华章程序员书库

PEARSON

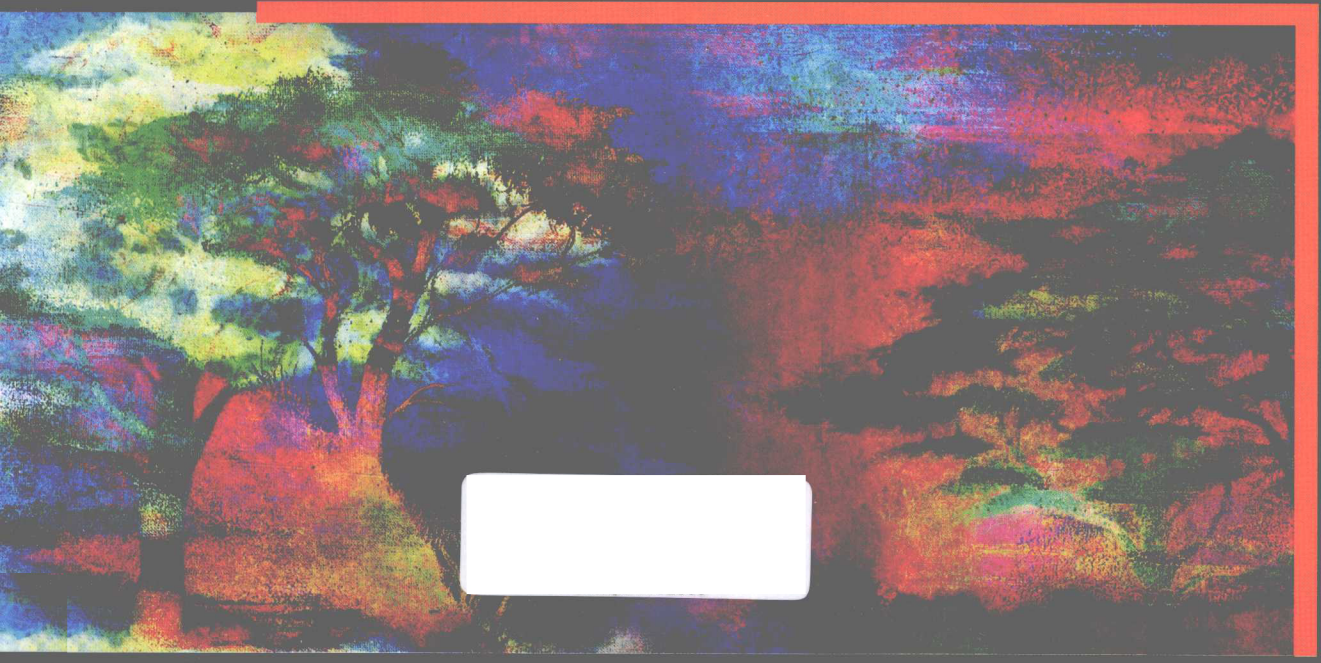
HZ BOOKS
华章IT

软件工艺师

专业、务实、自豪

The Software Craftsman

Professionalism, Pragmatism, Pride



[英] 桑德罗·曼卡索（Sandro Mancuso） 著

爱飞翔 译

机械工业出版社
China Machine Press

华章程序员书库

The Software Craftsman

Professionalism, Pragmatism, Pride

软件工艺师

专业、务实、自豪

[英] 桑德罗·曼卡索（Sandro Mancuso） 著

爱飞翔 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

软件工艺师：专业、务实、自豪 /（英）曼卡索（Mancuso, S.）著；爱飞翔译 . —北京：机械工业出版社，2015.8
（华章程序员书库）
书名原文：The Software Craftsman: Professionalism, Pragmatism, Pride

ISBN 978-7-111-51400-8

I. 软… II. ①曼… ②爱… III. 软件开发 IV. TP311.52

中国版本图书馆 CIP 数据核字（2015）第 207796 号

本书版权登记号：图字：01-2015-1916

软件工艺师：专业、务实、自豪

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：关　敏　　责任校对：董纪丽
印　　刷：三河市宏图印务有限公司　　版　　次：2015 年 9 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：14.25
书　　号：ISBN 978-7-111-51400-8　　定　　价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

