人们加班加得越多,他们能完成的工作反而越少。为什么?请往下看。

人们每天最多只能完成6个小时的技术工作。在较短的时间段内,至多1~2周,有些人也许每 天能多于1~2个小时。可是大多数人都不能进行长时间的加班。

长时间的加班会扰乱一个人每天的节奏,会让他把更多时间花在喝咖啡、吃午餐、给朋友或 是父母打电话、浏览网页、整理私人账目、关注自己的状态等与工作无关的事情之上。要不了多 久,这些人的心不在焉就会在他们的项目工作产出上反映出来。

项目经理如果发现项目工作进展过慢,可以考虑下面提到的这些方法,并用之来维持项目的节奏。

4.10 使用"小石子"

乔无法估算超过一周的工作

——艾德里安, 开发经理

我的团队中有一个技术高手——乔。他的工作非常杰出,去年由他设计的一个架构表现得很出色。乔可以预估小量的工作,但是他不善于估算大任务。不管是大项目还是任务,如果要他来估算,总会出现偏差,低一个数量级,甚至更多。

在上一个项目中, 乔和他的小组负责一大块任务, 老大一块儿。因为任务过大, 乔无法从 全局把握。他只能估算自己能够看到的部分。可是有些任务就像冰山一样, 乍一看, 你所能观 察到的只是任务的冰山一角。只有深入进去, 才能了解任务的全貌。

现在我们会把每个任务都拆成"小石子"(有时会用"探索式开发"的方式)。乔现在的估算就好多了。我也不再给每个人的估算加上一个随意的经验系数了。

如果要耗费很长时间才能完成一个任务,项目就会丧失节奏。团队成员看不到任务结束的时候,他们就会变得懈怠,从而没有节奏感。"小石子"可以帮助每个人保持自己的节奏,并让项目不至偏离正轨。

提示:帮助团队成员避免"学生综合症"

如果人们直到最后一刻——有时甚至已经过了这个时刻——才开始某个任务的工作,这就发生了"学生综合症"。如果人们用周来估算任务,就很容易引发"学生综合症",用天或是"小石子"的方式就不会。

"学生综合症"会导致项目延迟,打乱项目节奏。如果汤姆无法完成自己的工作,而杰瑞又得等汤姆工作完成后才能开始,那杰瑞现在就处于"等待"状态了。

通过指导团队成员使用"小石子"估算任务,或是使用有时间盒限制的迭代,就可以控制"学生综合症"。不管哪种方式,每个人每天要有所交付,可以有效避免"学生综合症"的发生。提交自己负责的工作所产生的一点点压力会让很多人开动起来,不再让别人空等许久,他们不想拆同事的台。如果人们能看到自己的工作成果,他们就会继续不断取得进展。这样一来,每个人都能保证合理的工作时间。

4.11 管理干扰

干扰是我们的障碍

——乔希,工程副总

我一直伴随着这家公司成长。当初我们只有7个程序员,十年之后,现在已经发展到几百名 开发人员、测试人员、文案、发布经理和其他各色人等。我花了一些时间才认识到:对我们来 说,干扰是个很大的问题,特别是我们还是一个小公司的时候。

我很早就知道干扰是个问题,不过却没有认识到它的严重性,直到项目经理泰德给我一个列表,罗列出了他的团队在过去一周遇到的种种干扰。为解决这个问题,我采取了如下措施:管理项目组合,使用有时间盒限制的迭代,将更多注意力放在我们如何向组织和项目中引入新人之上。

虽然我们现在还没有做到足够完美,但是干扰不再对项目产生严重负面影响。

干扰会摧毁一个项目的节奏。一次干扰还好说,要是所有的干扰一起上阵,就像被一群鸭子 围攻,也能致人死地。干扰会导致人们丧失多达40%的时间。有两种类型的干扰:其他项目和其 他人。

4.11.1 应对其他项目干扰

作为项目经理,你应该保护迭代的工作。如果使用顺序式生命周期,整个项目就是一个迭代。如果使用时间盒,迭代就是时间盒的持续时间。在迭代式生命周期中,迭代持续时间可以发生变化,不过与要完成的工作有关。在增量式生命周期中,本质上是没有迭代的。开发某个功能的时间也许可称之为"迭代"。

要推迟来自其他项目对团队的干扰,除非当前的迭代结束。结束后,可以推迟下个迭代的开始,处理干扰。

人们有时不知道他们的干扰会影响你的项目。将一周之内发生的干扰记录下来,让人们看到 这些干扰的影响。要以事实为依据,但是不要去责怪别人——项目经理要起到教育和通告的职责。

4.11.2 应对其他人的干扰

项目经理要建议团队成员及时互相讨论项目相关的问题,项目工作会因此而取得进展。不过当有人向别人提问时,被提问的人就相当于被干扰了。在充满隔间的办公室中,被提问者周围的人也会受到干扰。项目经理该怎么办?

你有下面几个选择。我的首选建议是:鼓励大家结对编程(或是结对制订需求、结对测试),这样人们就可以共同学习和了解系统。如果结对不起作用,就要让人们在隐秘的空间中讨论,从而不至影响其他人。如果很难获得一个"隐秘"的办公室,就弄一个项目"作战室"吧,可以在其中贴出项目的仪表板、架构文档等各种产出物。

要保证有配置完备的电脑,人们可以用其访问代码库或是其他电子文档,共同讨论同一个产出物。

如果从未要求过"家具警察"的帮助,项目经理可能会觉得有点儿难。不过"家具警察"也是人。看看下一节中的文本框,想想能做些什么。我发现用"点心加啤酒³"这样的小恩小惠效果不错(不过不是总能对同样的人起作用)。

4.12 管理缺陷,从项目初就开始

缺陷?我们没有什么烦人的缺陷!

——爱德华,程序经理

我给你讲个故事吧,这还是在我开始非常重视缺陷之前。那时我对软件开发还不甚了解,虽然已经管理过不少项目,不过绝大部分都是销售项目。开发人员完成编码工作就高高兴兴下班了,而测试人员就一直在抱怨他们无法让软件通过构建过程。我去询问开发人员,他们每个人都会说: "在我的机器上没问题啊。"我觉得这帮搞测试的也太孩子气了。

我们的一位资深经理生病了,而且要休长期病假。由于我是唯一有过管理公司经验的人, 因而资深管理的角色就落在了我的头上。项目本来预计再有2个月就可以完成,所以我就找了咨 询师珍妮丝来管理项目。

珍妮丝所做的第一件事情就是记录缺陷列表。她甚至强迫开发人员使用缺陷跟踪系统。人们都来向我抱怨,所以我就去问珍妮丝那个系统是怎么回事。她解释说,开发人员从项目一开始就置缺陷于不顾。即使他们想起来了,也只是会想到刚修复的缺陷,而不是还没有解决的缺陷。珍妮丝说:"如果不着力解决这些缺陷,这个项目永远都完不成。实际上,还有一个开发

① 要注意组织对于日常工作日饮酒的政策规定。很少有公司会在甜食方面做出限制。

人员跟我说: '缺陷?我们没有什么烦人的缺陷!'"然后珍妮丝告诉我,她认为我们已经有数百个没有解决的缺陷。"这些缺陷让我们的测试不能继续,而且客户也因此而不能使用系统。老实说,我觉得缺陷已经让一些开发人员没法继续写代码了。"

珍妮丝和我又讨论了一会儿。她说服我这是一个很大的问题。她是对的,项目当时累积的缺陷使得我们无法达成最初2个月的截止期限。我们又用了4个月才发布系统。不过我们做得不错,客户也很满意。现在我再也不会将缺陷置之脑后了。

