INCREDIBUILD

6个步骤
提升开发速度

6 STEPS TO SPEED UP DEVELOPER VELOCITY

介绍

2020 年，麦肯锡公司调查了 450 家企业级公司，以了解开发速度对企业成功的影响。调查结果很显著：开发团队的工作效率越高，企业在各方面的成绩越好。

麦肯锡的开发速度指数（DVI）追踪了从架构到员工满意度等 50 个参数，得分最高的公司收益增长比平均水平高出 4 到 5 倍，股东收益高出 60%，营业利润率超出 20%。这还不包括在创新指标上高出 55% 的成绩。

这也符合我们自己的发现。在 2022 年，我们调查了 300 多名企业级公司的高级经理，发现大多数人耗费大量时间等待开发关键任务（如构建）的完成。然而，一个拥有优越环境、工具和进程的团队指出，他们每天可以节省多达两个小时（或一年 64 天）的等待时间。

麦肯锡将开发速度定义为“通过提升软件开发速度驱动业务绩效变革。”

简单来说，开发速度是团队在指定时间内完成目标或任务的速度。

例如，如果我们测量了一个团队在 3 周冲刺内完成了 14 个目标，我们可以通常假设这是一个很好的基线。随着时间的推移，你可以建立一个平均值，因此可以灵活分配工作，这能让团队效率最大化。

快速开发是什么意思？

衡量开发速度听起来很简单，只需要测量一段时间内完成的工作量。但是这个简单的定义 忽略了几个影响速度的不同因素，更严重的是，这些因素会影响速度测量的结果。

如果一个团队在两周冲刺内完成了 20 个目标，大多数人都会称赞说这项工作效率不错。可如果这 20 个项目中充斥着劣质代码、偷工减料以及漏洞百出的产品呢？或者说如果开发人员工作过度，无法在截止日期前完成任务怎么办？在这种情况下，速度还重要吗？

另一个需要考虑的问题是，速度不一定是最好的指标。

如果你的队伍因追求速度而发生事故，损失会比低效却稳定时更严重。

所以也许最好还是谈谈可控的速度，或者说适度加速。

理解开发速度，我们需要正确定义。所以，从一开始，确定开发速度不只是一个时间长短的问题，更是团队为了更快部署工作的方法，而且要创造一个环境，让工作在时限内做好。

开发速度——新的定义

在问题还没太抽象之前，我们先来确定一下开发速度的定义。问题是速度并不是一个整体的术语，不同的方法以不同的方式测量速度(即使目标是相同的)。这很大程度上是因为工作是零散的，工作流程的管理也多种多样。让我们来探讨一些最流行的方法:

敏捷

在这种情况下，我们称之为“ 冲刺速度(sprint velocity)”，它的测量方法非常简单，找出最近三到五次冲刺用户故事点的平均值。这是一个标准的衡量方法，但在规划未来的冲刺时，对工作量的预测是不错的。即便如此，它也逃不过我们刚才提到的问题。

Scrum

Scrum 使用两种速度度量方法，每种方法都有自己的问题。第一种是实际速度，它的测量方法几乎与敏捷速度相同，即前几轮冲刺中完成的故事点数量的平均值。另一种是预期速度，这是一个前瞻性的指标，也还算有用，但可能相当复杂。

在这种情况下，开发人员估计在一组冲刺中可以完成多少个用户故事点。例如，一个团队可能可以在三次冲刺中完成 150 个用户故事点。

在这种情况下，他们的预期速度将是每次冲刺 50 个故事点。

这些衡量指标非常直接，浅层意义上，在开始新项目时，它可以有效帮助计划和了解团队能力。但是，当我们过度依赖它们，或者我们没有正确地使用这些衡量指标时，它们可能会出现问题。为了最大限度地利用它们，让我们首先了解接下来最重要的问题，为什么我们需要测量开发速度?

为什么我们要测量速度?

最准确的答案是,我们测量速度是为了跟踪开发人员的效率,以及最大程度优化开发时间。开发团队领导需要了解团队成员在每一轮冲刺中可以完成的工作量,跟踪他们的完成轨迹,并最大化团队的生产力。

但是过分关注效率本身对团队的长期成功是有损害的。速度永远不应该成为目标。当过于关注完成速度时,很容易忽略一个前提,速度提升可以帮助团队进步。所以也许我们问错了问题。我们应该问,为什么为我们应该测量开发速度:

理解开发团队的实际能力

这听起来很矛盾,但这对于理解团队的健康效率水平至关重要。也有利于长期计划以及更智能地规划冲刺。

找到需要改进和解决的问题

开发速度是一个流动的衡量标准,因此可以为某些任务或项目提供重要意见。这可以帮助团队更好地准备,以及找到正确的工具和解决方案来避免陷阱。大脑健康可以带来更好、更快的结果。

