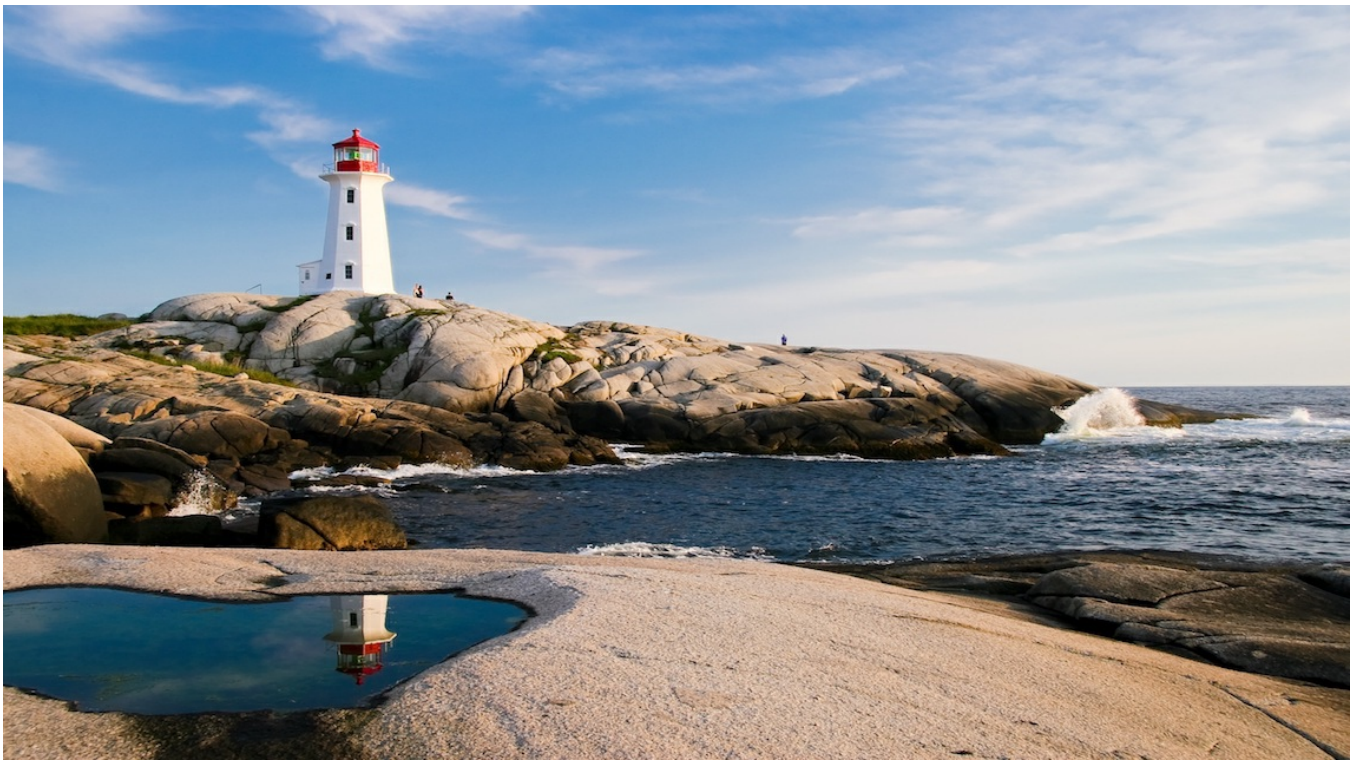
60 | 三视角：定位、自省与多维

2018-12-19 胡峰



记得以前阅读时碰到过一个观点，是关于“视角”的，其中说道：“视角的选择，对解题的难易，关系重大”。而关于成长，放到程序模型中来类比，就是一道图论题，我们求解的是适合自己的最优路径。

面对这道成长路径的难题，我们可以从哪些视角来求解？我自己找到了下面三个视角。

定位

定位，是一个时间视角，回顾初心，定位未来。

还记得当初为什么选择程序员这个职业么？如今程序员所在的行业处于发展上升期，薪酬待遇整体高于传统行业，所以各类程序员培训机构如雨后春笋涌现，流水线般地为各类只差程序员的公司批量供应，这样的批量生产似乎有点把程序员当成了工厂的工人。

而程序员的工作实际更贴近于工匠，既有创造性的工艺性工作，也有模式化的工程性工作。想清楚自己成为程序员的初衷是什么？如果只是为了进入一个相对高薪的行业，得到一份工资高于平均水准的工作，终究是走不了太远的。