The Translator's Words 译 者 序

这是一本谈论生活方式与做事态度的书。有一些开发者可能自幼喜欢技术，也可能是在工作中培养了对技术的兴趣，他们喜欢读书、写文章、倾听新见解、尝试新工具，并用代码来表达创意。如果你也属于这类人，那不妨翻开这本书，读读作者的种种经历和想法，看能否与自己产生共鸣。

作者把这些极具专业精神的开发者称为 software craftsman。无论是把这个词叫作软件工艺师、软件工匠还是软件匠人，它都蕴含着专业、务实而自豪的态度。

在这本书中，作者从热衷技术与认真工作这两个方面切入，系统地描述了软件工艺理念，完整地展示了软件工艺师是如何将兴趣与工作有机结合起来的。由于工作中所开发的软件要面向客户，要为商业目标而服务，所以开发者不能只凭对技术的热情埋头写代码，而是要协助业务人员准确把握客户的需求，并向客户提供有价值的软件产品。作者在书中反复强调这一点，就是要展现软件工艺的现实意义，说明它不仅是一种口号，更是一种能够使开发者、公司和客户都受益的多赢理念。

围绕软件工艺理念，作者详细讲述了几组话题。首先，作者讲述了软件开发者所应具备的工作和学习态度，以及如何交付软件产品、如何选择软件开发方式、如何规划职业发展道路等内容。然后，作者根据自己的工作经历，讲述了公司应该怎样吸引并招募这种对技术热衷、对工作负责的人，其中谈论面试禁忌的那一章给出了很多中肯的建议，值得大家参考。接下来，作者讲述了怎样在公司内部打造学习氛围，以及如何有效地推动技术变革。如果软件开发者自身和软件开发公司都能抱持这种尊重专业技能、用心服

务客户的理念，那么整个软件开发行业的面貌就会有所改观，我们会逐渐抛弃那种把开发者当成码农、把制作软件当成应付差事的消极态度，而这，也正是软件工艺致力解决的主要问题。最后两章及附录列出了在推行软件工艺及规划职业发展时的注意事项。

整本书里出现频率较高的两个词是热情和务实，而这两个词正好概括了作者所提倡的生活方式及做事态度。需要说明的是，由于作者是根据自身经历及业界转型趋势来阐述软件工艺理念的，所以在这个过程中，自然会多次谈到该理念对敏捷软件开发流程所起的补充作用，以及它对极限编程的运用。软件工艺、敏捷开发和极限编程，是一组相当理想的组合，大家可以参照《敏捷软件开发（原书第 2 版）》[㊀]和《解析极限编程——拥抱变化（原书第 2 版）》[㊁]等书，来了解它们的好处，然后根据周边状况自由选择合适的开发流程和开发方式灵活运用，而不必固守各种教条。另外，作者多次提到了测试驱动开发和代码质量，虽然这两部分远远不能涵盖软件工艺的全部内容，但它们却是把握该理念的绝佳切入点，大家可以参考《测试驱动开发：实战与模式解析》[㊂]《编写可读代码的艺术》[㊃]以及《代码整洁之道》等书，详加研究。

祝愿大家都能将技术融入生活，都能以开发软件为傲。

翻译本书的过程中，我得到了机械工业出版社诸位编辑和工作人员的帮助，在此深表谢意。

由于译者水平有限，错误与疏漏之处，请大家发邮件至 eastarstormlee@gmail.com，或访问 github.com/jeffreybaoshenlee/zh-translation-errata-craftsman/issues，给我以批评和指教。