确保团队没有过度劳累

我们上面提到的麦肯锡调查指出,在影响速度的众多因素中,心理健康始终被评为最重要或影响最大的因素。找到工作和心理健康的正确平衡可以带来更好、更快的结果。

微调开发团队的工作流程

测量速度是一项面向后方的工作。但它可以帮助理解流程中的微小变化如何影响整体的工作。也许它只是一个简单的添加,比如一层代码审查或使一些步骤并行工作。准确衡量这些变化也能带来很多价值。

效率陷阱

在我们继续讨论开发速度之前，我们应该稍微离题谈谈一个大的问题——我们对“效率”的过度依赖和痴迷。

单就其本身而言，脱离上下文去讨论，效率是一件好事。我们想要确保工作高效，尽快实现目标，避免浪费精力。然而，这种期望经常被用来掩盖一些错误的做法和工作流程，这些做法和流程很快就会产生适得其反的效果。

我们执着于最大化利用每一秒，这早已不新鲜，尽管这并不完全遵循生理规律。奥利弗·伯克曼(Oliver Burkeman)在写效率驱动的心理影响时指出，随着工业化的发展，我们过度关注效率，也过于痴迷高效利用时间。我们需要从每一种资源中获得更多收益，并将浪费降到最低。

虽然对效率的关注也是此起彼落，效率驱动也发生了变化和变异，但对效率的痴迷不一定会减少。这对开发人员来说是一个问题，因为它创造了一种以效率本身为目标的趋势，反过来意味着任何低于最高效率的管理都将是失败的。

然而，我们不能忽视效率。它是一个很好的，也是非常基本的团队效率的晴雨表，它可以帮助确定优先级。如果它保持在适当的环境中，作为当前活动的短期度量，那么开发效率（作为速度的产物）可以是一个有用的标尺。

问题出现在于当效率被误用作为衡量标准时，它是如何影响工作流程的。正如伯克曼所指出的：

> **“更有效率，只会让你更匆忙。快速解决问题，又快速生产问题。”**

这是真的。当目标是效率和时间最优化时，完成一个目标就意味着下一个目标很快就会到来。完成更多的任务意味着开发人员可以做更多的工作，新的工作就会堆积如山。

这是要避免的陷阱。如果工作节奏保持不变，效率是一个很好的衡量标准。如果我们把它作为创造更多工作的动力，那么没有人会是高效的，因为他们总是在追逐不断增长的未完成的任务。

平衡速度与效率

所以，总结一下。开发速度可以是一个很好的衡量标准，但如果使用不当，它也可能是一个糟糕的标准。这给我们带来了什么？逻辑上，永远不会有完美的指标，但是我们可以尽可能让其变得有用。

鉴于这一点，我们的目标应该是让团队尽可能高效地工作，而不是过分强调效率作为目标。

这似乎令人费解，但其实很简单。开发速度不是一个前瞻性的衡量标准，但它可以帮助我们提前粗略计划。

“将开发人员放在合适的环境中，可以减少团队压力，避开不切实际的 KPI 以及 KPI 回报率。”

让我们用一个例子来更好地理解开发速度是如何产生有益或有害的作用的。你有一个由 5 名开发人员组成的团队，他们正在进行一个新的应用程序开发项目。你已经确定了一种敏捷方法以确保工作按时完成，但需要确切地了解如何分工以实现准时交付。

在分解了大概之后，你根据之前的工作设定了你的第一个冲刺目标，即完成 60 个故事点（如果使用得当，开发速度可能会很有用）。然而，经过三周冲刺后，你发现实际的开发速度更接近每周 45-50 个故事点。

这就是开发速度二分法的由来。你的团队“表现不佳”，他们无法完成之前设定的开发指标。你会怎么做？一方面，如果使用速度作为 KPI，那么你的团队显然没有效率。他们需要做更多的工作来追赶指标。

另一方面，你可以说：“ 嗯，好像有什么东西影响了我们的工作速度。” 这可能会让你快开始思考上一个项目和这个项目之间的差异。是否是工具链中的新工具，既定流程的变化，团队成员的减少，或者只是工作节奏太快？以此不断改进，确定一个合理的指标。

所以，你会怎么做呢？你可以要求你的团队变得“更有效率”，去达到一个不切实际的效率目标，反而制造出更多的问题、更多的压力，结果反而更差。在这种情况下，衡量开发速度将不利于你的团队。他们会过度工作，压力过大，并且不太可能交付好的产品。

或者，你可以采取相反的路线。与其严厉批评你的团队，你可能会注意到这个项目更复杂，你是一个团队成员，此前有效的进程可能不适用于现在的项目。在这种情况下，开发速度是一个识别问题并努力解决问题的好方法。也许你的团队因为交付太多而没有机会检查质量，因此倍感压力。在这种情况下，降低每一轮冲刺的故事点可以减少压力并提升工作质量。

这就留下了两个问题。首先是何时以及如何使用开发速度，你可以在上面找到答案。

第二个，可能更实际的是，如何提高团队的开发速度，且避免负面影响？

如何提升开发团队的速度?

