《软件管理沉思录》读书笔记

本书要点:

这本书虽然主要讲团队管理，但作者在书中也以大量的篇幅分析了在一个团队中要如何管理你自己，包括你自己的时间、你自己的计划，甚至是你自己的上司。作者对计划的高度重视、对团队建设的透彻理解，都给我留下了深刻的印象。对于如何在团队中进行沟通和交流、如何建设高效团队、如何说服你的上司等，书中都有极为精彩的分析。

内容简介:

本书为软件工程领域权威人士之作，书中首先深入讲解了计划类型和计划过程，接着分析了项目团队建设和激励，然后描写了如何与经理共事，如何说服他们采用最佳实践，最后探讨了个人职责、承诺和过程。

精编书摘:

第一部分 管理你的项目

在软件开发过程中，为了保证质量始终如一，必须遵循以下 8 个步骤：

(1) 确立质量控制的策略、目标和计划；
(2) 正确训练、指导和支持开发人员及其团队；
(3) 确立和维护软件需求的质量管理过程；
(4) 确立和维护软件工程过程的统计控制；
(5) 审查、检查并评估所有的产品制品；
(6) 评估所有缺陷，加以更正并用以识别、纠正和预防其他类似问题；
(7) 确立和维护配置管理和变更控制系统；
(8) 持续改进开发过程。

软件产品的质量应当被定义为产品对用户的有用性。

只有得到了清晰的需求，才可能开发出高质量的程序。

把发现或识别缺陷的问题与确定其原因的问题分开，是十分必要的。简单地统计和记录软件产品中出现的缺陷，并不需要具体说明原因或者进行指责。

软件工程师是发现和更正自己所编写程序里的缺陷的最佳人选。因此，软件工程师为自己的程序质量承担个人责任是十分重要。当然，学习编写无缺陷的程序是一个巨大的挑战。它并不是一件任何人都可以迅速和轻易完成的任务，它需要数据、有效的技术和娴熟的技巧作为支撑。然而，如果不去努力开发无缺陷的产品，那么你将永远也不可能达到这样的要求。

软件质量是一段永远都没有终点的旅程，每一步质量改进都会为下一步质量改进提供所需的知识、经验以及数据。因此，应当聚焦于持续不断的质量改进，并且帮助你的项目团队成员真正相信并且遵循这些质量管理原则。

软件质量改进会经历以下阶段：

1. 测试并更正。在这一阶段，你要想方设法尽快让项目团队进入旅程的第 2 至第 8 阶段。
2. 检查。开发者和管理者在测试之前就开始消除缺陷。
3. 局部测评。当检查程序成熟之后，一些团队开始着手进行测评，并且使用检查得到的数据一方面来改进检查过程，另一方面把检查聚焦在最具破坏性的产品缺陷上。
4. 质量到人。
5. 个人测评。
6. 设计。一旦开发人员学会了管理他们的编码缺陷，他们就可以开始关注设计缺陷。
7. 缺陷预防
8. 基于用户的测评。这时面临的主要挑战是如何理解那些对用户来讲最重要的质量特征，并且使用一种对你以及对用户都有意义的方式来测评这些特征。

目标之所以重要，主要是基于以下两条原因：它们提供了努力的焦点，而且建立了一种优先级次序。如果没有一个清晰的、得到一致认可的目标，开发人员很难高标准地完成工作。而如果有了一个明确的目标，你就知道了什么是要做的，同时还有了清晰的工作的方向。

即使你的同伴和管理者认为目标已经十分清楚，只要你对它有任何疑问，那么一定要大声说出来。要坚持让你的问题得到解决。

最难以制定计划的时候，也是最需要计划的时候。当项目团队面临巨大的交付压力时，他们一定不能让步，要坚持制定一份计划。

作两类计划：阶段计划和产品计划。

为每一项主要的工作制定产品计划。计划对软件工程师来说是及其重要的工作，如果你向成为一位有效率的软件工程师，你就需要知道如何制定计划。制定计划的关键是实践，因此想得到最佳的实践，从现在就应开始制定计划。一份合格的产品计划应当包括三项内容：将要生产的产品规格和重要的性能指标；估算工作所需的时间；进度预测。