如何影响别人

只要你愿意放弃"命令和控制"式管理所带来的幻象,学习影响别人就很容易了。也许需要你在心态上做出调整。下面的提示供你参考。

- □ 记住问题不是你一个人的。项目经理有时会认为自己必须提供所有的建议和解决方案。并非如此。在项目中,发布日期、功能集合、缺陷水平,每个人都对这些有责任。不要认为一个问题就是你自己的。你有责任让团队解决问题,而不是强令他们如何解决。
- □ 思考你能为组织带来的价值。一旦明白了自己的价值,你就可以想想这些价值对于别 人的意义了。这些价值可以帮你寻求别人的帮助,并给予回报。
- □ 发现其他人或团队的驱动力(WIIFM,what's in it for me?)。发现其他人或团队中为 我所用的东西。有人喜欢做有意思的事情,有人希望得到公众或是个人的认可。很多 人希望把工作做得出色,而且知道自己能为整个项目的成功做出哪些贡献。大多数情 况下,参与多地点项目的团队有其特定的激励因素。要把这个因素找出来。
- □ 建议项目经理和负责的人(或团队)共同面对问题。这可以让项目经理能够友善、开放地接受他人的建议和想法。如果你直接告诉别人怎么做,他们会有千千万万个理由告诉你"你错了"。如果让他们自己找答案,他们会更容易去解决问题。
- □ 倾听团队。团队会告诉项目经理他们要怎么样才能达到最高的工作效率。如果你让人 们变换工作方式,他们可能会需要不同的工作空间、更多设备或是其他东西。
- □ 在讨论时,给他人思考的时间。在推行你自己的想法时,要允许别人认真思考和质疑 这些想法。有些人可能需要多一点时间来提供有价值的反馈。
- □ 不要固执己见。如果你借助影响力,以协作方式工作,别人就能改善你提出的解决方案。要确定你不会影响他们。我们有时会发现很难抛弃过去卓有成效的想法。要记住,你可以把自己的想法作为解决某个问题的起点。

在项目进行过程中,很多团队对缺陷都采取放任自流的态度。直到项目快结束了,他们才会 认真考虑如何管理缺陷。如果使用顺序式生命周期,在项目开始时可能不会有多少编码缺陷等待 审核。

V/ ・・・ 小^{乔爱问・・・・・・}

如果运营与开发同时进行,我该如何应对干扰?

这与选择何种生命周期模型无关。如果你正在研发新产品,而运营相关的工作不断出现,这就说明你的估算有问题。(在提供支持和开发工作同时进行时,也会发生同样问题。)来自运营的干扰会毁掉你的项目。

可以考虑下列建议。

- □ 调出几个开发人员,用1~2周的时间专门负责运营工作。并且在运营期间调换这些人员。
- □ 假定每个人每周只能用2~3天进行开发,其余的时间都用来处理突发任务。每个人在 一天之内不能从事多个任务。
- □ 加入更多人手,这些人必须是喜欢从事"灭火"任务的。他们的首要职责就是解决 突发问题,接下来才是项目中的任务。
- □ 成立一个小组,组员的职责就是负责运营。
- □ 如果项目经理使用迭代,并以相对大小和持续时间进行估算,不妨估算每项运营工作,然后将其加入到产品的待办事项列表之中。

不管选择何种方式,你都要对运营工作和开发工作负责。

如果项目经理不在项目初期就开始管理缺陷,那缺陷就会反过来控制你。项目的技术债务会不断增长,项目经理不到最后根本无从知晓。到最后就会有太多缺陷需要修复,也就没法全部搞定了。

项目经理有下列选择。如果条件允许,不妨转而使用敏捷生命周期,开发人员可以与测试人员同时进行开发和测试。团队报告的总体缺陷数目会减少,项目经理也能更快了解到缺陷,并且 马上着手处理。

要是不能使用敏捷生命周期,就用增量式生命周期吧,还要确保开发人员随工作进展进行持续集成。让测试人员用不同方式来测试开发人员已经完成的功能。

在缺陷跟踪系统中组织缺陷时,要考虑如何对缺陷分类。可以为每个缺陷定义严重程度和优先级。而且,项目经理也不要对后来出现的缺陷玩"缺陷提升和降级的游戏"(参见原书15.4.2节)。严重程度属于问题的技术层面。如果严重程度很高,系统就无法运行,要么就会提供错误结果。优先级属于问题的业务影响。如果优先级很高,客户会受到问题的严重影响。

在评估缺陷时,有些团队使用类似风险分析表的表格。图4-2中的缺陷并不是一个很大的技术问题(在严重程度上比较低),但是优先级比较高,因为很容易造成混淆。

缺陷序号	简要描述	优先级	严重程度	暴露程度	何时修复?
17	姓名与地址 混淆	盲	低	高(客户会感 到迷惑不解!)	当前迭代

图4-2 缺陷评估表

表中"何时修复?"字段,可能存在缺陷跟踪系统中。我见过团队使用日期、发布版本、迭 代填充该字段,都取得了很好的效果。

₩ 铭记在心

- □ 作为项目经理, 你要带头考虑使用或调整哪些管理实践。
- □ 评估项目的问题,然后根据这些问题来判断使用或调整哪些实践。
- □ 要寻找可以建立和维持项目节奏的实践。

创建并使用项目仪表板

丁页 目经理面临的绝大多数问题最终都会归结到:"我们的进度怎么样了?" 问题可能来自于高层,他们想知道你能否在截止日期来临之时交付产品;也可能来自项目团队,他们想知道自己的进度如何。要不了多久,项目经理就会觉得:这就像是在长途旅行中,他们坐在汽车后座上,不停地问:"我们到了吗?"

要想真正掌握一个项目,关键在于经常进行测量,包括定性和定量两种方式,并且要将结果公之于众。项目经理将测量结果作为项目进度的一部分展示出来,团队成员就可以调整自己的工作,并取得更多成果。

这些测量构成了项目仪表板。将测量结果放在一起,就能告诉大家团队的开发速度、整体进度、消耗和当前所在位置,这跟汽车仪表板的作用很像。

创建项目仪表板可以为团队提供反馈,同时让其他对项目感兴趣的人了解进度。可以使用"大的可见图表"(Big Visible Chart)或是"信息辐射器"(Information Radiator),这样所有的人就都可以看到项目进度了。

5.1 测量有风险

测量项目会面临三个大问题:项目团队花费过多时间进行测量,从而影响正常工作;人们不认真对待测量;测量人,而不是项目。

很容易就可以发现,人们在测量上耗费了太多时间。你的项目团队成员是不是总在写文档、做测量,而不是进行与项目直接相关的工作?本章中的测量都是要由你——项目经理完成的。团队的绝大多数人不应该帮你获得测量结果。有些人负责管理配置管理系统或缺陷跟踪系统,你可能需要他们帮助。(如果项目经理需要测量性能或可靠性,就会需要开发人员或测试人员的协助。)要是你需要很多人配合,还是先加强项目在一些基础设施上的支持吧。仪表板的目标是为了使用数据来评估项目状态,而不是为了花费时间创建仪表板。

人们不认真对待测量活动,一般是很难发现的。不认真的原因通常是因为只测量了项目中的一种因素。项目经理可能会见到日程安排游戏,或是其他无助于项目进展的行为。有一幅很著名的呆伯特漫画,其中老板承诺说他会付现金给修复bug的程序员。沃利(漫画中的一个角色)说道:"我要通过写程序获得一辆小货车。"沃利打算搞上一大堆bug,然后再修复它们。如果只测量一种因素,就等于是在鼓励人们只注重这一件事情。要确保存在多种测量方式,以评估项目的真正进度。

在选择测量方法时,要确保以项目和产品为测量对象,而不是对准人。如果测量的任何东西可以直接跟踪到某个人身上,这就等于是在对人而不是项目或产品进行测量。测量人就等于是鼓励他们不认真对待测量,项目经理会因此难以准确地了解项目的状态,而且整个项目也有可能无法终结。绝不要测量人。

项目的有些方面很容易测量,比如项目启动日期、当前日期、预计发布日期,还有"我们已经完成了百分之X了",因为项目团队就是用时间的百分比进行估算的。如果你只去测量日程安排完成情况,那就肯定不能在截止日期之前交付项目。

实际上,任何单一维度的测量都不足以全面反映项目状况。好比说你要开车去某地,也知道整个里程,可你却既不知道自己车的油耗,又搞不清楚还有多远的路程。在这种情况下,你不可能知道车上的油量能否让自己到达目的地。

要想看清项目真正的全貌,看看1.2节中提到的项目驱动因素、约束和浮动因素,根据这三者排列组合形成的6个维度,可从中选择出至少4个维度进行测量,再显示在项目仪表板上。这4个维度涵盖了项目经理最有可能修改的项目因素。如果不衡量它们,那项目经理就无从得知应该调整哪些因素,才能使项目取得成功。

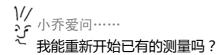
提示: 使用多维度的测量来评估项目讲度

常言道:"衡量什么,得到什么。"而且,人们有可能不认真对待测量活动。既然你所测量的指标会产生最后的结果,就要保证测量足够多的项目信息,这样才能得到项目进度的真实反映。

