程序员度量：改善软件团队的分析学

作者：【美】亚历山大, ePUBw.COM

O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了Make杂志，从而成为DIY革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2. 0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

本书由“ePUBw.COM”整理，ePUBw.COM 提供最新最全的优质电子书下载！！！

译者序

软件开发是一种团队作用特别明显的活动，一直以来，软件开发团队的建设就是一个很热门的话题。从数据分析的角度解析软件开发团队建设和改善的实践并不多，之前广为流传的TPE（Team Performance Evaluation）做法，虽然给出了具体KPI，但是如果操作不当，也容易使团队建设成为一种得分游戏。

拿到本书英文版时，我们曾经很犹豫是否有必要将英文版翻译成中文并介绍给国内的中文读者，毕竟从

书名上来看，很容易将其与程序员的绩效考核和评级联系起来，这是我们不期望的。但是细读之后，发现并非如此，遂打消顾虑。本书更多是从程序员度量的角度谈团队的建设，作者相信成功的团队是拥有一定的模型的，将模型量化，由合适的程序员与团队匹配，必将打造成功团队，对于软件开发行业具有点石成金的作用。作为集体性特别强的活动，长久以来认为对于软件开发缺少好的度量方法。有时候人们会采用代码行数、开发中的bug数等作为度量，但这些数据往往掺杂着许多干扰因素，很难将其与团队或者个人的改善结合起来。而本书的作者Johnathan是个棒球迷，通过体育运动与软件开发活动的类比（二者都是强调团队协作的集体活动），将技术统计的概念延伸到程序员度量领域，为软件团队的改善打开了一个全新的视角。方法的实践细致入微，配合作者亲历的案例，很自然地让读者产生试一试的想法。

本书详细解释了如何通过程序员度量来帮助团队更加准确地理解在项目过程中的事件，让团队中的每位程序员可以关注于特定的改善。

本书主要内容如下所示：

·学习如何通过程序员度量改变长期以来的假设，并且改善团队动态。

·获得将程序员度量集成到现有流程的建议。

·提出正确的问题来确定软件开发管理者需要收集的数据类型。

·使用度量来测量一段时间之后程序员个体的技能和团队效率。

·确定每个程序员对团队所作的贡献。

·分析对所开发的软件及其特性的响应，并且验证是否正朝着团队和组织目标而努力。

·建设更好的团队，通过使用程序员度量来进行人员调整和补充。

从译者的实践来说，本书中的方式不太可能全部直接套用，不过作者所提到的各种度量对软件开发管理者来说，至少起到了拓展视野、全面了解团队的作用。

在本书中，作者引用大量的棒球和橄榄球术语，译者在这一方面的翻译也主要寻求互联网的帮助，难免会有错译或漏译之处，请读者批评指正。

最后，我要感谢尹哲的引荐，让我们接触到本书，同时非常感谢机械工业出版社的编辑，感谢他们耐心细致的指导，以及在交稿时间上一次又一次地宽限。参加本书翻译的有周峰、张刚、宋励奋，及我还在上大学的妹妹张育萍。对于他们的辛勤工作，在此一并感谢。

本书的几位译者都已为人父，为人夫，谢谢家人对我们工作的默默支持，在家中构造了一个安静而温馨的环境，以便完成本书的翻译工作。在此，一并致谢对应的家人，他们分别是夏菁华、王雪娟、刘艳卉、吴俊4位女士，及张舒涵、周沫宁、张云桐、宋宇轩4位小朋友。

张燎原

前言

是否存在一种合理的方法来衡量程序员的技能与贡献，并且也同样适用于团队所有的人？是否可以通过度量来帮助个人提高程序员的自我意识，以及促进团队工作、出谋划策和目标设定？能否通过详尽的数据帮助你做出更好的聘用决策，或者更公平地进行绩效考核，从而让你的软件开发团队变得更成功？

无论你是程序员、团队负责人，还是项目经理，如果你对这些主题中的任何一个感兴趣，或者你对如何

采用不同工作方式将度量应用到软件开发团队中感兴趣，那么本书很适合你。本书的思路和过去在软件开发中使用度量的方式有很大不同。本书中提出的概念和技术，旨在帮助你从不同的角度思考构建软件开发团队以及开始你们的新旅程——在软件开发过程中采用更新、更好的度量方法。