与你的管理者共同审核详细的计划。

计划必须是易于理解、清晰明白、详细具体、精确缜密、准确无误。

若你不能使计划准确无误，那就常做计划。

计划必须得到维护。

即使项目计划得十分周密，变化的环境也会经常要求计划作出调整。除非团队成员定期修改他们的计划。否则计划对于手头的工作会越来越没有意义。项目团队应当进行定期分析，找出那些提前完成工作、能处理好额外工作量的成员、以及那些落后于进度、需要减少任务的成员。如果项目团队不能保持详细计划的平衡，有一些成员可能最终会远远地落在后面，甚至耽误整个项目进度。

第二部分 管理你的团队

团队应致力于共同的目标，没有一些共同的目标，人们就不会努力—他们只是投入时间。团队成员要担负一定的角色，角色提供了某种拥有和归属的感觉，这种感觉指导团队成员完成工作。团队行为中更加重要的一个方面是协作和相互依赖。每位成员的工作必定都或多或少地依赖于其他成员的表现。

团队合作会比个人独立工作表现更出色，团队成员联合后的才智使得更全面的知识成为可能。

常见的团队问题主要涉及领导、合作、参与、拖延、质量、功能蔓延和评估等方面。

团队失败的主要原因有资源不足、领导问题、不可能的目标和士气问题。

一个得力的团队在他们所做的任何事中都体现出一种道德规范、一种态度和一种活力。

团队需要团结、富有挑战性的目标、反馈以及通用的工作架构。

团队随时间而成长，随着团队凝聚力的形成，成员们接受了一套共同的团队目标。

团队交流的三要素是透明，倾听和协商。

一支团队想要成为高效团队，所需要的不仅仅只是恰当的任务和工作环境，它的任务必须是重要的，所处的环境必须有助于团队协作。团队必须直面一个艰巨的挑战，而且要有空间去计划和管理任务。目标追踪和反馈极其重要。

项目团队需要经常召开会议讨论议题，解决问题以及计划工作。

为了真正理解复杂的技术问题，集中全部注意力搞清发言者讲了些什么是绝对必要的。

目标追踪和反馈是极其重要的。高效的团队了解他们的表现，能看到他们向目标前进的进度。

承诺是一项必须学习的道德规范，只有基于计划才能做出负责任的承诺。

你有责任与整个团队分享你的思想。团队所有成员都应当奉献他们所知道的一切。团队创建需要所有成员的主动参与。

团队成员需要清楚地表明自己的看法，并且引起工作团队的关注，同时也要注意他人的意见并接受关注。

团队创建的义务有以下几条：作为团队的一员接受责任，并尽你最大的能力扮演好你的角色；参与确定团队目标和计划，并努力实现这些目标和计划；建立并维护一支高效和合作的团队。

第三部分 管理你的领导

前面两个模块都提到领导对于整个团队的重要性，甚至说领导力决定了团队的成败，因为领导拥有资源分配和最终决定的权利。我们常说“屁股决定脑袋”不是没有道理。领导是开发人员和更上级之间连接的纽带，领导所处的位置会使其以更广阔的视角来看问题，而视角不同、关注重点不同、目标优先级不同往往会导致与开发人员意愿产生或大或小的矛盾冲突。所以本模块的核心是计划，作者提议通过计划来与领导、管理者更加有效的沟通协商。

领导区别于其他团队成员最本质的一点就是领导是通过授权委派别人来完成工作，而不是事事都亲自完成。领导利用职权会有一些独断专行的行为。当然，在有些情况下，独断专行的行为无可避免。例如情况危急需要迅速做出决定的时候，领导者高度集权做出调度处理是最及时高效的处理方式；或者是没有团队成员自愿接手的时候，由领导直接任命比较快速有效；又或者有些循规蹈矩、鲜有变更的事情，处理方法已然固化，直接由领导者拍板决定节省了团队整体的时间，也符合团队期望和习惯。但是 Lord Acton 大概一百年前就断言：Power corrupts, and absolute power corrupts absolutely. 绝对权力导致专制武断的领导，这带来的隐患不容小觑，一旦爆发对整个团队可以说是灾难性的。专断的行为若是被迅速而且成功地执行后，独断专行者对自己一贯正确的信心会被加强，情感上得到使用权利的满足感。Petty powers are most corrupting1. 越是卑微的、有限的权利越容易独裁腐化。防治这一现象发生的手段之一就是采用集体决策。比如制定计划，应该交给团队自己制定最切合实际的计划，得到全体成员的共同承诺，而不是让领导制定出一个只是强制要求交付期限和项目完成成本的没有指导意义、单方决定的不切实际的无用计划。同时，详尽计划的制定可以有效地与领导跨越层次等级和观念意识上的鸿沟来沟通协商，让领导充分了解并且相信理想与现实、目标和实现之间的差距，然后做出一定程度的调整和让步。