工程副总罗伯给我打电话,他气急败坏。"JR,这些该死的测试人员!他们什么都做不好!" 罗伯的项目有1500个开发人员,还有大约350个测试人员。之前我已经见过一些测试人员,所以我说:"有意思,不过好像我见过的人都自我感觉不错。""不可能,"罗伯语带讥讽,"开发人员每个里程碑都做到了,可测试人员一个都没有完成。我希望你赶紧做个评估。"

嗯,"开发人员能够完成每个里程碑"听起来有点可疑。我认识很多开发人员,即使是他们 之中最棒的也不能做到总是按时完成。我以开明的态度开始评估,也许所有的1500个程序员都是 无与伦比的。

不过我发现的情况是:开发人员只需向项目经理报告完成日期即可。这就是原因。项目经理 只按日期考核开发人员,只按缺陷考核测试人员。除此之外,这个项目不测量任何数据。当我跟 开发人员谈论他们的工作后,一切真相大白。开发人员丹尼表情痛苦,并解释道:"我必须开始 开发功能,因为甘特图上是这么安排的;否则,我的绩效评估就会打折扣。我只写好方法存根, 这样就准时'完成'了。等测试人员报告一个问题,我就解决一个问题。"



可以,不过我不建议这么做。因为导致问题的机制(流程和人)——比如项目晚于计划一个月启动——不会发现自身的缺陷。如果没有图表展示为什么项目从一开始就落后了,项目经理就无法改变原有的项目启动机制。

与其重新开始测量,不如在图表上画一条线,标明"原定启动日期"和"实际启动日期",然后标明导致项目推迟启动的触发事件(见图5-7)。

罗伯所在组织的项目(和产品)完成得不好,无法按罗伯的要求交付。只要他坚持用单一维度来衡量团队的工作(只按日期考核开发人员,按缺陷考核测试人员),他们还会做出糟糕的项目。对罗伯来说,唯一的解决之道,就是让项目经理全方位测量项目,这样他们就能更准确说明项目的状况了。

5.2 根据项目完成度来衡量进度

驱动因素、约束和浮动因素这三者构成6种组合,项目经理可以从中选择几种衡量方式,根据项目的完成度来衡量团队进度。团队预先估算的准确性和团队已完成的进度,决定了项目完成度。不过仅从时间表上衡量还不够完善。要想准确判断软件项目的实际进度,应该衡量团队已经完成多少功能、已完成功能的质量以及还剩多少功能尚未完成。

5.2.1 使用速度图表跟踪日程安排进度

如果项目采取按功能逐个实现的方式,图5-1中所示的速度图表可以很好地指示出团队的项目进度[©],而且它还能告诉项目经理剩余多少未完成工作。

① 请参见http://www.xprogramming.com/xpmag/jatRtsMetric.htm。

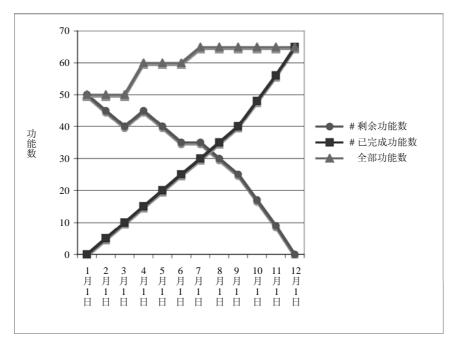


图5-1 项目速度图

可以按照下面的步骤来绘制速度图。将所有的功能汇总,得到项目的全部功能数。开发完一个功能,就将已完成功能数加1,并减少剩余功能数。如果必须在项目进行过程中添加新功能,就得把这些额外的功能加入到全部功能数中。即使这些功能的规模并不齐整,速度图也能有所帮助。

如果使用"小石子"切分任务,而且没有按功能逐个实现,那么跟踪"小石子"的完成状况 有助于项目经理了解目前进度。不过相比按功能逐个实现来说,这样做的准确性并不高。无论选 择何种方式,不要仅询问团队成员是否达成他们的里程碑,还要记得检查已完成工作的质量。

提示: 速度图是最佳图表

如果项目经理只能绘制一个图表,应该选择速度图。速度图集三种度量方式于一身(需求、已完成工作,还有时间)。虽然项目经理无法从中看到自己希望了解的是缺陷率或是成本,却能从该图中对项目的整体进度有所掌握。

这是因为你在一张图中同时度量了多个趋势:整体需求数量和已完成工作,其中包括所有的测试、文档以及项目需要的其他东西。这是最有用的图表。如果项目没有采取按功能逐个实现的方式,速度图中就不存在任何已完成的工作,而这正是项目的真实状态。速度图是项目经理的好朋友。

5.2.2 使用迭代内容图跟踪总体进度

除了使用速度图跟踪已完成功能之外,项目经理可能还需要详细了解每个迭代中的工作进展。(即使你没有使用带有时间盒限制的迭代,也不妨每过一段固定的时间就产生该图表。你会从中了解需求何时发生变化,缺陷何时出现。)

在图5-2中,可以看到发布的内容随时间变化的过程。在这个项目中,团队开始时的速度是每个迭代6个功能。进入到第9个迭代时,团队的速度降至2个功能,外加2个变更和4个缺陷。此时,项目经理意识到这样下去会出问题,于是停止在迭代中修改迭代的待办事项列表。团队的速度也因此在最后三个迭代有大幅提升。

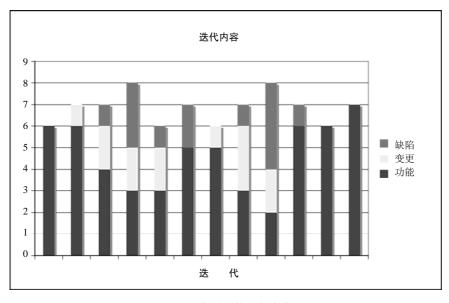


图5-2 一个项目的迭代内容图

在项目经理做出这张图之前,没有人了解迭代中发生变更的成本,也不知道这些变更会引入哪些缺陷。

5.2.3 计算软件项目的挣值并无实际意义

挣值可以用来衡量到目前为止已完成工作的价值。[©]然而软件总是在不断发生变化,要想计算出真正的挣值几乎是不可能的。如果无法明确定义一个指标,也就无法真正测量该指标。应该

①来自© 2007 R. Max Wideman, http://www.maxwideman.com; 经许可后复制。

抵制组织要求项目经理报告挣值的企图。一个有形的产品,计算其挣值很简单。如果你要打造一个桌子,可以计算出物料的成本和花费的时间,看看桌子腿和桌面在拼合起来之前是否有价值。 然而软件的挣值有所不同。

举一个例子。假设项目经理要在10个星期之内完成5个需求。设想你和项目团队认为整个团队在每个需求上要花费2周的时间。而且,假如团队由5个人构成。你的估算就是每个需求要花费相当于1个人10个星期的饱满工作量,总计需要1个人50个星期的饱满工作量。假定团队已经完成了前三个需求,包括测试,如图5-3所示。

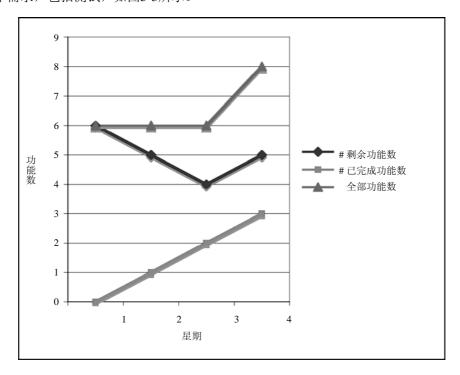


图5-3 六周的速度图表

客户见到团队已经完成的工作后,说道:"看起来不错,可我现在真地希望能在这里加入这个功能,在那里添加那个功能。"

"这个"和"那个"功能每个要花费团队另两周的时间。项目经理原先的计算认为,你们已经在60%的时间内完成了60%的功能。一切都在正常轨道上进行。而现在离轨道可差得远了。但是你提早获得了客户的反馈,而不必等到项目结束,这样你就可以为客户提供他们真正想要的东西。

ツック 小乔爱问……

为什么我们没有已完成的工作?