爱飞翔

2015 年 7 月

㊀ 该书由机械工业出版社引进并出版，ISBN：978-7-111-23166-0。——编辑注

㊁ 该书由机械工业出版社引进并出版，ISBN：978-7-111-35795-7。——编辑注

㊂ 该书由机械工业出版社引进并出版，ISBN：978-7-111-42386-7。——编辑注

㊃ 该书由机械工业出版社引进并出版，ISBN：978-7-111-38544-8。——编辑注

Foreword 序

1973 年，Roberta Flack 唱了一首歌，名叫《Killing Me Softly》[⊖]。大家应该在电梯里或奶奶家的收音机里听过吧？这首歌柔美而深情，讲的是一个女孩去听演唱会，台上男歌手唱的歌深深打动了她，她感觉那首歌仿佛唱的就是她，她怀疑歌词就是自己写过的文字。她甚至觉得歌手在弹吉它时，好似拨动着她自己痛楚的心弦，并唱出了她的整个人生。

面前的这本书对我来说也是这种感觉。作者的职业生涯和我完全不同。他比我稍微年轻一点。我们两人居住在不同的洲，从事着不同的工作，文化背景不同，国籍不同，种族也不同。我只见过他几次，每次不过几分钟。简言之，两人唯一的共同点仅在于我们都是程序员。但这就够了。

书中有两条精彩的主线互相交替，一条是作者自述的轶事，记录了他自己丰富的经历，另一条则是根据这些经历写成的权威建议。你要也是位程序员的话，一定会对这些故事和心得感同身受。你看书时也会像我一样在心里说："嗯，没错。真是这样。"而且，你会觉得作者的"歌声"一直萦绕耳畔，诉说着自己的辛酸。

我这么说可是认真的，因为这就是一本谈经验教训的书。书中所讲的问题每个程序员都会碰到。这个问题就是：开发者会经常觉得自己没办法把工作做好，也就是陷入一种不够专业的感觉之中。开发者想做得更好，但又不知应该如何去做。

⊖ Roberta Flack 是一位美国女歌手，生于 20 世纪 30 年代。《Killing Me Softly》又名《Killing Me Softly with His Song》，歌名大意是：他温柔的歌声令我迷醉。——译者注

这本书不仅指出了问题，还给出了改进措施。实际上，我认为它给出的是指导思想。整本书都在谈软件行业的专业水平。这不仅指的是程序员的专业水平，还包括整个软件开发组织的专业水平。因此可以说，这是一本谈论软件工艺（Software Craftsmanship）的书。

作者在书中提出计划和策略，并指明做事态度以及一系列指导原则，程序员、开发团队及软件组织都可以由此摆脱平庸、走向专业，使工作变得更有效率，并对工作倍感自豪。

这本书包括的范围很广。从设计模式、结对编程及测试驱动开发开始，谈到如何安排面试、如何评价面试效果，再谈到怎样应对紧迫的工期、怎样撰写职位描述信息，以及怎样与同事和管理层相处。

总之，对于正在努力提升专业水平并坚持软件工艺原则的组织来说，这是一本百科全书，它包含了此类组织所追求的行为、特征及架构。

身为一位程序员、团队主管或项目经理，你是不是那种想在忙碌了一天之后赶回家，对着镜子说声“今天真棒！”的人呢？是的话，就看这本书吧。

——Robert C. Martin

Preface 前　言

那是20世纪90年代中期，我的职业生涯刚刚开始两年，巴西圣保罗有家大型国际公司宣布要一次招纳60名开发者。选拔过程分四个阶段，共需数周时间。第一阶段是三小时技术测试；第二阶段是两周的公司专有技术培训，培训结束后考试；第三阶段是一整天团队互动；第四阶段是最终一轮面试。该公司在一家大报纸上刊登了这一消息，大约有900名开发者应聘。那时我正在一家小软件公司上班，工作非常开心，但我觉得自己已经准备好干一番大事。因为第一阶段安排在周六，所以我决定去试试。不到300名开发者进入第二阶段，我也在其中。我信心满满但略带忧虑。接下来的选拔过程需要很多天，如果我要继续参加剩下的三个阶段，那就必须辞掉现在的工作。当时我经济条件并不好，而且也无法指望家人资助。为了追求理想的工作而放弃现有的工作，是非常艰难的，何况那份工作还未必能够被录用。而且一旦找不到新工作，我也不知道怎么维持生计。然而我还是辞职了。我必须去试一试——我就是想去那样的公司上班，我就是想从事那份工作。