作为软件开发团队的经理，我本人亲身实践了这些方法。本书中所提出的技术已帮助一些身处麻烦中的团队走出困境，并且帮助一些本来已经不错的团队锦上添花。固然，本书提出的度量方法并非成功的唯一途径，但它对我来说极具价值，我相信对你也会同样有效。

也许你会通过使用软件的人数、交付版本的效率或者软件中错误的数量来衡量软件的成功。通过使用度量方法可否将这种成功提高5%、10%或者15%？自己测试一下这些想法就知道了。虽然本书描述的方法相当有效，但即使仅仅有5%的改善，也意味着很大的价值。如果度量可以简单地帮助开发者在一个团队里变得更自觉，并继而成长为更好的团队成员，那么这有多大的价值？最起码，我相信潜在的收益足以抵得上你收集和使用书中描述的各种度量指标所花费的时间和精力。即使最终你决定不再收集这些度量，但是我相信通过阅读本书，你仍然能够从中学到很多有用的概念并应用到你的团队中。

本书结构

本书按阅读的顺序分为三部分，虽然在以后将度量方法应用到实践时，可能其中某些章节对你而言更有价值。第一部分对程序员度量背后的思想进行详细介绍，并介绍了多种关于可以从度量获得的分析和那些可用于测量程序员和软件开发团队的数据。第二部分可以作为各种度量方式的参考指南。每种度量方法都包含一些例子和注释。第三部分介绍将度量引入团队并将它们用于开发流程中的一些技术。

第一部分由以下章节组成。

第1章详细解释了本书的构思、动机和目标。

第2章讨论了度量、程序员度量以及团队协作和团队绩效分析背后的基本概念。

第3章讨论有用数据的组成，怎样获取它，以及在程序员度量方法中使用的详细数据元素。

第二部分由以下章节组成。

第4章覆盖各种关于程序员技能和贡献的度量方法。

第5章包括对软件的各种正面、负面用户反馈的度量方法。

第6章包含度量程序员为团队带来价值的方法。

第三部分由以下章节组成。

第7章给出一个多步骤的方案，用于检验和把度量引入一个组织中，并提供了在开发过程和绩效评审中使用度量的方法。

第8章描述怎样使用度量来确定团队的需要，以及如何把它们应用于个人计划、招聘以及培养现有团队成员。

最后给出了关于度量价值的结论，如何处理那些很难量化的关键度量，以及如何在未来改善和扩展度量方法。

联系我们

有关本书的任何建议和疑问，可以通过下列方式与我们取得联系：

美国：

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

中国：

北京市西城区西直门南大街2号成铭大厦C座807室（100035）

奥莱利技术咨询（北京）有限公司

我们会在本书的网页中列出勘误表、示例和其他信息。可以通过http://oreilly.com/catalog/9781449305154访问该页面。

要评论或询问本书的技术问题，请发送邮件到：

bookquestions@oreilly. com

有关我们的书籍、会议、资源中心以及O'Reilly网络，可以访问我们的网站：

http://www.oreilly.com

http://www.oreilly.com.cn

在Facebook上联系我们：http://facebook.com/oreilly

在Twitter上联系我们：http://twitter.com/oreillymedia

在You Tube上联系我们：http://youtube.com/oreillymedia

致谢

本书中的观点受到了Michael Lewis关于赛博计量学以及运动统计分析的启发，这些分析指引我了解了Bill James的著作。这些观点体现了作者知识渊博、见多识广和幽默风趣的特点。尽管他们没有理由来读这本书，但我还是要首先感谢他们。

我还要感谢曾经在过去的许多年中一起工作过的卓越的程序员以及管理者。我非常幸运，很难想象从单一的职业生涯中我能学到如此多的东西。特别要感谢Vocalocity公司的CEO Wain Kellum以及整个团队，他们在我写作本书的过程中给予了我支持。

感谢Andy Oram, O'Reilly的编辑，他在整个过程中给予我帮助，使得这本书在各方面都更上一层楼。与你共事非常愉快，谢谢你，Andy。同样也感谢O'Reilly的编辑Mike Hendrickson，是他最早支持和鼓舞了我的想法。同样感谢整个O'Reilly媒体制作团队。我要感谢Google的Brian Jackson、Taleo的Nagaraj Nadendla、Zuora的Ben Wu给予的反馈和评审。他们自身都是卓越的领导者和管理者。谢谢他们。