很多入门的新手程序员都是刚从学校毕业的，曾记得在吴多益的一篇工程师成长分享的材料上，如是说：

> 从小到大的教育，你习惯性被安排：“课后作业是 X1、X2，后天必须交”“本学期的必修课有

> XX、YY，必选的选修课有 ZZ、WW”。
>
> 十几年来你都是这样度过的，但现在你已经不在学校了，你要安排你的未来。

刚入职场的程序员依然保持这个习惯，等着主管来安排。但如果你每天的工作就只是完成被安排好的任务，那么你自己的成长就会非常缓慢，因为主管安排任务时并没有那么多的精力来考虑任务是否适合个人的成长发展。这些任务是组织发展的需要，而不一定适合个人的成长发展，但组织是付了薪酬来让你完成任务的，所以这是工作的必需部分。

自己才是职业生涯的管理者，要想清楚自己的发展路径：远期的理想是什么？近期的规划是什么？而今日的任务和功课又是什么？今日之任务或功课哪些有助于近期之规划的实现，而近期之规划是否有利于远期之理想？

为什么今日除了任务外还要有功课？功课是学校里的概念，职场里没有，所以离开学校进入职场的功课都是自己给自己安排的。任务来自主管的安排，功课来自自己的安排。很多时候你只去完成任务却从未给自己安排功课，而等着被安排和主动安排之间，在未来将产生巨大的差别。

一开始你可能只有模糊的远期理想，也没那么清晰的近期规划，但一定要有足够清晰明确的今日任务和功课，即使在你的主管因为各种原因没给你安排的情况下。虽说方向不太可能一朝就定好，但也不要不管不顾地埋头走路，你需要抬头看路，定期检视，因为如今环境和大势的变化也很快。在边走边看的过程中逐步就清晰了近期的规划，甚至远期的理想。

另外，主管在你职业发展的路上，除了大部分时候给你安排任务，偶尔也可能给你创造机会，而机会出现时你能否抓住，全在今日之功课上。

定位的视角，是关于一条成长的时间路径，它关乎：昨日初心，今日功课，明日机会。

自省

自省，自我的视角，关乎自身，是一个观察自己成长路上行为的角度。

乔治·海尔迈耶（George Heilmeier），是一位美国工程师和技术管理者，他也是液晶显示技术的主要发明者之一。他在科研领域最著名的事情就是他提出的“海尔迈耶系列问题”：

> 你要做什么？不要用术语，清晰地表述你的目标。
>
> 这件事现在是怎么做的？现在的做法有什么局限？
>
> 谁在关心？你的方法有哪些创新？你为什么觉得你的方法能够成功？
>
> 如果你的方法能够成功，它能带来怎样的变化？

> 你的方法需要花多少钱？需要花费多少资源？要怎样在过程中和结束时进行评估？

我觉得这个系列问题，用在程序员个人成长上也有异曲同工之妙，因为现在的技术方向和路线太多，即使选定了路线依然会有很多茫然和困惑。如果你想要学习一门新技术或在项目中引入一项技术，就可以试试套用“海尔迈耶系列问题”来自省一番。

- 你学习这项技术的目标是什么？清晰地表述出来。
- 这项技术现在是怎么做的？有什么局限吗？
- 这项技术有什么创新之处？为什么它能够取得成功？要是在项目中引入这项技术，谁会关心？
- 如果这项技术能成功，会带来怎样的变化？
- 采用这项技术的成本、风险和收益比如何？你需要花费多少资源（时间、金钱）？如何去评估它的效果？

程序员有时粗浅地学习并了解了一点新技术，就想着如何应用到真实的项目中。这时用上面的问题来问问自己，如果有回答不上来的，说明你对这项技术掌握并不充分，那就还不足以应用到实际项目里。

除了技术领域，你成长路上的许多行动，都可以此为参考坐标来反思：“这项行动的目标清晰吗？行动的方法有可参考的吗，局限在哪？我能有何创新之处？完成这项行动，会给我带来怎样的变化？我要付出多少时间、金钱和精力？行动过程中我该如何评估？行动结束的标准是什么？”

这就是**自省，从埋头做事，到旁观者视角的自我反思**。

多维

多维，是一个空间视角，关乎如何选择不同维度的成长路径。

有些时候，程序员写了几年代码，觉得太枯燥乏味，就想着是不是可以转管理，比如转技术主管之类的。从技术到管理似乎就是一条多维度的发展路径，是这样吗？不是的，这不叫多维扩展，而仅仅是想从一个维度逃离，转换到另一个维度。