那年我21岁，虽说很年轻，但其实已经有好些年编程经验了。我11岁开始写代码，19岁开始上班。问题在于，像这种年轻而又稍微有点经验的人很容易骄傲。我自然也不例外。我就是一个傲气的年轻开发者。我觉得自己比谁都强。我当时认为自己特别优秀，比学校里的同学强，也比从前共事的大部分同事厉害。

四个阶段都结束后，公司宣布没有招满60名符合要求的员工。他们只招了32个人，我就是其中之一。我当时真是兴高采烈、志得意满。公司的系统中有多个业务模

块，第一周上班，我分到了负责交付某个业务模块的开发团队里。头几周时间，我在和负责其他业务模块的开发者聊天时，听他们提到了一个团队，据说那是公司里最好的团队。那就是所谓的“架构”（architecture）团队，他们负责系统核心，并且要提供所有的基础代码（infrastructure code）给各业务团队使用。

架构团队的主管是个了不起的家伙，除了做管理工作之外，他还是个出色的开发者。他是个大忙人，但总能挤出时间去写代码、以自己的身份提交代码，并审校本团队其他成员的代码。我听说那个团队一直在做些有趣的事，而且代码写得超级棒。那正是我想去的地方。我想和最优秀的人在一起。

漫长的几周过去了，我决定和架构团队的经理谈谈，他就是我刚说的那个人。我既不知道该谈些什么，也不知道结果会怎样。我能确信的事只有一件：我没什么可失去的。最坏的结果就是他说他不想把我放到他们团队里。某天我看他一个人在咖啡间，于是哆哆嗦嗦地走了过去。我向他介绍自己：“你好，我是 Sandro。”他对我微笑，跟我握手，并且平静而从容地说道：“我是 Namur。很高兴认识你。”几秒钟尴尬过后，我紧张地说了一句：“我想加入你们团队。”他稍微有点惊讶，但看上去很高兴。我开始和他聊应聘的过程，聊自己为什么要来这家公司，以及公司对我的要求等。他也问我有没有参加业余兴趣项目，问我对什么技术感兴趣，以及是否在工作时间之外写代码，然后还说了其他一些事，我记不得了。聊了大概 30 分钟之后，他就问我什么时候可以开始过来上班。我愣住了。我根本没想到他这么快就会答应。我以为还要安排一次面谈，再来一场正规面试什么的。后来我才意识到，其实整个谈话过程中，他都在判断我对软件开发的喜爱程度。他在分析我是不是那种认真负责的人。他并不担心我目前的技术知识水平。我对 Namur 说：“我去和现在的经理谈谈，希望能快点过来。”几周之后，我就和新队友坐在一起了。

第一天上班是个周一。Namur 早上跟我说，要布置一项任务给我。他描述了应用程序的某个部分，以及我需要完成的事项，然后说周五会过来看看我的成果。我太激动了。这正是展示自己能力的好机会！我要叫他看看我自己在团队中的价值。于是，我一直到深夜才离开办公室，只睡了几小时，周二一大早就爬起来上班，到周二下午两点就把任务完成了。只花不到一半时间就做完自己的第一份任务，这感觉妙极了。我一向都认为自己很棒，但这次，在这样一个高水平的团队里，面对这样一份从未参与过的代码