你已经在项目中投入了几个月的努力,也没有人偷懒。可是当你试图绘制速度图时,发现你们没有任何(或是几乎没有)已经完成的工作。怎么可能呢?这是有可能的。如果使用顺序式生命周期,或者用其他生命周期,但是按照架构进行实现、没有计划好如何完成功能的话,那可能性就更大了。

如果项目团队使用顺序式生命周期或是按照架构进行实现,他们就会产生很多部分完成的工作。在精益社区中,部分完成的工作被称为"浪费",因为它们没有完全完成。速度图展示的是已完成的工作,项目经理可以了解到团队是在制造浪费,还是在开发完整的产品。

如果更多地使用增量式,甚至是敏捷技巧,你的速度图就会更多地展示出取得了哪些 已完成的工作成果。让项目团队看到完成了哪些工作,这对维持项目节奏有好处,而且能 让人们取得更多成果。

现在你获得了多少价值?我真不知道如何回答这个问题,因为这无法解释下面的事实:客户意识不到他们想要的功能在项目所分配的时间内完成是不可能的。最开始的度量方式并不正确。项目是有一些价值的。也许到目前为止,项目的价值要比项目经理想象得多,因为客户提早了解到之前所提的需求不太准确。但是项目的完成度恐怕就不是60%了,而是其他的什么百分数。

有些组织喜欢使用"完成百分比"。我也不赞成这种说法。这经常只包括开发工作,而未将测试涵盖在内。没有经过测试的产品功能就不能算完成。使用"完成百分比"会导致人们实施日程游戏。

如果项目经理想了解真正的进度,可以使用速度图展示正在测试的功能。速度图能够揭示团队的真实进度,而不是计划进度。它还能展示出项目发生的变化以及正在发生的变化数目。

所以, 跟挣值说不吧。用速度图取而代之。

5.2.4 用 EQF 跟踪最初的估算

Tom DeMarco在中描述了估算质量因子(Estimation Quality Factor,EQF)的度量方式。EQF 可以使项目经理了解最初的估算质量。在项目中每隔一段时间,团队就要回答这个问题:"你们认为项目什么时候可以完成?"团队就项目预期结束时间达成一致,这个预期时间就是一个数据点。项目结束后,可以从发布日期到项目启动日期之间画一条直线。看看图5-4这个例子。你所绘制的线和"我们将何时完成"的线之间的区域反映了团队估算的准确性。这是一个很好的技术,

人们可以用之来提供反馈,检查个人的估算。即使不将其用作反馈,项目经理也可以用之观察项目实际发生状况。总之,这是一个很好的技术。

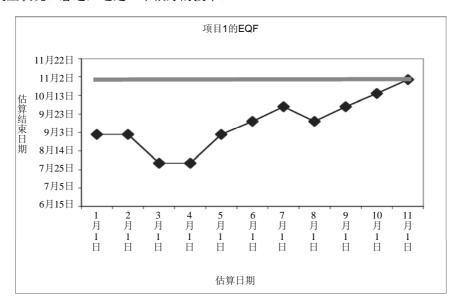


图5-4 估算质量因子

也许你在想: EQF不利于后期发现新的需求。没错, EQF不是完美的度量指标。不过, 要是项目经理没有使用敏捷生命周期, 迟来的需求(或后期搞清楚的需求)的确会带来影响。我宁愿了解项目延迟的原因, 而不愿对其一无所知。

如果使用敏捷生命周期,项目经理的速度图会提供量化的答案,而不是定性的结论。要是没有用敏捷生命周期,EOF是一种很好的量化度量方式,可以用来观测估算的准确性。

图5-4是一个项目的EQF图,该项目预计9个月完成。在前几个月,当项目经理询问团队何时完成时,他们说是9月1日。前几个月,他们都很乐观,认为可以提前完成。但是到了第5个月,大家认识到他们对需求理解得还不够。他们的发现改变了整个架构,并推迟了完成日期。接下来的几个月,他们仍然不确定何时完成。到了项目的最后三个月,他们发现:由于架构发生改变,带来许多当初未曾预料到的问题。作为一种依据进度图表的检查,评估EQF这种定量度量方式对项目经理和团队很有帮助。

EQF不仅可供软件项目使用。任何项目工作都可以使用这个技巧。我在写这本书的时候就用了它。你可以用其帮助开发人员(或是测试人员、文案人员等其他人),指导他们提升估算准确性。

汤米在开发一个功能,他认为自己用三周可以完成。他确定自己每周都可以交付一些"小石

子"仟条。完成一个"小石子"仟条后,他更新了自己对于该功能的EOF,见图5-5。他觉得自 己运气不错,因为可以提前交付这个任务。一直到该功能开发中期,汤米都没有改变自己的EOF, 即使他已经提前成功完成了该功能的前一半任务。

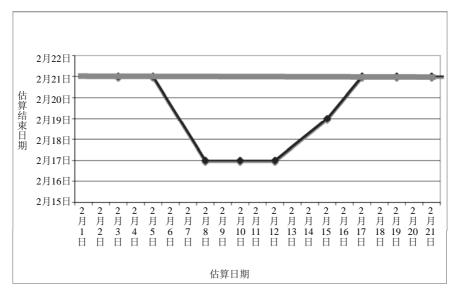


图5-5 汤米的估算质量因子

当汤米继续前进时,他就没有再像预先规划的那样取得进展了。虽然仍符合最初的估算,但 是不可能提前交付这个功能了。

日程估算只是猜测而已,如果可以展示出实际进度与初始规划的差距,并解释其背后的原因, 对于希望知道目前实际进度的人来说是很有帮助的。

5.2.5 更多的度量能提供更多信息

也许管理层只关心项目完成的度量数据,可如果你是团队的项目经理或是技术带头人,我想 你一定希望尽早得到项目日程可能不准确的警告信号。为了保证了解项目的一举一动,我会监控 如下多个数据。

- □ 除EQF之外的日程估算与实际值。如果使用速度图,其中会部分体现该数据。
- □ (有适当能力的)人加入团队的时间,以及实际需要他们的时间。
- □ 整个项目过程中需求发生的变化。如果使用速度图,其中会部分体现该数据。
- □ 如果没有使用敏捷生命周期,还要知道整个项目过程中故障反馈率(fault feedback ratio)

的变化。

- □ 整个项目过程中的缺陷修复成本,特别是在没有使用敏捷生命周期的情况下。
- □ 整个项目过程中,缺陷发现/关闭/未关闭比率值。

注意,这些是评估使用的度量数据,而不是试图找到项目中存在问题的度量数据。这些度量会暴露问题,但是仅凭它们不足以发现真正的问题。要把这些度量数据放在一起分析,才能发挥度量的威力。

5.2.6 如果只能度量项目日程,那就这么做

作为项目经理, 你所在的组织也许一直都在使用顺序式生命周期, 也许你刚刚加入某个项目, 老板认为你的工作做得不能令他满意。这时,可以首先度量项目日程,看看到底发生了什么。

没错。注意我可没说只度量日程,我是建议这是你首要进行的度量工作。不过可别只用甘特图。项目经理有很多工具可以用。比如,可以看看团队应该在什么时候达成某个里程碑,再看看他们实际上何时达成该里程碑,如图5-6所示。要是团队启动项目时就已经落后了(无论第一个里程碑是什么),这个项目就不会在期望的结束日期完成。

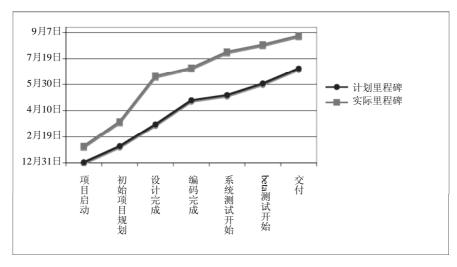


图5-6 估算日程和实际日程

失去的时间永远无法恢复。

图5-6中展示了一个项目发生的状况。这个项目使用了修改过的瀑布生命周期(下阶段工作的开始不必等到上个阶段结束),但是没有用迭代。注意项目的启动整整晚了一个月。当项目经

理贴出这个图时,他同时对管理层说道:"别指望我们可以赶上一个月的时间。我们起步晚了, 无法弥补失去的时间。"他对项目团队说:"我希望你们抓紧时间工作,但是不要加班,也别把自 己搞得太累。我们可没有容许大家犯错误的时间。尽全力做出最好的表现吧,我们会一直跟踪咱 们的进度。"

