开发经理的职能

开发经理是个工作压力比较大的职位。作为“中间人”，你需要在管理层、客户、销售、开发人员等多种角色之间周旋。没人会注意你的工作做得有多好：一切都运转顺利，工作进展得波澜不惊，所有人都各得所需。但如果事情失败了，不论什么原因，可都是你的错。

要成为一名成功的开发经理，秘诀就是管理好期望，第一步就是确保所有人都理解你的职能。你和你工作相关的人，都要对开发经理的期许达成一致。

我看过很多开发经理的招聘信息，但我都不太赞同上面的描述。有一个要求深入了解大量编程语言和环境，还有一个要求66%的时间进行编程（为什么不直接写三分之二？），还有一些要求有PMO认证，类似的要求不一而足。我承认开发经理的职能是有点儿模糊不清，但像这样的招聘信息让我觉得发布这些职位的公司并没有真正思考过开发经理的职能。这种情况对公司和受雇的人来说都后患无穷。

作为开发经理，你要承担很多责任，但重要的是发布产品。你的目标是采取所有必要的措施，确保能把产品交付给客户或市场。要做到这一点，你需要确保开发团队能尽可能高效地工作，而且要确保他们有明确的目标（无论是短期的还是长期的），扫除阻碍他们工作的一切障碍。从最初的项目范围，到在客户网站上部署产品，每一步都是你的职责。你可以（而且应该）尽量把事情委派给下属去做，但你要检查事情是否和你预期的一样，如果不是可要自己投入。

[项目范围界定](http://searchcio.techtarget.com/definition/project-scope)

作为开发经理，你需要知道如何界定项目的范围。根据你所在组织的情况以及你和外部群组的协作方式，这可能是你工作的重要组成部分。如果你经常承担、负责第三方的项目，那你应该知道如何对RFP（需求建议书）作出回应，包括交付物、时间表和预算等。即便你只做内部项目，没有正式的文档系统，你也应该养成为每个项目写一份项目范围说明书的习惯。另外，如果你从事的是敏捷开发，这些文档就要随着项目的进展持续维护和更新。

“总置顶”项目

这是项目范围界定的一部分，但它应该单独说明一下。我听大家谈论过“总置顶”项目，这类项目不需要预算和时间表。这可是错误的！如果弄不清楚成本和交付物对这些“总置顶”项目有怎样的依赖，那可能会扼杀你的团队，因为这些“总置顶”项目会拖延进度、消耗其他工作需要的资源。你承担的每个项目至少都要有一个内部成本和一个交付物。你要和其他利益相关方一起协商你所承担的一切。

管理关系

记住，你是“中间人”，任何失败都是你的错，即使失败原因是你无法控制的事情。你需要和参与的人保持良好、开放的关系。

你不仅要让你的直接上司了解情况，还要让他的上司和同级别的人知道。此外，你也要让项目的其他利益相关人了解项目情况。确保他们都在“消息圈里”，能定期获得状态更新，清楚你的团队正在做什么。

谁处理客户关系？这些人可是除了老板之外你最需要知道的人。他们能管理客户期望、处理投诉（真实的或想象出来的）、与客户保持联系。另一方面，他们能让你苦不堪言，没和你核对就给客户许诺，提交不必要的Bug报告，缠着你按不切实际的时间表执行，诸如此类。

了解你的团队，他们到公司有多久了？每个人分别有什么优势和劣势？谁能和他们协作得比较好？他们有多忙？留意他们的生日、纪念日等等……虽然都是些细枝末节的事情，但意义却非同小可。

确保管理人员知道你在做什么，并能看到你的进度，这样他们才会满意。沟通和可视化是关键所在。我用过各种各样的工具，让管理人员始终能了解状态、发现更多信息。使用程序工具箱、公告板、白板及任何你能想到的工具，以便他们了解最新信息。

如果利益相关者了解你和你的团队遇到的挑战，那他们可能会少提一些不切实际的期望。我说的是他们可能会少提，而不是绝不会提。有些管理者永远不会明白为什么事情不能“运转”。这种情况下你可能得重新找个工作了。

项目计划

只要你不是在大型项目里，一般都不需要单独的项目经理。对小型或中型项目，以及使用敏捷方法的项目来说，你可以担任项目经理的角色、承担相应的责任。但开发经理并不是[经过认证的项目经理](http://www.pmi.org/Certification.aspx)。抛开传统项目管理和敏捷开发之间的争论不谈，开发经理和项目经理的关注点一直都有冲突。作为开发经理，你的工作是尽可能完成所有的事情，项目经理的工作则是确定什么时候能完成哪些内容。你必须要在两个出发点之间做好平衡。如果你的项目足够大，需要专业的项目经理或[Scrum](http://en.wikipedia.org/wiki/Scrum_(development)) Master，那就给你的团队找一个，不要尝试着亲自扮演这个角色。不过，不论是瀑布模型还是敏捷过程，你都应该确保项目计划是处于活动状态的，要持续更新、跟踪进度。