库，我能把任务完成，那真是了不起！

我跑到 Namur 办公室，兴奋地告诉他："做好了。程序已经写完，而且能够运行。"他当时正在敲键盘，看到我之后停了下来，平静地说："写出的程序能运行，只是我对员工最起码的要求。你告诉我某个程序已经写完，那本身就意味着它要能正常运行才对。"这话给我泼了盆冷水。我脸上的笑意稍稍收敛了一下，不过我认为这也许只是他的说话习惯而已。可能他今天不太顺吧。我想他绝不至于故意讲难听话。"咱们坐下来看看你写的代码吧。"他继续说着。我坐下看他从源码控制系统里获取一份 .pas 文件，我写好的所有代码都在这份文件里。他通过命令行，用一种黑绿相间的神奇编辑器打开了这份文件。那是我头一次看到 vi。当时我们一般都是用 Delphi 来开发的，Delphi 是一种非常流行的集成开发环境（Integrated Development Environment，IDE），功能特别强大。我看到有人用 vi 打开 Delphi 文件，觉得特别怪异（vi 是一种 UNIX 文本编辑器）。"坐过来一点，咱们一起看代码。"他说。我写的代码大概有两百行。他把光标移到第一行，然后一行一行往下看。每五六行，他就停下来，跟我说下面这些话："你知道分配内存和释放内存的时候会发生什么吗？看看这里，你在一个方法里面分配内存，但却在另一个方法里面释放。这可能会导致内存泄漏。听说过临时耦合（temporal coupling）[⊖]吗？看看这块代码，你仔细想想就会发现，可以把它从 8 行精简到 2 行。你知道在 `try/catch` 块中放这么多语句会出现什么问题吗？这个变量和方法的名字起得合适不合适？它们表示什么意思？你是否想过，有些同事可能需要修改这段代码，但他们又不像你那样，拥有足够的信息和明确的语境，他们怎么理解这些名称呢？他们在不了解这段代码的情况下，又该如何维护呢？这个地方怎么会出现硬编码？你有没有想过，现在把固定值写在这里，将来万一要修改，那我们又要打开代码、修改代码、编译代码，而且要重新部署整个应用程序？文件里为什么老是重复出现这几行代码？噢，还有这里，这个方法怎么这么长呀！要是每个方法都写这么复杂的话，我们看代码的时候还能不能理解它了？应该使它们短小精悍，而且要按照功能起个合适的名字。"他就这样一直说下去……

然后，他在一个地方停了下来，开始盯着那几行代码。他在那上面花了几分钟，偶尔把光标移到前一屏或后一屏看看。20 世纪 90 年代那阵，如果某个开发者能写出别人看不懂的代码，那大家就会觉得这个人很强，他们会说："这个人肯定特别厉害，写出

⊖ 在程序设计中也称时间耦合、暂时耦合或时空耦合，指两个操作仅因为同时发就封装到了同一模块中的耦合现象。——译者注

来的代码我们都看不懂喔。”我在那份源文件里也写了一些晦涩难懂的代码，用来炫耀自己很聪明。等 Namur 看明白那些代码的时候，我在想，他最后总该鼓励我一下吧？但是他却冷冰冰地说：“你知不知道这么写代码很不好？我们在做一个非常大的系统，有很多团队和开发者都要在同一份代码库上面工作。要是每个人都这么炫耀的话，这代码理解起来该有多困难？不要说有几百万行代码，就算只有几千行，这样写也不行。”第二盆冷水又泼了下来。

我写的代码只有两百行，但针对这两百行代码，我却没办法回答他的任何问题，他指出那些缺点时，我也想不到该怎么答复才好。他就这样一行一行看着代码，批评我的写法，然后跟我说应该怎样写才会更好。等他看完整份文件之后，我已经十分羞愧苦恼了，但他依然淡淡地对我说：“你明白我刚说的那些了吗？你觉得我的建议对不对？”他说话时那种口气，就好像写代码的这个人不是我，而是另一个不在场的陌生人一样。我什么都没说，只是点点头。他又问：“你现在觉得自己能把这份代码写得更好一些吗？”我没看他，只是点点头。他还接着问：“你觉得自己以后写代码的时候能不能做到我们刚说的那些？”我还是只点了点头。于是，他按了几个键，把我写好的整份代码文件全删了，然后说：“嗯，那就好。还有三天时间呢，重写吧。”

我又愣住了。我不知该怎么回答。站定之后，我缓步走向门口，一句话也没说。“ Sandro，”快走到门口时他叫住我，“方式与结果一样重要。”说完之后，他又回过身，对着那个怪异的绿色编辑器开始打字。