作为一个讲求实效的项目经理,如果团队按时启动了项目,你可能希望让团队看到他们的进度状况如何。想看到这会产生什么样的图表,请看图5-7。

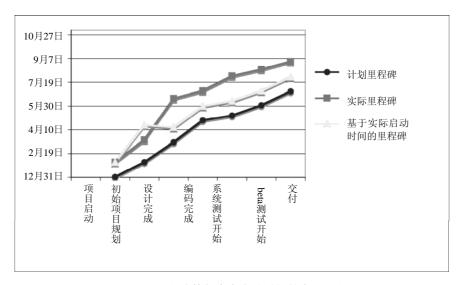


图5-7 日程估算与真实启动时间的实际日程

5.2.7 与项目团队人员分配情况相关的图表

项目在资源不足的状况下启动,这种情况经常发生。很多时候,项目最缺乏的资源是人。而且,人是软件项目取得进展的必备资源。然而不是任何人都行,这些人必须能够完成这个项目需要的工作。

打算仅用小团队启动项目,到了后期再加入新成员,这样做没问题,只要你做好计划。在此类项目中,计划就等于是在说:"我们现在有这些人就够了,以后需要更多。"有些项目启动时缺少需要的人,与此类状况不同。我曾启动下列这些项目:有的项目在一开始就有做原型的人,而且团队中有技术带头人;有的项目需要我去要求开发人员修补缺陷,因为没有技术带头人;有的项目使用短迭代,而我只有开发人员,没有测试人员。这些项目中,我们都有计划,可以在项目需要其他人加入时,将他们顺利整合到团队中。

如果启动一个人力不足的项目,这就等于自寻烦恼。(如果项目启动时没有足够的计算机或

是办公桌等其他资源,只要团队可以处理好这些资源稀缺问题,这就可以接受。)项目经理要了解团队中有多少人,还得知道他们能否把工作做好。

要是项目经理面临类似状况,就使用带有时间盒限制的迭代吧,然后测量迭代的速度。你就可以把数据展示给团队,同时不断去跟那些仍不肯释放资源的人解释这些数据。如果总发生类似情况,再想想这个工作是否还值得做。

图5-8展示了一个真实项目的人员变动历史。到了第2个月,此时项目需要的人仍然在做之前项目的工作(那个项目已经拖期了)。管理层没有继续等下去,而是让项目经理启动项目,他也遵命而行。到第3个月时,项目团队只有4个人,而不是所需要的10个人。这种状况下,项目经理和团队打算改变工作方式,却被告知其他的人可能"随时"到位,而项目经理和团队也信以为真。

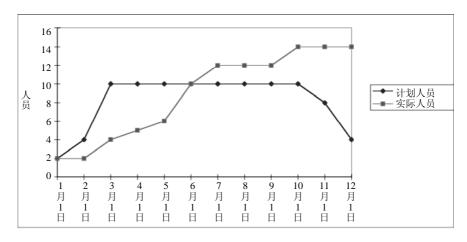


图5-8 项目计划人员安排与实际人员安排

到了第6个月,团队人员终于全部到位,可是进度却远远落后。为了赶进度,项目经理要求 更多的测试人员,也如愿以偿。(该项目使用顺序式生命周期。)测试人员发现了很多缺陷,因此 项目经理要求给予更多开发人员。于是开发人员创建出更多缺陷,测试人员又发现更多缺陷,最 后,他们终于到达了一个相对稳定的点。

这个图度量了分配到项目中的人员趋势,而不是总数。如果把人员总数加起来,项目用到的实际人月要多出三分之一。人员成本不是这个项目的约束因素,所以问题不大。真正的问题在于:团队只交付了之前期望完成的三分之二功能,而且系统也不是特别稳定。

如果项目经理面临人员不足的状况,要小心不要陷入上面例子中项目经理所面临的同样状况,不要认为你不必重新设计项目计划。这个项目到最后还是分配了很多人,而下一个项目已经 开始迫切需要人力了。但是项目经理从这个项目中学到很多经验。在他负责的下一个项目中,他 让两名开发人员按功能逐个实现,并使用为期一周的迭代。如果项目人员的到位状况无法达到当 初规划项目时的要求,那就重新设计项目计划吧。对我来说,这几乎就等同于转向敏捷开发,因 为它能让我在安排人员时拥有最大的灵活度。

5.2.8 判断项目的变化率

如果项目经理使用顺序式生命周期,也许无法绘制出有实际意义的速度图,因为你会被为数众多的未完成工作所淹没,同时又没有多少实际完成的功能。我仍然推荐使用速度图,不过项目经理需要将其拆分成几个部分。这种状况下,项目经理可以使用需求变更图,如图5-9所示。

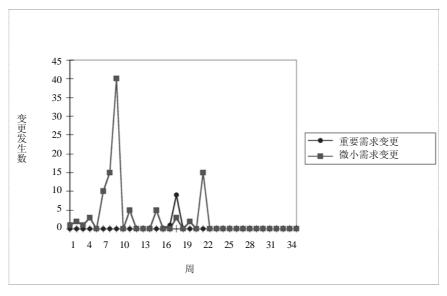


图5-9 需求变更图

我曾出任过一次类似情况下的项目经理。当时我中途加入了一个采取顺序式生命周期的项目, 无法测量其速度。我们有太多没有完全完成的工作。不过我可以开始测量需求变更情况。在这个项 目中,我有一个简单的标准,用来判断需求变更的重要程度:基于模块间接口的变化情况。因为这 种变化很容易产生缺陷。只影响一个模块就是微小变更,如果多于一个模块受影响,就是重大变更。

在图5-9中,有许多微小变更,这也是我们希望在项目中看到的。不过到后面(第22周),也 遇到一些重大需求变更。看到这些需求变化后,我就能向管理层解释项目可能会延期,或是缺陷 数目会增加。有了记录的变更,我们就很清楚地知道:最初预计的交付日期、有少量缺陷数目的 最初功能集合是无法达成了。

5.2.9 查看开发人员是在取得进展还是在白费时间

团队进入代码开发阶段后,项目经理就可以马上开始测量故障反馈率(Fault Feedback Ratio,FFR)了。FFR是指被拒绝修复的缺陷数(也就是没有真正解决问题的缺陷数目)与修复缺陷总数之比。经验告诉我,如果FFR高于10%,就说明开发人员遇到了问题,难以取得实际进展。

成功的敏捷项目中,开发每段代码都会用到测试驱动开发、结对编程和单元测试等实践,FFR 会很低。而没有使用持续集成和持续代码复查的项目,其缺陷会不断累计,并欠下重大的技术债务(请参见原书附录B)。这时测量FFR就显得非常有用(如图5-10所示),从中可以了解到开发人员实际解决了多少缺陷,并与被拒绝修复的缺陷数做比较。

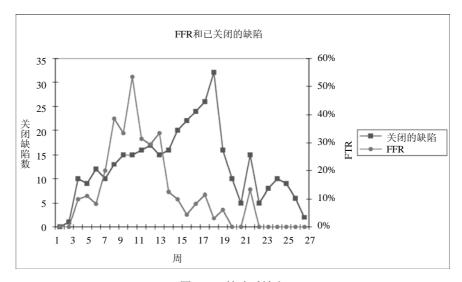


图5-10 缺陷反馈率

在图5-10中,当开发人员使得FFR降低后,成功解决的缺陷数就会增加。当FFR居高不下时,项目经理可以确定:开发人员们修复的缺陷数会减少,除非他们可以修复那些阻止他们前进的缺陷。

测量FFR可以每周一次,并可与开发人员和测试人员讨论该数据。如果看到某一周的FFR很高,先检查这一周修复的缺陷总数。如果修复了4个缺陷,有1个被拒绝了,开发人员跟测试人员可能遇到的问题不大。如果看到修复的20个问题中有5个被拒绝了(25%),很有可能是某个人或某些人遇到了麻烦。在图5-10中,可以发现在第6周前后FFR升高了,而且在第13周之前一直高于10%。一旦项目团队连续两周保持高FFR,项目经理就应该针对所有的修复开始同行复查(peer review)了。这会起效,FFR会降低到正常水平,缺陷修复不再会干扰正常进度,但是这个阶段与启动同行复查时刻之间会有时间差。