到目前为止,这个问题依然比较抽象,但我们可以变得更为实际一些。如果仅仅强迫你的团队更快地工作不能让你得到好的结果,那么你能做什么?这里有一些使用策略,可能会有所帮助。

1.优先考虑开发人员的利益

回到我们上面引用的麦肯锡研究,在公司的多方面指标中,决定开发速度的最大因素是心理安全。

组织重视构建安全、开放的环境,让开发人员可以试验、失败和学习,更有可能拥有高效的开发人员。

心理安全的最大原则之一 是“共同的信念,即在追求创新解决问题的过程中,可以允许员工去冒险,并保护他们的创意。”

但不仅仅是在开发方面,理解开发人员是人类,由于各种各样的原因,他们可能无法永远保持高效,这可以帮助团队更有效地衡量开发速度。

这意味着为团队创造时间,让他们停止工作,构建流程,找到可以减少加班、避免过度加班的工具。

优先考虑开发人员的健康也意味着他们会更有动力,更愿意尝试,更能适合按时交付高质量的代码。

2.失败是好事,快速失败更好

世上没有完美的软件。事实上,大多数软件都是反复失败的产物。不同的是,好的软件是从这些错误中学习的结果。

无论是客户投诉还是构建错误,或者只是一个不该出现的逗号,如果结果是改进,那么失败就是有意义的。如果你能创造一个循环,让失败更快地转化为积极的结果,那就更好了。

在产品方面,即使是像 AB 测试这样的小步骤,也可以更快地对功能、更改和更新进行反馈。测试驱动开发和静态代码分析等方法可以让开发团队更快地发现错误,并在产品生命周期中更早地修复它们。

转向更小的迭代和更小的故事点可以让团队快速找到解决方案,并花时间迭代和改进。

3.构建好的架构，而不是泥球

你选择什么开发和代码管理策略真的不重要，有一个明确的架构和流程将帮助你的团队减少摩擦，工作会推进得更快。没有什么比一个由混乱的代码和架构组成的巨大“泥球”更能阻碍工作效率了。关键目标是保持概念上的完整性。

因此，在构建基础设施时，要考虑这几个至关重要的因素。先考虑代码架构，但也要考虑比如维护适当的文档和发行说明、有效的沟通渠道，以及易于复制和迭代的清晰流程等方面的问题。

拥有更清晰的代码文档，意味着更容易找到错误和 bug，有效的沟通可以让团队清楚地识别问题、分享想法，并进行合作。减少混乱进程的失败点以及出错率，你可以让团队更快乐、更有效。

4.支付技术债务

就像我们上面提到的，完美的代码是一个白日梦。无论你的团队建设得多么好，架构多么完美，文档多么丰富，任何长期构建的代码都会积累技术债务。这里的“快速修复”，任何一个建立在过时架构上的功能，或更新时缺乏测试，都能导致软件突然失败，而且不是以一种好的方式。

技术债务越高，你的团队就需要花费越多的时间去解决这些债务。

反过来，他们本该用于构建新软件的时间就是减少。

自动化部分工作流程，无论是在测试、bug 查找还是代码分析中，都可以帮助减少这些问题。同样，不断地审查你的流程和技术可以帮助你降低债务。

5.管理代码分支

长时间运行代码的另一个副作用是，最终代码库的每个部分都会有分支，分支继续分支。这本身并不是坏事，它让团队并行处理问题，并更快地解决问题。但是当这些分支没有得到有效管理时，开发团队将花费更多的时间来处理代码膨胀，并陷入没有合并或不应该打开的长分支困境中。

拥有一个有效的分支管理策略可以让你的开发人员工作得更快，让代码更容易访问，避免最终代码分支森林的困扰。

一种策略是保持小的变更，有效地减少分支的大小，使它们更易于管理，以及更具可读性。对于你根本无法分解的代码分支，找到一种重构方法（例如添加新的抽象层）可以帮助你减少它们的大小，且不影响代码质量，让生活更容易。