过程控制

这是工作里另一个关键的部分。不论你用的是[敏捷方法](http://en.wikipedia.org/wiki/Agile_software_development)还是[瀑布方法](http://en.wikipedia.org/wiki/Waterfall_model)，你都要掌控过程，让事情遵守流程。记住，交付是你的本职所在，任何影响交付的事情都需要你最优先处理。

你采用的开发过程是什么？是何种形式的？如果大家说它是“敏捷”过程，那就检查它是否真的敏捷（我保留着一张很方便查看的敏捷宣言海报，提醒自己遵循敏捷原则）。你的过程如何得以改进？不存在完美的系统，要不断寻求改进过程的方法。我们已经做了很多工作来应对Bug的Root Cause分析，但更多时候却是过程有缺陷，导致设计不好、代码糟糕，或者误解了客户的需求，以至于产品不能发布。

委托他人是件好事，但你必须跟进、确保事情都完成了。伟大的想法往往因为没人检查处理结果而在执行中失败。我接管过好几个项目，接手前都是各个方面都不错，唯独执行不好导致一塌糊涂。

最后，你需要向各个利益相关人报告基于确切度量数据的状态。所以要用对阅读报告的人来说有意义的方式衡量、总结过程。根据你组织的情况确定报告频率（每天、每周或根据需要）。要明确报告的频率、格式和内容。注意阅读报告的人和他们期望的详细程度，并达到这一目标。所有这些会让你的报告清晰、明确、易读。这会减少误解报告的人，但并不意味着能消除误读。读报告的人有很多事情要做，可能只会略读一下，或者按他们的方式理解，所以你要准备好澄清和解释，不过听起来可不能是在为自己辩解。

技术

我多次在工作职位上看见过这个要求。有些公司要求开发经理深入掌握特定领域的知识。作为开发经理，你并不是技术专家！把这个要求留给高级开发人员和[首席开发人员](http://en.wikipedia.org/wiki/Lead_programmer)吧。你应该熟悉现有的技术，了解新的和即将推出的技术，但不要让自己成为专家，这会耗费大量时间、从其他任务上转移你的注意力。你要非常了解团队正在使用的工具，看团队成员是否在有效地使用它们，还要知道团队何时会在知识面上出现缺口，但你不应该是“去填补”的那个人。你必须放手，委托团队的高级开发人员去掌握空缺的知识。

开发

这也是一个你需要熟悉，但不必是专家的领域。伟大的程序员能写出最好的代码，不过通常会是个糟糕的管理者。你要能区分好代码和坏代码，还应该相信你的团队负责人。你可以在关键时刻亲自投入、接手一些开发任务，但别忘了要有大局观、要聚焦于项目的完成。可不能好几天都埋头编程，忘了自己的工作 。

自动化测试

我把自动化测试和质量保障分开是因为我觉得它们有着不同的功能。两项工作通常都会指派给QA团队成员，但我发现把它们分开考虑是有益的。自动化测试包括单元测试和测试脚本，而质量保障是手动检查系统，不仅仅是为了Bug，还要关注一致的外观和体验、性能、用户接口和设计问题。（参见“[测试vs.检查](http://www.developsense.com/blog/2009/08/testing-vs-checking/)”）。

有些事情可谓“一劳永逸”——花费99%的时间去做，剩下1%的时间享受成果。自动化测试就是这样子的。记住Jell-O果冻编码原则，一个系统“此消”就会“彼长”，恰如果冻按下一头，另一头就会弹起来。我曾经遇到的情况是，每个人都确定代码改动的影响非常小，结果“小”到导致十多个自动化测试用例执行失败。这还只是自动化测试仅覆盖了90%代码的情况下，很庆幸剩余的10%代码没有出现新Bug。

作为开发经理，你需要知道测试的代码覆盖率是多少。单元测试覆盖了多少基本代码？自动化测试又覆盖了多少？具体的值是在增加还是在减少？一个好的QA组长会告诉你这些数字，但你应该有一个系统，至少要能评估具体的值。代码覆盖率的降低几乎不可避免，你可以忍受一段时间，但有些时候你必须减慢开发速度，以便QA团队能跟进上来。如果不关注测试的代码覆盖率，发布的代码就可能到处是Bug。

质量保障

我见过所谓的“冒烟测试”，但它只是质量保障过程的简单替代。QA还需要找出不一致性、偏离规范、性能下降等问题。作为开发经理，你的工作就是积极主动地去做、确保完成这些测试。让QA团队加入开发圈子，以便让他们知道接下来要做什么。我一般会为开发计划制定几个独立的QA计划，并保证它们的时间和开发计划同步。然后链接代码变化，这种方式能让开发人员在不同的位置给测试写备注，而不会和编码备注混杂在一起。