要想发现问题区域,不妨先问问开发人员,看看他们是否在修复时遇到了问题。一般我会这么问:"你在修复这个问题时,是不是会在其他地方出现新问题?"我还可能问其他问题,最终都要问开发人员是否需要其他资源,以修复该问题。如果我听到开发人员希望重新设计某个模块,我们就会讨论这样做产生的影响。

接下来的问题是给测试人员的:"你能否确定发生该问题的所有条件?"我从这些问题开始,看看开发人员是否能够一次解决问题的某一方面,或者了解测试人员是否足够了解系统从而进行全面的测试。

FFR是一种较迟的测量指标,项目经理要通过提问帮助团队解决问题。你可以在任何工作产品上使用FFR,不过人们不大愿意重新处理设计文档或是需求文档上的缺陷。没有代码,就无法测量FFR。因此,在顺序式生命周期中,越早开始测量FFR,项目经理就可以得到越多反馈。

5.2.10 测量查找和修复问题的成本

如果项目经理没有使用少于4周的迭代,团队查找和修复问题的成本就是一个关键测量指标。 也许你已经见过类似图5-11的行业标准数据。

阶段	需求	设计	编码	测试	发布后
成本	1	10	100	1 000	10 000
项目1成本	未测量	未测量	0.5人日	1人日	18人日
项目2成本	0.25人日	0.25人日	0.5人日	0.5人日	8人日

图5-11 两个项目中修复缺陷的实际成本

这里是说:在需求阶段修复一个问题耗费1个成本单位,设计阶段修复一个问题耗费10个成本单位,编码阶段修复一个问题耗费100个成本单位,测试阶段修复一个问题耗费1000个成本单位,发布后修复一个问题耗费10000个成本单位。我们通常以美元或其他货币作为成本单位。

我曾测量过修复一个缺陷的成本, ①.②发现这个数字在不同情况下各不相同。图5-11中展示了

① 请查看http://www.jrothman.com/Papers/Costtofixdefect.html。

② 请查看http://www.stickyminds.com/s.asp?F=S3223_COL_3。

两个项目的测量结果。项目1在出现问题时没有立即找到缺陷并移除,团队在代码中查找问题时 三心二意。很多缺陷都是在测试时发现的。项目2从一开始就采取积极方式主动寻找缺陷。

要记住,重要的不是每个缺陷的成本,而是每个缺陷的成本乘以缺陷总数。要是不查看总成本,项目经理就不会知道该如何利用时间。基于之前项目修复缺陷的成本,项目经理可以判断是要采取积极手段,比如从一开始就查看关键项目文档、实施测试驱动开发或结对编程,还是只处理更棘手的缺陷,或者你可以监控修复缺陷的成本,并选择更为灵活的响应方式,比如同行复查修复代码或是检视所有的代码。

如果项目经理从未实施过积极的缺陷查找活动,找到一个缺陷的成本就会比较低,而修复一个缺陷的成本就会很高。修复所有缺陷的总成本将会非常高昂,因为如果团队中没有积极寻找和修复缺陷的文化,那一定会产生很多缺陷。如果一直在使用诸如测试驱动开发、结对编程、代码检视、同行复查等积极手段,那么找到一个缺陷的成本可能相对较高,因为已经付出了寻找缺陷的成本。但如果团队可以及早发现缺陷,修复一个缺陷的成本将会大幅降低。缺陷的总体数目也会减少,对于项目的发布来说,修复缺陷的总成本也就降低了。

我会监控寻找和修复缺陷的成本,这样就能了解代码库中的代码是否会让开发人员或是测试 人员感到吃惊。我有如下一些基本原则,同时假定开发人员不曾主动寻找缺陷。

- □ 开发人员修复一个问题所用时间越久,他们就越有可能不敢碰触系统的某些部分。这很有可能是因为开发人员对系统的某些部分不了解。
- □ 测试人员找到问题所用时间越久,他们就对产品了解得越少,或者他们测试产品的手段 就越匮乏。

修复一个缺陷的总成本越高,项目经理就越有必要管理缺陷相关的风险。比如,为了发布某个版本,何时停止接受修复,以及要修复哪些缺陷。

5.2.11 了解开发人员和测试人员是否在解决缺陷方面取得进展

绝大多数项目都会测量缺陷趋势。我曾见过一些很复杂的缺陷趋势图表,不过我最看重三个指标:每周发现的新缺陷数、每周关闭的缺陷数、每周仍未关闭的缺陷数。如图5-12所示。特别要说明的是,我不会在图表上标明缺陷的优先级,因为这会使得团队和高级管理层非常愿意玩出"缺陷提升和降级的游戏"。而且,无论缺陷的优先级有多低,开发人员也必须要把所有缺陷从头看到尾。所以我只要统计所有的缺陷就好了。

我计算尚未关闭的缺陷数目,是为了知道缺陷关闭率何时可以超过缺陷发现率,此时尚未关闭的缺陷数目就开始减少了。我希望看到尚未关闭缺陷数目曲线的拐点,当这条曲线的斜率变成

负值的时候, 当前发布版本的风险也就减少了。

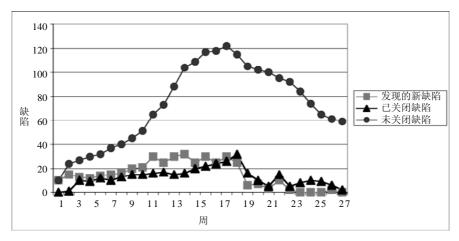


图5-12 一个项目过程中的缺陷趋势

5.2.12 展示测试过程

项目开始测试后,项目经理就可以马上绘制测试进度相关的图表了。无论你采用何种生命周期模型,我都推荐将测试与开发结合在一起。请看第13章。

在图5-13中,你可以看到规划的测试数目在不断增加,这是因为需求在不断变化。从中还可以看到,团队能够运行的测试数目在稳步提升,运行通过的测试数目也是如此。不过,在运行通过的测试数目与能够运行的测试数目之间有明显差异。

该图来自一个真实的项目,其目标是要发布带有新功能的新版本系统。在开始时,团队已经有了900个可以运行的自动化回归测试。可是当测试人员开始工作时,只有600个可以运行通过。这是因为团队没有使用持续集成,而是从一开始就采取了按阶段集成方式。当他们看到图表之后,意识到给自己增加了这么多额外的工作,他们就开始转向持续集成了。

测试进度图是用来衡量测试进度的绝佳手段。不过它也可以展示出(可能是不可预知的)需求的变更情况,同时也能从中看到开发人员集成工作的效果好坏。

如果采用敏捷生命周期,那么针对功能的测试就会在开发这个功能的迭代中进行,项目经理 也许就不再需要上面这个图了。要是项目团队无法在迭代中完成测试工作,这个图还是会有帮助, 特别是按迭代进行绘制的时候。如果使用迭代并且在迭代中开发功能,在迭代的开始就会看到"规 划的测试"数目产生跳跃式增加。

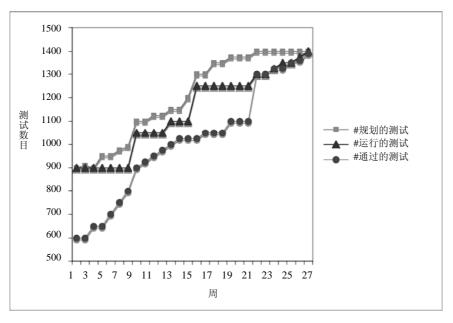


图5-13 测试进度图

5.2.13 展示定性数据

如果项目所有的数据都可以通过趋势图展现出来,那事情就容易多了。不过项目经理需要另外一种图表,特别是试图解释某种状态的时候。

功能、区域、模块	上次测试时间	状态	计划下次测试	
功能集合1	3月3日,构建版本 145	通过	构建版本150	
性能场景3	3月1日,构建版本 140	失败,等待杰夫和安 迪修复	构建版本150	
功能14	3月2日,构建版本 142	通过全部回归测试	构建版本147,供用 户验收	
总体状态	到3月3日上午10: 08	当前迭代开发进展 至中途,看起来在按 计划进行	当前迭代最新构建 版本:构建版本165	

图5-14 测试仪表板

当试图解释算法研究、性能场景和测试^①等工作的进度时,我一直用如图5-14所示的进度图表,特别是针对时间较长的迭代或是顺序式生命周期。

5.2.14 用图表展示团队同意采用的实践

在第9章中,我建议团队可以采纳一系列技术实践。当团队同意之后,项目经理可以发现: 让团队用图表把实践展示出来会很有帮助。项目经理不要自己绘制这些图表。你的团队成员都是成年人了,别把他们当成小孩子。如果他们不去推行实践,项目经理要找到原因。这就是所谓"项目管理"中"管理"的含义。