感谢父亲，他培养了我对运动和统计的爱好。感谢母亲，她用爱鼓舞我写作本书。

最重要的是感谢我的妻子Barbara。她在写作本书期间大力支持我，不仅仅是用她出色的编辑技巧校对本书，发现缺陷并提出一些改进意见——虽然她是一名律师，并不会写一行代码（可能会）。谢谢她，亲爱的！同样感谢我两个漂亮的女儿，Naomi和Vivian，她们使得我的每一天都是特殊的。感谢她们！

第一部分 概念

该部分涵盖关于度量、模式分析、数据采集和数据元素的常见概念。

第 1 章 概述

让我们不要太确信，我们没有错过一些重要的东西。

——比尔詹姆斯（棒球统计学家和作者），摘自"Underestimating the Fog"

这是一本关于程序员、软件开发团队的度量和模式的书。本书的一些想法源于我在多年前开始的对软件开发团队构成的思考：无论好坏，所有细微贡献以及无名英雄的辛勤汗水都是项目成功的关键组成部分。近二十年里，我一直在负责设计师、程序员和测试团队的组建与管理工作。这些年，我意识到一个软件开发团队就像一支球队一样，需要有各种角色的球员和不同的技能的专业人员才能成功。我同样认识到成功和失败的模式未必像我之前所设想的那样简单。

我见过一个简单的模式，或许你也看到过：我曾经所在的每个成功的软件开发团队中，总是至少有一位同事无怨无悔地去做一些琐事，比如创建安装程序，改善编译脚本，或者修改一些其他人的错误来帮助团队实现产品功能。如果团队里没人去做这些琐碎的事情，那些项目就总是无法完成，或者至少是做得不够好。

另一种模式是：我见过很多经验丰富的软件开发团队，其中一般都有一到两位程序员在充当明确的技术领导和关键人物，虽然他们未必拥有与之对应的头衔。这些关键的程序员不仅解决问题，而且他们对其他人产生了强大的影响力，比如其他的程序员技能飞速发展，越来越接近技术领导者的水平。其最终结果就是，一到两个牛人提高了整个团队的水平。

这里还有我在曾经亲身参与的一个长期项目中观察到的一个模式，这种模式尤其常在处在创业阶段的小团队中发现：当项目进展到80%的时候，项目团队往往就“撞墙”了。像马拉松运动员跑到20英里标志点一样，项目团队经过几个月的努力奋斗，每一个人都身心俱疲。有时候当团队遇到困难时，我们就停滞下来，并且无法重获生机。这样项目剩下的20%工作量似乎永远也完不成，最后，我们基本上都是跌跌撞撞地走向终点。但有时某些团队可以穿越那堵墙，重新恢复生机，再次调整好步伐。在任何情况下，能够重获生机源于团队中一些人的优秀品格，他们能够减轻团队的工作负担，营造轻松的工作氛围，鼓舞团队士气，并让每一个人都感觉良好。感谢团队中那些爱开玩笑的伙计，他们让团队中的每一个人重新找回（多数是）积极的心态，准备冲刺到终点。

一旦我们看到这些，成功的模式似乎是显而易见的，但要看到它们，我们必须学会相应的方法。当我开始思考这个问题的时候，我就在琢磨我们是否可以建立一套指标，以便给我们一个明确、客观的方法来识别、分析并讨论软件开发团队的成败以及全方位地看待程序员的技能和贡献。这并非只是一种评判绩效的方法，而是一种有助于我们更好地理解和获得成功的关键因素，并且它指明了从哪里和如何提高。我在自己的团队中进行了一些尝试，并取得了优异成果。令人鼓舞的是，这些方法对其他人也同样适用。

本书是我对这些想法和实践进行分析的一次尝试。在这一方面，很少有关于软件开发团队度量的材料——无论是书面的或其他方式的。我们有关于面试、技能测试、项目估算、项目管理以及团队管理的大量书籍，还有关于敏捷和其他更有效提高开发流程的方法学之类的书。但是，我们从未有过讨论或探寻一种量化分析方法，该方法通过理解个体程序员的技能和工作来提高软件开发团队的效率。