打造多维度竞争力的前提是，要先在一个维度上做得足够好，让其成为你赖以生存的维度，这个维度就是你的核心基础维度，而它是其他维度得以发展的根基。其中，“足够好”的程度，可能就是指我们常说的“精通”。

关于“精通”的概念，每个人的理解可能会有所不同，但我认为“精通”肯定不是无所不知，而是可以拆解成两个层面：第一，如学校时期学过的卖油翁所说的“无他， 惟手熟尔”；第二，在一个领域形成自己的体系和方法论。

第一个层面，表达了在当前维度的不断精进，在精进这个方向上，有一本书和咱们专栏主题类

似，但更微观一些，偏向于“术”的层面，但又有点从“术”悟“道”的意思。这本书叫《程序员修炼之道：从小工到专家》，书里覆盖了一名程序员真正面临的一些问题，比如：

> 与软件腐烂作斗争
> 避开重复知识的陷阱
> 编写灵活、动态、可适应的代码
> 使你的代码“防弹”
> 捕捉真正的需求
> 无情而有效的测试
> 无处不在的自动化

这些具体问题的解法，就是第一层面。然后逐步上升到了第二层面，它的方法体系，一篇书评中将其称为本书的“哲学”：

> 本书的哲学将渗入你的意识，并与你自己的哲学交融在一起。它不鼓吹，它只是讲述什么可行，但在讲述中却又有更多的东西到临，我们有时称之为“无名的品质（Quality without a name）”。

当这些问题倒下而你还在程序员的阵地上时，想必你就会让人感受到那种“无名的品质”，此时你也就在当前维度走到了“精通”的门前。在第一层面上你达成了品质和效率，然后在第二个层面上，抽象出了当前维度的“解”，那么就可以通过“启发式”方法应用到其他维度，具备了向其他维度扩展的基础，从一个细分领域到另一个关联领域的“精通”能力。

所谓“**启发式**”方法，就是“在某个视角里，使用这个规则能够得到一个解，那么你受此启发，也许可以把这个规则用在别的问题上，得到别的解”，而规则就是你在一个维度里抽象出来的方法论体系。

当你感觉在技术维度进境迟滞时，可以尝试扩展到英语维度去，接触一手的技术论文或资料，说不定就能获得启发，找到新的技术维度进境之路。作为多年的程序员，已经形成了用工程师思维分析和求解问题。抽象出来看，程序员都是对问题领域进行建模，然后再用代码实现求得一个“概率解”。

编程实现得到的难道不是一个确定的系统或服务吗？为什么是“概率解”？系统或服务是确定的，但解决的问题，如：需求满足率、服务可靠性，却是概率的。每完成一个系统版本的发布，到底多大程度地满足了用户需求，其实是一个概率，这个概率可以通过用户反馈得到一个大概的感知，但你几乎不会知道一个确定的值。而可靠性，相对来说会更可量化，比如在 99.9% ～ 99.99% 之间波动，但也不会是确定的百分百。

工程师的这个求解模型，可以转移应用到其他很多与你息息相关的工作生活领域，比如投资理财，把钱存银行定期赚钱的概率解无限接近百分百；但买基金、买股票的概率解对大部分人来说就完全靠赌和猜了，因为缺乏一个合适的模型去求解，而对领域建模应该是程序员的强项，也是

可迁移扩展到其他维度的能力。

即使是学习成长本身，也可以用工程模型来求解。这时你的学习维度就需要扩展一下，不仅仅局限于你当前的专业领域，还可以了解点神经科学，认知心理学之类的，并配合自己的现实情况、作息习惯，去建立你的学习模型，获得最佳学习效果。而学习效果，也是一个“概率解”。虽然你不能知道确切的值，但我想你肯定能感觉出不同模型求解的效果好坏。

简言之，**多维的路径，其实是从一个核心基础维度去扩散开的**。

最后，我们总结下，在求解成长的最优路径时，视角的不同，对求解的难度差别巨大。我分享了我的三个视角：**定位，时间视角；自省，自我视角；多维，空间视角**。通过三个不同的视角，探讨了关于“我”与所在现实的时空关系，从中尝试提炼出一种方法，用于探索最适合自己的成长路径。

成长最优路径，求的依然是一个概率解，只是我感觉通过这三个视角去求解，也许概率会更高。

你不妨将这个三维度，套在自己身上，感受一下呢。
