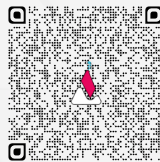


# 开发消费型计算设计师的十五项原则



## 1. 工具并非中立

每一款软件都嵌入了关于工作应该如何完成的假设。当你不加批判地接受一个工具时,你就接受了这些假设——包括关于加班和生产力的有害观念。当你构建自己的工具时,你编码了自己的价值观:效率、人性、对时间的尊重。

## 2. 设计师与工具制造者之间的界限是人为的

这种分离在工业时代是有意义的,因为那时工具需要工厂。但在计算时代,当工具需要思考时,这种分离就没有意义了。使你成为一名优秀设计师的逻辑思维,同样使你有能力构建工具。

## 3. 构建工具就是设计研究

当你构建一个工具时,你被迫将关于问题的思考形式化。这种清晰性使你成为更好的设计师,即使不使用该工具。你需要在成为问题解决者之前先成为问题制造者——深入理解,然后优雅地解决。

## 4. 从问题出发,而非从工具出发

不要问“这个软件能做什么?”而要问“这个问题需要什么?”然后构建(或寻找,或修改)能够满足这一需求的工具。工具服务于愿景,而非相反。

## 5. 自动化苦工,而非思考

构建能够消除繁琐、重复、令人沮丧的工作的工具,这样你就可以专注于创造性、战略性、有意义的思考。不要构建替你思考的工具——要构建给你时间思考的工具。

## 6. 让你的工具可共享

知识在分享中成长。你的自定义组件、脚本和工作流程是送给社区的礼物。分享它们。记录它们。让他人你的工作基础上继续构建。使用共享许可证(就像Ladybug Tools那样)来鼓励这一点。

## 7. 接受不完美的工具,而非完美的工作流程

一个能工作80%并为你节省数小时的工具,胜过一个100%完美但需要数天且令人精疲力竭的手动流程。完美是已完成的敌人。在实践中迭代,而非在理论中。

## 8. 学到足以成为危险人物的程度

你不需要计算机科学博士学位来构建工具。学习足够的Python、C#或JavaScript来解决你的即时问题。深度随着实践而来。我在工程学中用MATLAB和C++学习算法,但重要的是逻辑结构——计算思维,而非语法。

## 9. 迭代式构建,而非纪念碑式构建

不要试图从第一天就构建完美的工具。构建能解决今天问题的东西。明天改进它。让它随着你的理解而演进。(作者注:Ambrosinus工具包是一个组件一个组件、一个问题一个问题地成长起来的)。

## 10. 工具就是信息

当你分享一个工具时,你不仅仅是在分享代码。你是在分享一种思考问题的方式。你通过演示来教学。工具就是教学法——它展示了你如何构建思想。

## 11. 领导力必须随技术演进

一个有效的管理者必须知道如何识别和重视每个团队成员的独特性。不存在通用的方法。每个专业人士都有自己表达才能的方式、自己的特定能力、自己解读项目的钥匙。领导者的真正能力在于知道如何将这不同的乐器调和成和谐的设计交响乐。

## 12. 坚决保护人类时间

时间是你永远无法找回的唯一资源。节省时间的工具不仅关乎效率——它们关乎尊严。它们关乎与家人共度的夜晚、休息的周末、反思的清晨。不要让为工作牺牲生活变得正常化。

## 13. 结合学科以获得创造力

组合创造力来自与许多人和许多操作环境的互动。工程+建筑。编程+设计。环境科学+参数化工具。人工智能+人类判断。创新存在于交叉点。

## 14. 在人工智能时代保持人性

随着人工智能的兴起,一些观察可能会变得模糊或被遗忘,而另一些可能会得到验证。在这个文化不确定的时期,记住人的要素至关重要。建筑师和工程师被认为是AEC行业技术官僚化中最后的知识分子,能够提供超越生产力的反思、观察和方向。

## 15. 培养热情和好奇心

热情是将工作转化为使命的内在火焰,它让你在黎明时分起床去追求别人未曾想象的解决方案。好奇心是低语“如果.....会怎样?”和“为什么不?”的火花——驱使你探索未知领域并在遥远的想法之间找到意想不到的连接的引擎。当热情和好奇心融合时,它们创造了非凡的协同效应。热情提供了追求好奇心所提出问题的决心,而好奇心则不断用新的挑战 and 视角滋养热情。它们共同使你能够持续成长、重塑自我,并留下重要的印记——在你的工作室、你的公司、你的学科或世界上。这些品质没有年龄限制。它们是你终生培养的礼物,将每一天转化为发现的机会,将每一个挑战转化为成长的契机。