目前绝大多数软件开发团队所使用的度量，一般是项目估算或项目管理过程中的一个简单的计数集合。我们使用bug数量、任务数、时间增量（时/天/周）以及敏捷团队中的故事点数（story point）和速率（velocity）来度量。项目估计中也有些更复杂的系统和工具，如使用千行代码量（KLOC）和功能点之类的数据进行规模度量。

但我们常用的度量标准没有提供足够的深度来回答我们所面对的很多关键问题，例如：

·我们的软件开发团队可以变得多么优秀？

·什么样的团队成员才能有助于团队的成功？

·哪种能力的提高有助于团队取得更大的成功？

如果我们不能很好地回答这些看似简单实则深刻的问题，或者缺乏一种清晰的方式来讨论和思考这些问题的答案，那么作为个人和团队成员，我们并未竭尽所能去取得成功。当然，我们必须从根本上探究成功究竟是什么，以及如何衡量软件开发团队的成功，而不是想当然地认为这些可以得到充分解决，而事实上这些问题还存在。在接下来的内容中，我将尝试建议一些全新的、与众不同的方式，来帮助我们更好地理解这些问题以及得到可能的答案。

我是个体育迷，因此在本书的很多地方，我选择用体育来作类比。然而，这并不意味着为了理解本书中的这些概念，你需要喜欢或者懂得体育运动。像所有的类比法一样，其目的只是帮助我们更好地领会及更容易记住一些概念。就个人而言，我认为采用体育类比来探讨软件开发团队是恰当且饶有乐趣的。

我把软件开发团队想象成一支球队。通常，软件产品是通过团队而不是单个人开发出来的，尽管有单个程序员独自完成工作的例子，但此时那位程序员一个人扮演了一个团队里的各种角色。我们知道，在体育比赛中，成功的球队需要球员之间能够互补，而不需要也不应该要求每个人具备同样的技能。球队里除了需要擅长跑动、传接球的球员，也同样需要擅长防守和抢断的球员。不是所有的人都擅长做同一件事。事实上，所有球员都拥有相同优势的一支球队，不管这个优势有多强，大多数时候都比不上拥有不同的优势互补技能的球员的球队。因此，只有球队中的每一个球员都做好自己的本职工作，球队才能取得成功。

通过使用统计分析来度量程序员的初步想法来自于体育活动中有组织的量化分析法。计算机和软件已经带来了职业球队队员统计数据分析的巨大变化，以及帮助他们确定队员的哪些技能可以最直接地帮助球队获胜。比尔詹姆斯和其他的记录分析家已经建立了一个围绕棒球运动员进行统计分析的学科，称为“赛博计量学”。通过作者Michael Levis的书《Moneyball》、《The Blind Side》和他在《纽约时代杂志》以及其他出版物上的文章，这些新的方法已普及到球队的管理中。

将这些新方法应用到球队管理中的先驱在数据分析领域都接受过较多的训练，比如Daryl Morey（NBA休斯顿火箭队总经理）在美国西北大学主修的是计算机科学，Paul DePodesta（MLB纽约大都会队副董事长，前洛杉矶道奇队总经理）在哈佛大学主修的是经济学。这个应用于体育中的新方法，相对于占大多数的、基于主观和直觉判断的人才评估和球队建设的方法，经常被看做一种应变和迁移。多数球队现在都属于很大的企业，拥有大量的资金。在这个新时代里，球队的管理者花费更多的时间来收集和分析度量数据，用更合理和可预测的方式来帮助打造获胜的球队（就像《Moneyball》描述的，用更有效的成本效益和盈利的方式）。度量并不是要代替个人的直觉和创造力，而是帮助我们增进了解。这个新方法中所遵循的关键步骤包括：

·发现测量获胜球队和失败球队差异的方法。

·发现测量单个球员对球队贡献大小的方法。

·确定那些决定球队胜负的关键球员的特征。

发现体育中有意义的度量指标和公式的过程不是一成不变的，而是一个持续演进的过程。很容易理解，许多重要而细微的技能难于测量和分析，比如防守型球员能发现带球球员的本能，或者在压力下进行比赛的能力。例如，比尔詹姆斯在关于棒球的连载文章和年鉴中所介绍的新的度量指标和思路，一些被人采纳和使用，一些被改进了，也有一些用处不大，逐渐消失了。