我很沮丧。实际上，我特别愤怒。离开他办公室后，我直接下楼，走出了公司。当时我心想：他以为他是谁呀，居然用这种口气跟我说话，混蛋！我才不给这种人干活呢。真是受够了！我不想留在这家公司了！辞职！抽了几根烟后，我冷静了一点，开始回想刚才的事。Namur 毕竟花了一个多小时来看我的代码，而且还跟我解释了怎样才能写得更好。在我偶尔表达自己观点的时候，他也能倾听，并且会认真地指出我的错误，或者告诉我还有另一些更好的写法。我发现，自打开始编程，这是头一次有人肯花工夫告诉我怎么样才能写出好的代码。另外，他水平确实比我高很多，而且经验也远比我丰富，他是真心希望别人能把代码写好的那种人。我想，我发现了一个追求优秀软件和高质量代码的人。这样一个人会花时间来指导我，而且，更重要的是，我找到了自己的第一位导师。

几根烟过后，我重新振作起来，跟变了个人似的。那天我终于明白自己并不如想象中那般优秀；我明白了应该如何保持谦虚的态度；明白了我还有很多东西要学；明白了只把事情做完是不够的，尤其是身处团队之中；明白了怎样做才能受到同事与客户尊重，而不是留下一堆烂代码不管；更明白了一名真正优秀的专业人士一定会在乎自己的作品。

我和导师 Namur 及其他一些优秀开发者共事了两年半的时间。这段经历不仅使我变得更加专业，而且使我变得更会做人。虽然当时并没有提及“软件工艺”这个词，但十年之后我发现，我第一次接触这个概念正是在那个时候。我从他们身上学到了很多。从技术角度来看，那是一段有益的经历，但最重要的并不是技术本身，而是我在那段时间里从主管和同事那里学到的对待工作的态度。第一次审校完代码时，Namur 对我说的那句话彻底改变了我。十年后，我成立了伦敦软件工艺社团（London Software Craftsmanship Community，LSCC，网址是：http://www.meetup.com/london-software-craftsmanship），而 Namur 的那句话——“方式与结果一样重要”（how it is done is as important as getting it done），成了我建立网站时第一批写上去的文字。后来我给 LSCC 定制了一些 T 恤衫，这句话也印在上面。这句话不仅使我成为一名更优秀的开发者，而且也把我变成了一个更好的人。

关于本书

数十年间，众多软件开发方式（methodology）涌现，可是软件项目依然会失败。失败的原因很多，但有几个问题不容忽视：管理者只把软件开发当成一条生产线；公司不知道怎么管理软件项目，不知道怎么招聘优秀开发者；很多开发者仍然不够专业、不够积极，他们向雇主和客户提供非常糟糕的服务。有了敏捷软件开发（Agile）[⊖]这种方式之后，软件行业确实出现了巨大进步，但软件项目的失败率还是很高。这些项目为什么依旧做不好呢？为什么无法顺利完成软件项目呢？我们到底缺少什么？

尽管“软件工艺”这个词十几年前就出现了，但直到最近它才成为一种可靠的解决方案，能够用来应对软件行业时下面临的许多问题。

软件工艺为开发者和软件公司提供了一套独特的理念。尽管它并不是一种软件开发

⊖ Agile Software Development（敏捷软件开发）可以简称为 Agile。——译者注

方式，但却强烈建议我们采用某些技术实践手段及指导原则，这些手段和原则基本上都在极限编程（Extreme Programming）中做了界定。通过与敏捷（Agile）和精益（Lean）原则紧密结合，软件工艺可以大幅提升软件行业的水平。它关注的重点是专业精神、高超技术，以及良好的客户满意度。而其中一个重要方面就是转变态度：不要再把软件开发者看成生产线上的工人，也不要再把管理软件项目当成开工厂。

如何才能变成更好的开发者？如何才能交付更好的软件项目？本书将会讲述真实案例，给开发者和软件公司提出实用的建议，直接参与软件项目的每位开发者和专业人士都适合阅读本书。