不过，在执行计划过程中，往往会受到来自与管理层的控制欲等而带来的阻力，例如管理层不定期分配计划外的任务却没有更新计划。随着这种情况出现的累积，原先可靠的计划

会渐渐失效，不得不加班甚至延期，从而管理层开始质疑团队计划的可靠性。书中所说的 Lean Really Is Mean 给了我一些启发，是否可以考虑将这些额外的重复的任务交给支持人员解决，或者针对某些普遍性问题对外发布解决手册，对于一些不常见的、难以解决的额外任务由支持人员汇总后找一个特定的时间反馈给开发人员。

第四部分 管理你自己

本模块回归到了最根本的单位——自己，也就是本模块的核心。只有注重自己的管理和提升才能带动他人，从而成就高效且不断进步的团队。管理我们自己以便知道自己的时间是如何流逝的，通过历史数据更精确地制定计划、发现工作过程中存在的问题和有改进空间的方面，从而优化计划，节约自己的时间、提高自己的效率，实现利益最大化。

Peter Drucker，著名的“当代管理学之父”曾提出观点：You can’t manage knowledge workers. They have to manage themselves. 软件开发人员属于知识工作者范畴，必须自己管理自己，但问题是大多数开发人员其实并不清晰地懂得如何去正确高效地管理自己，合理分配利用自己的时间。”Know The Time” if he wants to, and be well on the road toward contribution and effectiveness.而书中提到开发人员真正花在任务上的时间其实并没有看起来的那么多，主要原因作者总结成为四点：①Interruptions ，开发过程中常常会被外界打扰而不得不中断任务，这样会使工作非常低效且容易出错，为了继续中断前的工作往往我们还需要时间恢复中断前的思路，为此作者建议记录问题日志来帮助快速恢复，也防止遗漏需要完成的任务；②Only some tasks contribute directly to our project ，对于一些杂务，建议交给专门的支持人员，虽然支持人员刚开始接触工作的时候表现可能并不尽如人意，但是熟练后团队的工作效率和质量会有显著的提升；③Our processes are often informal and our working practices ad hoc. 开发过程往往是非正式的，开发实践也往往是临时性的；④Even if we wanted to, it is hard to do demanding intellectual work for long uninterrupted periods.3 软件开发工作属于脑力劳动，尤其是探索创新型开发工作非常耗神，难以长时间、持续开发，需要让脑子休息放松，这样才能保证高质量的工作，应对办法很简单，采用多类型任务交替的安排，劳逸结合。

The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination.4 因此，想要争取更多自由创造和自主开发的权利，就要做自己的管理者，而不是管理层不切实际地施压、七脚八手地“指导”你的工作。而想实现这一目标，你就必须做到能掌控自己的工作，更重要的是让管理层相信你，从而同意你管理自己。前文数次提到与管理层协商的法宝——计划，在这个场景下仍然奏效。计划执行期间，要有主人翁心态。《人月神话》中指出 Day-by-day schedule slippage is harder to recognize, harder to prevent, and harder to make up than calamities.所以，发现问题要鼓起勇气积极面对并尽快告知相关人员着手解决，切莫逃避忽视问题，拖延不会解决问题，只会带来更糟的结果。

阅读感悟:

本书从管理你的项目、管理你的团队、管理你的领导、管理你自己四方面简明扼要地阐述作者在长期从事技术管理和研究工作的过程中总结的对于软件管理领域的一些心得体会。你无论是普通员工还是管理层，无论是从事软件行业还是其他行业，都可以从本书中获益良多。以上感想只是初读本书的浅薄之见，我相信随着自己知识和阅历的不断累积，再翻阅本书会有新的视角和领悟。

最后以 Peter Drucker 作为本文的结束，也是管理自我、追求卓越的起点。
Effectiveness can be learned.

MF1632020 管登荣