和公开的演进相同，度量指标也在悄悄地演进。体育运动是一个竞争性领域，因此球队实际采用的统计数据和公式是保守的机密。很多分析师在公开撰文的同时，也为单个球队充当私人顾问的工作。Theo

Epstein（MLB红袜队总经理）以及Billy Beane（MLB奥克兰运动家队总经理）可能会彼此分享一些信息，他们也都可从大社区中众所周知的度量指标中获益，但最后他们都在试图战胜对方，因此，属于他们方法中的某些元素，他们组织之外的人是无法知道的。

有别于大体育联盟的竞争压力，我们的软件开发领域不那么公开化，大多数程序员也不在公众的视线范围内。我们没有或者从来就不会有粉丝关注我们的统计数据，或者把我们的海报贴在他们家墙上（有点儿可怕的想法）。颇具讽刺的是，我们所在的这个领域，在许多方面使体育（以及其他行业）的深层的统计分析成为可能，但我们自身并没有拥抱或者完善地考虑量化分析在我们软件开发领域中潜在的益处。

像其他工作者一样，我们可能很自然地怀疑是否可以找到一个好的度量指标，是否存在一个真实有效的例子，并且也可能担心这些统计数据会被管理者错误地运用到绩效考核中等。然而，本书的前提是，在我们的领域中，有多种技能和结果是可以真正地测量的。从那里我们能够针对我们自己和团队获得有意义及有用的见解。这些数字不是非黑即白，并且依靠个别的几个数字无法说明全部。知道Derek Jeter的平均安打率（batting average）或者Tim Duncan的投篮命中率，只能告诉你他们作为一个高效的球员或者队友的一个很小的部分，但当我们看到多个统计，我们就可以识别个体和团队的模式，有时我们的发现甚至是意外的而且富有启发性的。

让我举例告诉你一个我曾管理了多年的软件开发团队的故事。

注意：关于本书中的一些故事的说明：这些故事来自我之前的工作经历，但在很多案例中，这些故事做了简化或概述，以传达要点。

为了保护个人隐私，我将不使用姓名，包括我本人。

这个示例发生在一个风险投资创业公司，这个团队有6位程序员和3位测试人员（本书重点关注程序员，因此在这个例子中，我将着重描述他们）。我们前两年的经历中有三个关键阶段：1.0发布版的最初开发工作，大概花了9个月的时间；1.0版本发行之后，我们花了6个月的时间支持第一个客户和开发1.1版本；接下来又花了大概9个月的时间来开发2.0发布版。这个团队拥有3位资深程序员，每个人都拥有超过10年的开发经验和优秀的领域知识，还有3位初级程序员，拥有很好的教育背景及大约两年的商业软件开发经验。在这两年中，所有的资深程序员依然留在团队中，但其中两位初级程序员在完成第一年的工作之后就离开了这个团队，之后我们又招聘了两位新的程序员。

我们的执行委员会和投资者认为我们最初的1.0版本取得了巨大的成功。我们在一个关键的行业展览中赢得了大奖，并且收到了许多正面的产品评价。许多中间商对我们感兴趣，客户评估的数量两倍于我们的预期，以至于我们的销售人员忙得不可开交。该预制（on-premise）的软件解决方案运行在客户的环境中。在产品发行后的第一个季度，收入也同样好于预期。

这有足够的理由使得我们的软件开发团队感觉良好，每一个人都在夸奖我们。但是，我们的1.0版本真的成功吗？

我们花了一些时间才意识到这个问题，通过探查当时的数据，应该能够发现一些严重的问题。关键而糟糕的事实是：当我们成功地获得了知名度，强化了客户兴趣的时候，每个试用的客户平均要来7个电话寻求客户支持。尽管每个客户事实上都收到了安装程序和安装帮助。这7个电话使得需要平均3整天与客户一起工作来调查问题，而且结果证明每个客户平均在产品中发现了从前不知道的3个新bug。花在支持客户试用（包括辅助支持的时间和修改重大产品问题）的程序员时间是以周为单位来计算的，而不是以小时或者天为单位的。