Acknowledgements 致　谢

我的职业生涯是一段精彩旅程，从巴西小镇到欧洲大城。我在大西洋两岸的数家公司就职过，遇到了很多高手，他们以各种方式帮助我，使我更会做人，也更会工作。

首先我要感谢 Maria Cecilia Capelache 教授。当我告诉她，自己可能要因为经济问题而辍学时，她立刻给我提供了一份工作，使我可以继续完成学业。直到今天，我依然觉得，她给我那份工作主要是为了帮我，而不是那个团队当时真的需要多加一名开发者。她告诉我学校里教的那些知识和现实中的软件开发有何不同。假如我当时从大学退学，今天可能就不会写出这本书了。

然后我遇到了 Luiz Fernando Ferreira，他是那家小软件公司的一名雇主，他给了我第二份工作。虽说我是他们的首位雇员，但公司并没有把我只当成员工对待。第一天上班，我就感觉自己在和一位老友共事。今天我俩依然是朋友，无论身处顺境还是逆境，他都非常坦诚，在我需要提升自己的时候，他也给了我莫大帮助，这些我永远心存感激。

去跨国大企业上班，并得到良师指导，这改变了我的职业生涯。真的很感谢 Eduardo Namur，他告诉我做事的方法与做事的结果一样重要，他把公司变得像大家庭一般融洽，他鼓励我努力前行，并使我意识到软件工艺的种种原则，那时还没出现软件工艺这个词呢！

Alexandre Ehrenberger 是我从前的经理，后来成了我的好朋友，他经常激励我，在我决定搬去伦敦时大力支持我。他在加拿大住了六年，并告诉我怎样做才能实现伦敦之

梦。非常感谢他的友谊，感谢他给我的建议、鼓励和支持，这些都使我明白，努力工作会令梦想成真。

要去伦敦，必须做两件事。第一是学 Java 语言，为此，我努力研究 Java，并找了份与 Java 有关的工作。还有一件事就是学好英语。我要感谢可爱的 Ana Maria Netuzzi 女士，谢谢她这些年来专门给我补习英语，她是我的朋友，而且有时也是我的心理辅导师，听我诉说心事并给我宝贵建议。

在英国一家小软件公司稍微工作了一段时间后，我加入了一家创业公司。那里有很多优秀的开发者，那是我第二次感觉到有些人真的很在乎他们手头的工作。他们使我提升了自己的技术水平，但更重要的是，他们成了我在英国的第一批朋友。他们不仅令我变成了更好的开发者，而且令我融入了英国社会。感谢 Chris Webb、Greg Cawthorn、David Parry、Russell Webb、Simon Kirk 和 James Kavanagh。

加入 Valtech 公司是我职业生涯的一大进步。在那里，我初次接触了敏捷软件开发和极限编程，那是迄今为止我学到知识最多的地方。虽然我无法列出遇到的每一位优秀同事，但还是要特别提到其中几位。感谢我的良师益友 Akbar Zamir。我从他身上学到了许多，最重要的就是要开拓思维，接受新事物。他令我明白自己还有很多事情不懂。Akbar 说服我尝试测试驱动开发。他也向我介绍了领域驱动设计（Domain-Driven Design，DDD），并且告诉我许多敏捷软件开发和极限编程的知识，此外还教我如何处理人际关系、如何保持务实作风和专业精神。多谢，Akbar，我从你这里学到了很多。Valtech 公司还有几位同事也极大地影响了我的职业生涯。他们是 David Draper、Kevin Harkin、Andrew Rendell 和 James Bowman。感谢你们提供的大力指导和帮助。

在 UBS 公司上班时，我头一次真正接触了大型企业级系统，它迫使我重新考量自己对软件开发的看法。我在那里同一个优秀的团队一起工作，感到非常愉快。他们使我能够以不同的角度来观察事物。在 UBS 公司工作时发生的一件妙事，就是能和 Mashooq Badar 搭档（我们在 Valtech 公司时就是同事，后来我开办 Codurance 公司，他依然是我的搭档），他是一名非常优秀的开发者，也是我的挚友。Mash 使我有机会加入 UBS 公司。他也把 Balint Pato 带进了公司的一个项目里，使得我们之间能够紧密协作。Mash 和 Balint 同我讨论了许多事情，他们的热情、上进心与专业精神都鼓舞了我，让