很多时候都是因为有些东西妨碍团队执行实践。项目经理的职责就是移除这些障碍。可以让团队在回顾会议中用图表绘制出这些实践。

图5-15就是一个按阶段交付项目中实践图表的例子。为了取得项目成功,该团队选择了5个实践:伙伴复查、数小时内修复构建版本、按功能实现、波浪式规划、开发自动化冒烟测试。团队在伙伴复查、按功能实现和波浪式规划都做得不错,但是开发自动化冒烟测试和数小时内修复构建版本这两项就不怎样了。项目经理看到这个图后,不要去强迫让人们做得更好,你的职责是要发现妨碍他们成功的原因。

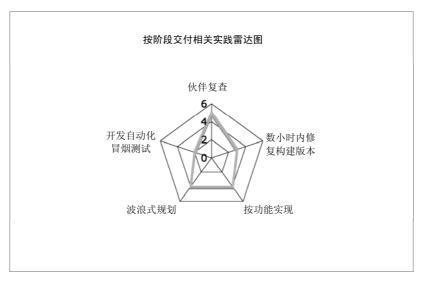


图5-15 按阶段交付相关实践雷达图

蒂娜看到这张图后,她召集了一次项目团队会议,并将解决实践相关问题放入会议提纲。蒂

① 请查看http://www.stickyminds.com/s.asp?F=S7655_COL_2。

娜发给每个人即时贴和笔,说道:"有些东西阻止你们开发自动化冒烟测试和修复构建版本。请你们写下来解决这些问题所需要的资源。每张即时贴上写一个主意。写完之后,把即时贴粘贴到对应着'自动化冒烟测试'和'修复构建版本'的两张挂图上。"

过了七八分钟,大家都完成了。蒂娜大声读出即时贴上的字。她让团队将类似的相关想法分组汇总。对于这个团队来说,问题很明显都是相关的。让人吃惊的是,问题在于自动化冒烟测试使用的源代码控制系统。蒂娜将解决这个问题作为项目新的需求(而不是目标)提了出来,并让团队中能力最强的两个人完成该任务。几周之后,团队重新组织冒烟测试和构建版本,这样人们就可以独立签入他们新增和变更的代码,同时不会影响到别人。

如果蒂娜不去询问团队障碍何在,恐怕她永远不会知道关键问题是什么。

图5-16展示了一个敏捷项目中的实践雷达图。^{①,②}项目经理也许会觉得这个图中的数据有点奇怪。人们怎么能认为自己在测试驱动开发上做得很好,可是在100%单元测试开发上做得很差呢?当查理在团队回顾会议上提出这个问题后,马里奥解释说:"是这样的,我开始时用了测试驱动开发,后来过于兴奋,就忘记要先写测试了。我给自己在测试驱动上打了高分,但是在100%单元测试开发上打了低分。我觉得我大概做到了98%。这可能还不错,但是有几次,如果我真地总是先编写测试,也许有些问题就可以更早发现了。"

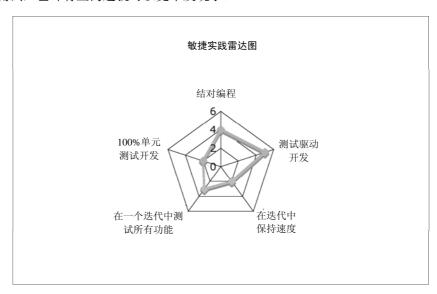


图5-16 敏捷时间雷达图

① 请查看http://www.xprogramming.com/xpmag/BigVisibleCharts.htm。

② 请查看http://xp123.com/xplor/xp0012b/index.shtml。

团队实践图可以让团队开放讨论他们选择的实践,哪些收到成效、哪些运转不畅。项目经理 要知道如何阐释图中的数据,帮助团队发现问题的根源,并解决这些问题。

5.2.15 度量敏捷项目

如果项目经理选择使用敏捷生命周期, 迭代时间长度不超过4周, 同时在一个迭代中, 所有 分配给项目的人员不会被重新分配工作;如果你可以完成迭代中应该完成的工作,包括找到并修 复当前迭代中引入的缺陷,那么你也许只需要速度图、测试进度图和迭代内容图就可以了。至于 是否需要跟踪实践的运作状况,可以询问团队的意见。

5.3 为出资人创建项目仪表板

项目的出资人(或者其他需要听取项目状况报告的人)也许喜好某些特别的进度报告方式。 但愿这些方式都能体现在项目经理的项目仪表板中。要是出资人不想了解任何情况, 那么可以将 仪表板上的数据反馈给项目团队。出资人希望知道项目何时可以完成。为了让他们知道这一点, 项目经理可以告诉出资人项目有哪些风险,以及为了达成发布条件,项目目前的进度状况。

5.3.1 展示风险列表

项目经理在编写项目计划时就已经开始制订风险列表了。风险列表会随项目进度发生变化。

在图5-17的风险列表中,列出了可能阻止项目团队达成项目驱动因素、约束和项目因素的事 物。我也遇到过这样的出资人,他们希望像鸵鸟一样把头埋进沙子里,视风险而不见。不过大部 分投资人都知道:视而不见毫无裨益。

在我的一个项目管理课程中,上课的同学们希望看看项目风险的分布图,如图5-18所示。他 们发现,知道风险在图表中的分布很有帮助,这样就可以对风险有整体认知。我认识的高级管理 层中,至少有一位希望先看看项目风险的分布图,然后再看下面列出的详细数据。

5.3.2 展示在满足发布条件方面取得的进展

查看项目满足发布条件的状况,可以让出资人了解项目的进展,如图5-19所示。如果他们不 让你使用增量式或迭代式的生命周期,除非项目接近尾声,他们不会看到项目在满足发布条件方 面取得多少进展。如果项目经理希望转向更加具备迭代或增量式特性(或者二者兼备!)的开发 方式,就可以开始跟踪项目在满足发布条件方面的进展了。(没错,这是一种游击队式的流程改 进方式。)

风险 序号	风险说明	发生概率	严重程度	暴露程度	发生日期	应对计划
1	露辛达和她的手下 无法在我们需要的 时候参与原型审查	闾	高	高,高	5月1日	1. 向露辛达说明时 间表 2. 及时向露辛达告 知项目进度 3. 提前1周提醒她
2	如果在9月1日之前 发生变化,"供应链" 相关的功能会改变 整个数据库的设计	高	中	中,高「	9月1日	继续与露辛达沟通 这些改变造成的干 扰。告诉她一个我 们可以接受改变的 日期。

图5-17 初始风险列表

风险的严重程度

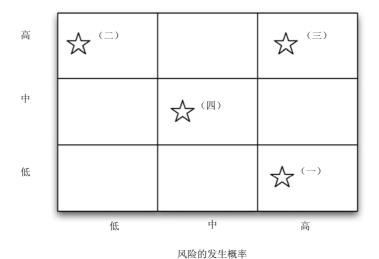


图5-18 风险列表分布

条件	状态	状态	状态	状态
可靠性条件1: 运行13小时	构建版本121, 失败	构建版本123, 通 过,但是发现缺陷 x、y、z	构建版本125, 通过	构建版本127, 通过
复查所有 在线帮助	构建版本121,模 块A复查完毕	构建版本123,模 块B、C复查完毕	构建版本125,模 块D复查完毕	构建版本127,除 模块G之外其他复 查完毕

图5-19 发布条件

5.4 使用项目气象报告

有些情况下,出资人或是管理层(甚至是客户)希望每天都能了解项目的进展。这是"微管理"。人们只有在没有数据或是不理解数据的时候才会选择微管理。如果项目仪表板上的数据对于这些人来说难以理解,他们会不断向项目经理询问信息。要记住,高级管理层会从较高的层面思考问题,而不会深入过多细节。我曾用气象报告[®]的形式,向这些希望了解细节的人报告进度,但是不在项目仪表板的层面上。

很多时候,我经常见到管理层在最后测试阶段进行微管理,所以我见到的气象报告经常基于测试人员的数据。然而,如果你的上司希望"微管理"你的项目,除了项目仪表板之外,再生成一个可以纵观项目全局的"平衡计分卡"气象报告吧。