那些看似积极的收益也在误导着团队。由于几单大的生意，我们超出了早期的收入计划，但是我们从评估者到真实客户的整体转换率及转换所花的时间远不能满足达成一个成功业务的要求。这种情况至少部分是因为从支持工作量和发现bug的数量上反映出的可用性和质量问题。

换句话说，虽然局外人可能认为我们的初始发布版取得了巨大成功，但是事实上它充其量不过是部分成功。图1-1里的数据揭示了新用户与bug和支持问题相比是多么微不足道。

图 1-1 1.0发布版的关键指标所揭示的关键问题一览

还有另外一个大问题。随着时间的流逝，团队里的一些程序员遇到了麻烦。花在刺激的新功能上的时间越来越少，花在乏味的问题调查和bug修改上的时间越来越多，加上初创阶段紧张的支持工作，成员间和团队内开始显露出裂痕。个性差异被放大，某些程序员慢慢地开始彼此回避，甚至在工作场所大喊大叫也常有发生。

在1.0版本发布后的6个月，即在团队提供支持和开发1.1发布版的这段时间当中，团队内充满了混乱，甚至是灾难，尽管团队之外的人依然觉得一切都很好。每个程序员的大部分时间都花在了bug的修改上，我们不得不延迟大多数的产品增量改进。1.1版本修复了所有严重的bug，依然还有很多问题留到了发布之后，支持工作的负荷和转换率并没有实质性改变。

接着，突然有一天，团队里所有的事情都变得好了起来。尽管客户支持的工作比率依然还是那样，团队却开始更有效地处理软件中的问题。每个软件问题涉及了更少的人，更多的时间被解放出来，用于新功能的开发和问题集中领域的重大改进。1.1发布版几乎没有任何功能的提高，却花了6个月的时间。2.0发布版包含了许多新功能和产品的重大改进，相同规模的团队仅花了9个月的时间。随着2.0版本的发行，转换率和软件问题率有了显著的改善，基于这一点，我们可以清楚地说2.0发布版本取得了更大的成功。

到底发生了什么？是不是所有的人都习惯了解决问题，或者软件问题开始重复或者不太严重了？从某种程度上说，确实如此。但关键的变化是两位初级程序员的离职和两位新的初级程序员的加入。

两位离职的程序员是基于自己的决定离开的。尽管在1.0版本的开发工作中他们很高兴，但他们并不喜欢版本发布后的大多数支持工作。如果他们遇到不清楚的问题或代码，他们总是习惯于寻求其他人，特别是资深程序员的帮助。随着时间的流逝，其中一个人脾气越来越大，并且变得好斗起来。

新加入团队的程序员在教育背景、工作经验或者才能方面，与离开的那些人并没有明显的不同。不同的地方在于，在第一个产品发行之后，有两个关键技能变得非常重要和有用：强烈的愿望和乐意独立解决问题，以及冷静地甚至是开心地处理紧急状况的能力。图1-2展示了一个替补者是如何比他的前任做得更好的。

图 1-2 前任（程序员A）与替补者（程序员B）的比较展示了团队成功的一个关键因素

因为新的程序员拥有合适的技能，所以他们能够独自承担和解决更多的问题。并不一定是我们花了更少的时间在客户支持和修改具体问题上，但我们可以让更少的人涉足其中从而使其工作更少被打断。这样，其他的团队成员就可以把精力集中在其他的工作上。最后，我们时来运转。因为我们曾经与离职的两位程序员有一些个性上的冲突，所以我们有意识地偏好和选择那些不同个性的应聘者。但我们并没有意识到，这给我们整体的生产力和团队成功带了多么大的好处。

在这些事情发生的时候，我们并没有密切关注我们的度量指标。回头看，我认识到，如何关注团队的关键度量指标，能够帮助我们在第一个产品版本之后更快速和更有效地做出反应。当人们正在为好的事情接受局外人的祝贺的时候，很难让每个人相信存在问题或者认识到问题的重要性。让团队滋生自满的情绪很容易，或者相反，在没有得到应得的赞赏时士气低落也很容易。关注产品开发的全过程，团队的度量可以平衡你所得到的奉承或者批评，并且围绕你的真实状态和所要做的事情提供一个新的视角。围绕着自力更生和善始善终的测量和讨论，能够帮助我们培养这些技能，并且保证拥有这些技能的程序员因为他们对团队作出的贡献而获得他们应得的信任和认可。