我学会了很多东西，我要谢谢这两位真正的软件大师。Portia Tung 是我在 UBS 公司遇到的另一位优秀同事，她也许是我见过最好的敏捷教练。她的热情很有感染力，给我们打造了有归属感的工作环境，她积极地与我们分享她的深刻见解，告诉我们怎样使大型机构变得更加灵活，这使我能够在 UBS 公司把工作做得更好。Portia，谢谢你与我为友，谢谢你告诉我怎样把大家凝聚起来。感谢 Robert Taylor 和 Alexander Kikhtenko（也就是 Sasha，他的绰号是“The Machine”）这两位杰出的软件工程师。和你们在一起工作，特别开心。

我要感谢 David Green 的地方实在太多了。他是做软件的高手，是我的朋友，也是个特别聪明的开发者。我总是记得，晚上经常和他在酒馆里讨论项目、讨论编程技巧，讨论怎样才能把软件开发做得更好。我从 David 身上学到了很多。David 和我一起创立了伦敦软件工艺社团（LSCC），若没有他，社团恐怕持续不到今天。

2013 年 10 月，我和 Mashooq Badar 一起创办了 Codurance，这是一家咨询公司，它以软件工艺的原则和理念为基石。Codurance 公司使我们的职业生涯更上一层楼。这次总算可以按照自己的想法来运作一家公司了。而 Samir Talwar 的到来则使得这段时光更为精彩。Samir 是我们招募的首位软件工程师，这位朋友本领极强。Codurance 公司及 LSCC 今天的成功是与他分不开的。

最后衷心感谢 LSCC 中每一位热情的开发者。正因为你们愿意花费个人时间来与大家分享知识，所以我才能在这么短的时间内学到这么多。诚挚感谢 Gonçalo Silva、Samir Talwar、Tom Brand、Tom Westmacott、Emanuele Blanco、Carlos Fernandez Garcia 及 Chris Jeffery，你们是 LSCC 杰出的组织者。LSCC 变得这么棒，多亏了你们。

谈到成书过程，我要感谢 Micah Martin 与 Tyler Jennings，他们提供了与软件工艺的历史有关的知识。感谢 Kevlin Henney 给出的好建议，这优化了本书的结构，感谢他劝我把原来写好的第 1 章删掉，并重新写一遍。感谢 Gianfranco Alongi 与 Mani Sarkar 校对本书。我要特别向 Samir Talwar 和 Andrew Parker 致意，他们找到了很多语法错误和笔误，而且为提升本书质量提供了良好建议。还有许多朋友也对本书提出了建议并指出了拼写错误，在此谨表谢意。

作者简介 *About the Author*

Sandro Mancuso 很小就开始写代码，但直到 1996 年才开始以此为业。他曾供职于创业公司、软件公司、产品公司、国际咨询公司及投资银行。2013 年 10 月，Sandro 与人合资创立了 Codurance，这是一家以软件工艺的原则和理念为基础的咨询公司。

在他的职业生涯中，Sandro 用各种编程语言和技术参与了不同领域的多个项目。他以丰富的经验，将软件工艺理念与极限编程方式推广到各种规模的组织之中。Sandro 在各国因推广软件工艺原则而出名，同时也是许多全球技术会议中的知名演讲者。他的职业理想是通过提升开发者的水平来促进软件业发展，使得开发者在分享知识、技能与心得的过程中，更加擅长自己的技术，更加珍视自己的作品。

Sandro 于 2010 年开始接触软件工艺这一概念，他当时建立的伦敦软件工艺社团后来成为世界最大、最活跃的软件工艺社团，其中有两千多名软件工程师。过去四年间，他鼓励并帮助开发者在欧洲、美国及世界其他地方创建并组织软件工艺社团。