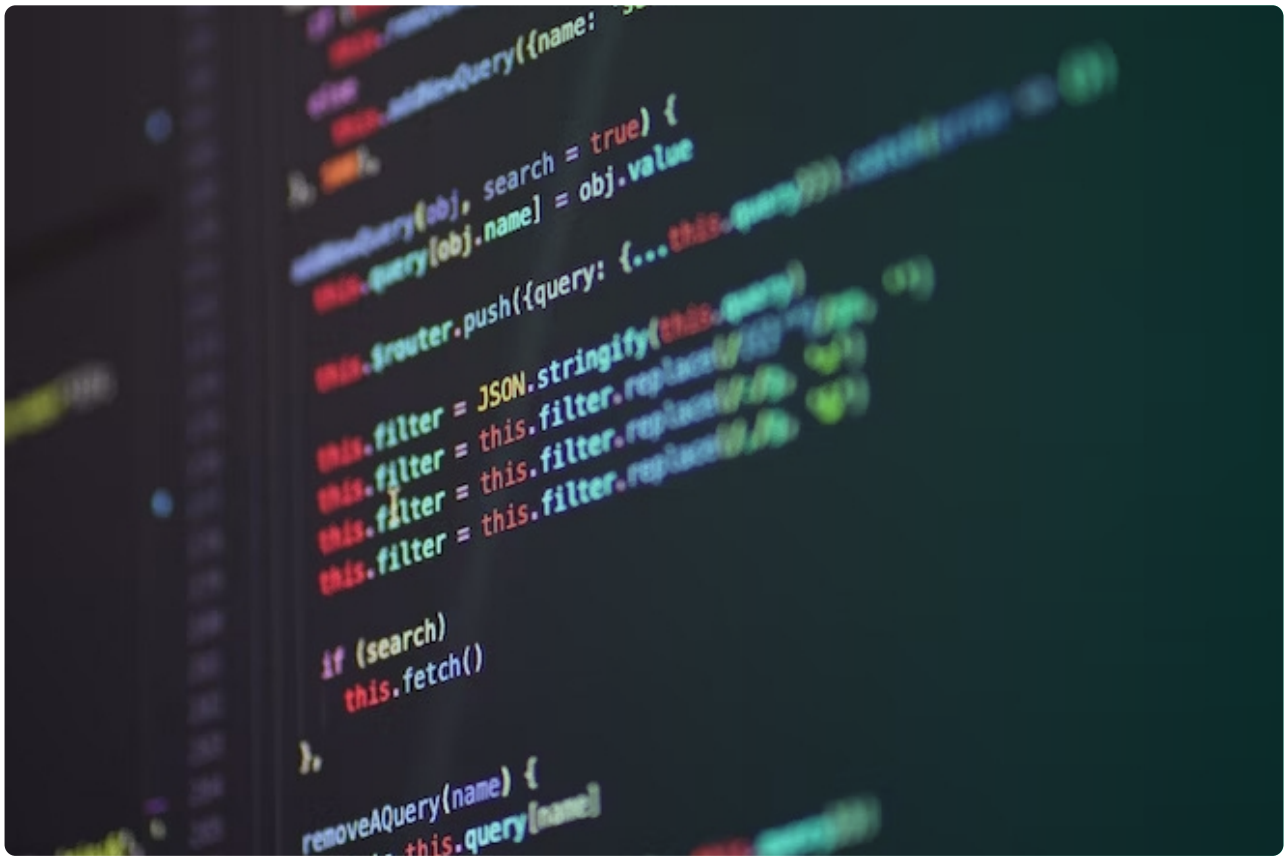
6.正确使用 KPI

KPI 很难避免，有时它们可能极为有效。它们可以帮助了解开发进度和交付状态，但也可能分散注意力。

过度依赖 KPI（如开发速度）会让团队只见树木不见森林。达成 KPI 目标是一个很好的激励因素。但它不是，也不应该成为团队的主要目标。我们已经讨论了速度，所以让我们使用另一个例子：代码覆盖率。

毫无疑问，代码覆盖率很重要，但当你努力专注于实现一个特定的目标时，你可以开始做 naïve 测试，覆盖代码，但并没有出现任何相关或有趣的场景。

相反，应该基于创建软件的主要目标，专注于设置现实和相关的目标，并将 KPI 用作工具，而不是目的。



7.避免欲速则不达的陷阱

在快节奏的开发世界中，开发速度是一个绕不开的指标。应用程序实时在线，软件不断更新，这意味着团队很少有时间在项目间隙喘口气。这就是为什么开发速度是一个很好的工具。

理解速度可以帮助团队，知道是什么问题导致速度变慢，什么时候某些团队成员可能需要更轻的工作量，甚至哪些流程更有利于又好又快地工作。

另一方面，过于依赖效率指标或过于关注KPI，反而忽略了真正的目标，可能会导致工作变慢，或者工作变快，但开发人员满意度降低。

找到工作实践、环境、工具和过程的正确组合至关重要，它可以为团队提供心理安全感、技术能力和有效的工作流程，从而真正构建最出色，也最接近完美的软件。

如需了解更多信息：

www.incredibuild.cn

电话：+86 -28 -63207464 | 邮箱：marketing_cn@incredibuild.cn