项目气象报告会评估对比项目的当前状态和它的应有状态。我见过有些项目经理使用红、黄、绿三色交通灯模型来指示项目状态。交通灯模型展示当下的状态,而且易于理解。但对许多项目来说,有很多细微的差别,是只有三种状态的交通灯模型无法表现的。

我也见过一些出资人,他们想知道项目何时转向正轨。"永远不可能",这样的回答虽然准确,却不能为人接受。我也知道,有很多项目经理主动或是被迫地控制交通灯模型。在最后一周之前,项目的交通灯一直是黄色或绿色,到了最后一周,却突然变成了红色。

① 请查看http://www.stickyminds.com/s.asp?F=S10522_COL_2。

一般来说,项目会按照团队现在的前进方向走下去,除非团队或项目经理采取某些行动改变方向。气象报告模型假设:除非项目经理和团队采取行动,否则天气不会转好。想要人为操控气象报告比较困难(但并非不可能),因为其中包括很多状态。

大多数高级管理层希望每周了解项目状况,他们不想费心琢磨你的项目仪表板,而是想让项目经理快速评估项目状况。而且这个评估要能让你和你的听众了解项目进度和未来进展。

气象报告能够让项目经理迅速掌握项目的全景,包括项目实际状态和计划状态,也就是项目的"天气"。如果"天气"一直在不断恶化,而且没有任何改善行动,大家都能看得出来:项目情况会变得更差(预测未来的项目天气)。

5.4.1 要小心定义气象报告的图示

由于项目状态的交通灯模型没有足够的状态标识,项目经理要明确定义不同的气象报告图示对组织的含义。你可以根据组织需要调整定义。下面这些定义适用于我的一个客户。



晴天:项目日程一切正常。



有少许云量:项目在日程上有点小问题,但是仍可以按计划进行。



多云: 日程上有些问题, 但是通过额外的努力, 还是可以按计划进行。



阴: 当前的日程或功能集合有很高的风险。



有雨:在项目目前的状况下,很难按计划完成,或是无法实现所有的功能。



恶劣:无论如何,我们都无法达成项目日程或是实现期望的功能集合。

为什么我推荐项目经理使用"有雨"和"恶劣"?"有雨"意味着"嘿,高管们,咱们得谈谈。我们无法完成你们想要的东西,不过还有机动空间。我们可以放弃某些功能,也许还能再加几个人。延长一下时间表,或者再重新规划一下这个项目。""恶劣"等于是在说:"我们搞砸了重新规划这个项目的机会,不过只要你们不中止,我们就会一直做下去。"

一个关于交通灯和气象报告的童话

本曾是一位项目经理,管着一个为期6个月的项目,其中问题层出不穷,不仅仅与项目相关。参与项目的20个人中,有一位突然结婚了,一位怀孕的开发人员必须有部分时间卧床休息,一位测试人员在滑雪时摔断了腿。但毫无疑问,这个项目的客户非常重要,客户希望得到更多的功能。

在最后3个月的时候,本将项目状态描述为"黄色"。项目还剩6周时间,本向非常重要的客户和管理层解释说,项目的几个新功能很可能无法完成了。没错,项目团队一直在努力工作。他们竭尽全力,希望按时交付,但是他们并没有十足把握。每次进度会议上,本的上司和非常重要的客户都会拍拍他的背,保证说他们对本很有信心。

本准备在下次进度会议上尝试使用气象报告。他将项目状态标注为"阴"。"阴"你是说完不成?"非常重要的客户怒气冲天。本的上司提议短暂休息一下,然后将本叫到一边,对他说:"本,你什么意思?项目状态一直是黄色的。你也总是可以让黄色状态的项目有所突破。"

本解释说,即使之前其他项目也是黄色的,但都没有像这个项目这么危险,难以完成。 "我们真地需要点好运气。"

非常重要的客户询问本还需要哪些资源,并尽早在下周一提供他的实验室和测试人员供本调遣,希望这能有所帮助。本解释说这应该能起点儿作用,这些测试人员可以开始测试已经完成的功能,同时可以继续与开发并行测试。他们最后按时交付了,非常重要的客户提出的额外功能也只有一个没有完成。

在这个案例中,交通灯使得出资人相信本只是比较悲观,而没有认识到真正的问题所在。 气象报告在这里协助出资人讨论真正的问题,并试图解决这些问题。通过改变进度的表现方式,气象报告也可以对你有所帮助。

假设你所管理的项目没有多少风险,而且一直按计划进行,你就可以给这个项目一个大晴天。

要是项目经理的风险列表条目与日俱增,而且你也不确定有两个功能能否准时完成。虽然项目仍然按计划进行,但是你确定将来一定会出问题。又好比现在还是项目早期,但是你的测试团队无法按计划运行测试,更不用提成功通过了。对于这两种情况,项目经理都可以将项目标识为"有少许云量"。这样的状态维持几周,开发人员和测试人员仍然不能取得进展,你就可以将项目标识为"阴"了。

如果项目风险列表在不断增加,开发人员虽然花了不少力气却是白费功夫,项目经理找到的 缺陷也越来越多。这个项目就变成"有雨"状态了。 气象报告模型评估项目数据,以预测项目进度,项目就像是在面临季节的更替。预测来自于我们对于天气的经验,季节性的天气不会发生太多变化。即使有雨雪寒暑,天气总是在按着这个季节的规律变化。项目进度也是如此。项目经理也许会遇到一个可以解决的问题,可如果要是反复发现问题,你就得改变原来的评估了。随着天气图标的变化(或保持不变),看图的人会更清楚了解项目的状态,也许就会想要深入了解仪表板上的数据了。

5.4.2 建立可信的气象报告

如果气象报告每周总是发生剧烈变动,它就会失去了可信度,除非确实发生了某些严重影响项目的事情。一周之内可能影响气象报告的因素包括:相当可观的人员转而从事其他工作、厂商错过截止时间,或是没有认识到架构师无法为规划好的功能集合提供支持。

使用不太职业的气象图标也会影响可信度,尤其是在管理层非常关心工作外在表现的时候。项目仪表板应该让查看的人一清二楚,同样地,气象报告的图标也应该提升项目经理的可信度,而不是造成不良影响。

如果项目经理已经收集了大量项目数据,比如日程相关数据、速度图、缺陷趋势、测试覆盖率、人员分配、风险列表等等,那么气象报告就是评估整体状况的最佳方式了。要是没有收集这些数据,那也别光凭直觉产生气象报告,应该转而展示为了满足发布条件的项目进度。

5.4.3 每周发布气象报告

气象报告的目标是帮助人们了解项目评估状况,不让他们感到意外。还有多于两个月剩余时间的项目应该每周发布气象报告。在某些时间点上,比如项目状态仍在快速改变,或是接近某个重要的里程碑时,气象报告每周可以发布多次。

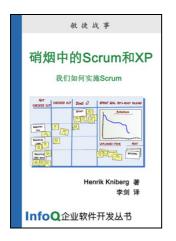
有了气象报告,项目经理的项目状态弹药库中就多了一件武器。可以将项目仪表板作为项目 度量和状态报告手段的首选,不过无论使用何种方式,都要符合项目所在的组织及其环境。

₩ 铭记在心

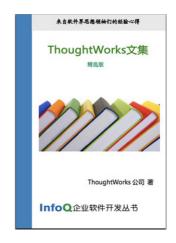
- □ 使用速度图和迭代内容图作为首选。
- □ 数据是工具,不是目的。要记住,图表应该为你服务。
- □ 如果无法获取需要的数据,项目经理就遇到了比数据更严重的问题。先解决这个问题吧。

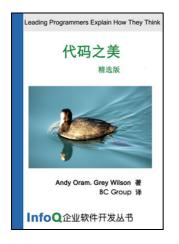
InfoQ企业软件开发丛书

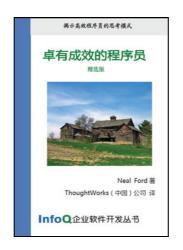
欢迎免费下载

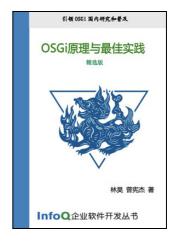












商务合作: sales@cn.infoq.com 读者反馈/内容提供: editors@cn.infoq.com