QA测试失败的原因有很多，比如糟糕的测试、代码变化、对规范的误解，以及网络和系统错误。要知道测试失败的根本原因，你必须要让开发人员和QA团队一起工作。如果你的QA人员是开发团队的一部分，这并不难做到；万一他们有各自的经理，那你必须与他们的经理进行协调。

[发布管理](http://thedailywtf.com/Articles/Release-Management-Done-Right.aspx)

根据项目的复杂度，发布版本也会是一个单独的项目。简单系统的发布可能只是简单地构建一下、拷贝覆盖可执行文件；而对复杂系统来说，发布可能需要构建大量的包（可执行文件、组件、Jar文件或DLL），添加数据库脚本，甚至添加一些第三方应用。要确保每个组件的版本都正确可是件困难又耗时的工作。在非常复杂的项目里，你可以指定一个Build Master来跟踪版本。无论如何你都要有一块记录组件及下个发布所需版本的公共区域，并且要确保每个人都能访问、能根据需要更新它们。

你还要发布代码归档，准备、测试安装程序，编写、发布版本说明，并保证合适的人能访问新版本（旧版本至少要改名，防止访问的人弄混当前版本）。

部署

部署往往被看成是发布管理的一部分，不过我觉得应该单独看待。

如果你有Beta测试站点，这就变得更加重要了。准备好站点发布新版本本身就是一个项目。谁使用你的产品？他们能接受新版本么？已经了解修改内容了么？怎么报告问题、表示赞许？测试周期是多长？你准备好在最短的停机时间内回滚版本了么？

测试结束后，你的客户会接受么？上面的所有问题都必须在推出版本之前得以解决。

行政管理职能

这是大家都讨厌的一部分工作，但它是工作的一部分，其间还需要做一些最困难的决定。预算、员工的雇用和解雇、争夺资源和空间、写报告、评估等，所有内容你都没在学校学过，但在工作过程中你可要全部学会。最重要的就是“知道自己不知道什么”！我见过很多优秀的管理者会在这部分工作里迷失，你要是自以为是，公司和你自己就会陷入很多法律问题。如果你对所做的事情有一丁点儿的怀疑，就研究一下、寻求帮助、找一些行家。自以为是会让你的公司受到起诉，你也会遭到指责，所以再强调也不为过。

团队人员配置

这是行政管理职能的一部分，但值得单独说明一下。让合适的团队成员担任合适的角色是永远达不到的理想状态，但这并不意味着你可以不努力去做。达到合适的水平很棘手，大型团队的花费要比小型团队的多、但效果却不及小型团队，不过你不应该让每个人都百分百投入。你需要预留一些空闲时间，以便团队能够应付突发的新增需求，而不会给项目带来风险。

如果团队成员因为家庭事务或其他原因需要休假，不要太紧张。就我个人而言，只要这个人是有效率的团队成员，我就会批准。已经有大量研究表明，（在合理范围内的）高缺勤率实际上比近乎全勤的团队更有效率。这是因为有人休假时，他平时负责的部分就需要团队的其他成员去学习。所以让团队成员因个人原因休假实际上会提升团队的效率。

出现人员流动可能会是个好事情。我告诉年轻的开发人员应该每五年换一个公司。这样做可以接触不同的环境和技术，以及不同的过程。我看简历的时候会关注这方面的内容，所以我也希望我团队里的人可以继续发展。你需要对此作出规划，不能因为项目或系统的特定部分只有一个人知道而恳求他留下来。确保信息和知识是团队内共享的，而不是集中在一两个关键人物那里。

最后，每个开发经理都会经历解聘。无论是因为金融原因（团队裁员）还是表现不佳，你都必须做好准备，了解公司的政策和程序。你甚至要知道法律责任。在加拿大，无故解聘某人需要遵守法定解雇要求，人们也享有普通法里“提前通知”的权利：有时间找下一份工作，没有收入损失。只符合法定要求会引来非法解雇的诉讼，所以最好从一开始就让他们能够接受，而不是跑去找律师。善待他们会赢得好的口碑，在准备重新雇用人的时候这可没什么坏处。

结论

上面的清单可不短。你不必是任何一个方面的专家，事实上不是专家才最好。不是专家的话，你才不会在做更有价值的事情时深入、接手某项具体的工作。

对我来说，如何成为一名开发经理最好的例子来自制造业里[Bata鞋](http://en.wikipedia.org/wiki/Bata_Shoes)的故事。在原来的工厂里（现在在捷克共和国），Thomaz Bata把办公室建在货运电梯里。无论哪一楼层出现生产问题，Thomaz Bata都能把整个办公室移到那一层进行微观管理，直到问题得以解决。然后他会下到一楼，不受生产楼层的干扰、继续运营公司。需要的时候进行微观管理，其他时候让团队继续工作、你处理其他的事情。