本书旨在介绍一些方法和度量指标集（也就是程序员度量），它们覆盖与个人开发者以及软件开发团队

相关的多个领域。这些方法旨在挑战我们的假设，希望借此我们能够更好地发现通向成功的可能模式中哪些是可行的。为了让它们更易理解和记忆，本书介绍的度量指标遵循了体育中类似的统计指标的命名。这些度量指标意在提供一些术语，以便于更好地沟通，也希望我们能觉得，这些度量在我们的软件开发领域是有用的。最后，可通过它们在多大程度上帮助我们回答了那些关键问题来衡量其价值，诸如我们面对的关于“胜利”意味着什么以及怎样可以改善自己和团队的关键问题来衡量。

我希望本书中的概念能够帮助程序员、团队组长和管理者之间形成更有成效的对话（组织内的或组织间的）。毫无疑问，这里介绍的许多个人度量指标可以改进，或者将会得到改进；其中的一些想法也可能会放弃，也可能会有更好的度量有待发现。就我而言，我已经认识到以下行为的巨大价值：在团队中定义多样的度量指标，识别如何度量个人与团队的活动度量并将它们连接到组织目标，然后在团队内部对这些数据进行分享与讨论。即使你可能不太乐意使用度量指标，我也希望你可以从中找到有价值的东西，并且希望本书里的一些想法能够积极地影响你思考一些关于程序员和软件开发团队的问题。如果有人开始考虑这些概念，并且或许能使用本书里列出的一些方法来对程序员贡献和软件开发团队构建进行更宽、更深的理性分析，我就觉得本书成功了。

必须注意，软件开发流程中的很多参与者和技能并不在本书的讨论范围之内。本书仅涵盖了一部分，这是因为在一本书里很难涵盖全部的参与者和技能，更是因为我本人并没有为其他技能定义相应的度量指标。也许在将来，我们可以为设计师、测试人员、管理者或其他角色开发出度量指标，或许也会有关于这些方面的著作。

第 2 章 测量程序员的工作

不要混淆活动和成就。

——John Wooden，1946~1975年间担任UCLA男篮教练

度量关心哪些方面？它们有什么样的用途？它们怎样才能运用于程序员和软件开发团队中，它们又如何在其他领域使用？

本章将开始探讨度量的一般用途，以及和某些度量相关的有用特性及其他非相关特性。我将会讨论不同的模式，以及度量能帮助你识别和理解的一些显著信息。我也会关注那些我们可能使用的各种各样的数据类型，以及那些用来确保数据精确和一致性的数据收集方法。这里提及的一些概念为度量中的关键概念提供了一个基本的介绍，并作为后续章节讨论的基础。

度量的目的

收集和使用度量有3个目的。当然，目的可以更多，但我在本书中只关注这3个。

度量的第一个目的是帮助你跟踪和理解发生了什么。尽管对事件的主观观察有时是非常具有洞察力的，但也经常带有个人偏见和经验色彩。在观察时，人们常被自己注意的细节和习惯的方式所左右，而且易于错过一些自己未曾注意到或识别的事情。

例如，如果你去看了一场棒球比赛，之后有人问你还记得什么，你可能会描述比赛中非常精彩的部分。可能是一个本垒打，或一次激动人心的防守。但你会忘掉更多的细节——即使这些细节就发生在几个小时以前。有些只是你不记得了，有些也许是没有注意到，也有些你根本就没有看到——因为当时你可能正在卖热狗的小贩那儿。同样，你能记得多少以及你能描述什么，取决于你有多熟悉棒球、你之前观看过多少次比赛，也取决于你对这场比赛的各个方面有多了解。

另外，如果你看到比赛关键统计中的得分统计，不论你有没有去看比赛，你都可以了解到比赛中发生的许多事情。并且，如果你看到那些完整的统计数据分解，包含完整的进攻及防守统计和所有得分方面的细节，而且如果你也知道那些统计数据的意思，那你就能够看出关于那场比赛的大量信息、球员的贡献，以及那些决定胜负的关键因素。

统计，或者度量，是过去发生的事情的详细档案。对于球员或者球队所做的事情，以及为什么球队能够取得成功或者失败，他们提供了一个更科学的、实证分析的历史记录。度量也保留了历史。日月如梭，时间不可避免地增强或者模糊了你所记得的以及你觉得重要的事情。拥有越多的统计数据，你就越不容易忘记或者曲解过去。例如，当我还很小的时候，我爸爸带我去看过UCLA的很多次篮球比赛。我记得Brad Hollan d是我在20世纪70年代后期喜爱的球员之一，他是个很棒的得分手，但我无法记得具体的细节。然而，如果我在书中或网站上看球员统计，很多细节都会浮现出来。去年发生过的事情同样会忘记，就像忘记30年前发生的事情一样。拥有的统计记录允许我们回顾以及在某种意义上再次体验曾经发生过的事情，并且平衡一下我们选择性的记忆。

度量的第二个目的是帮助人们沟通发生的事情。度量本身成为术语的一部分，允许一组人在讨论一些情境的时候，对大家讨论的同一件事有一定的信心。定义和命名度量迫使你澄清你用于沟通的语言。没有这样的定义和清晰的术语，在沟通的时候，你会更容易进入误区，或者甚至都不能够很好地讨论那些事实上非常要紧的问题。

例如，在棒球比赛中，众所周知的投手投球的统计是投手责任失分率（Earned Run Average, ERA）。“投手责任失分”指的是在对方的得分当中因为投手的投球所造成的部分，ERA又表示一个投手在每9局投球中（意味着一个完整的比赛）的平均责任失分数。对理解和描述而言这意味着很多东西。但是，如果你说“那位投手的ERA是4.29”，对于熟悉棒球的人来说，你就可以快速简洁地传达丰富的信息。

度量的第三个目的是帮助人们关注那些他们真正需要改善的事情。度量记录了你所做和所完成的事情，并且给出了一个关于你的期望水平和实际水平的比较。缺少参考点将使得你很难知道目前所处的水平，到底还有多长的路要走，是否已经实现了目标。

美式橄榄球运动员测量他们在场上和场下的表现。他们测量在比赛中的推进码数、达阵得分和拦截抢球。但同时他们也测量40码冲刺次数和225磅杠铃卧推重复的次数，而这些事实上都不是球赛的一部分。他们之所以这样做，是因为他们知道速度和力量是赢得比赛的重要因素。他们同样拥有多年的数据来显示对不同位置（如角卫、线卫、跑卫和前锋）对速度和力量的要求范围。记录测度来显示球员目前所处的水平，可以帮助球员关注他们最需要提高的部分。

度量不是评级

在初中、高中和大学，你会得到对你学习的评级。评级应该反映了你对一门课程的掌握程度，和你相对于班级中其他人进行的一个排名。他们可能与你的客观测验的成绩紧密关联，虽然很多时候测验或者评级的其他元素包括了老师的主观评价。在早期，评级给你提供了反馈以帮助你确定哪些方面需要提高，但是，在后期，评级渐渐地演变成了夹杂着排名及奖励的一种竞争。由于在制度上和传统上的一些原因，不论好坏，学校不仅负责教育学生，而且负责给学生进行排名。

如果你打算接受度量，建立并且理解度量不是评级这一概念非常重要。度量测量特定个体的技能和贡献，并且像我在本书中将检验的，一般没有固定的“好”或“坏”分界线。成功的团队由不同的个体组成，这些个体之间可能并且也确实存在着非常巨大的个人度量差异。就像一个橄榄球球队，前锋比跑卫更慢；好的角卫常常只有较少的拦截抢断或突破防守，因为四分卫很少朝他们传球。

度量的目的就如同前面讨论的一样，为我们提供了一幅更清晰的画面来揭示发生了什么，帮助我们更好地沟通，并且帮助我们确定和发展有用的技能。本书后续章节将探究度量如何应用于招聘、绩效考核和其他团队管理的方方面面。然而，在所有的案例中，只有在度量作为每个程序员当前的工作技能、强项和弱项的反映，而不是作为一种评级方法的前提下，度量才真正有效。

团队动态

虽然不应该严格地认为度量是评级，但不可避免的事实是，任何好的度量系统都无法避免地识别出人与人之间的必然差异。在组织中，如果人们通过劳动获得报酬，并且他们的工资标准与他们的贡献和技能密切相关，那么好的度量数据与工资标准和其他额外津贴就存在着一些关联。但无论如何，你不应该因