檢測時機

昨日，我着手创建一个 Shadowsocks Outline 的自动配置工具，旨在将其转化为一个 Python 项目供他人使用。我开发了一个脚本，通过解码 `ssconfig` 文件中的 Shadowsocks URL，更新 `config.yaml` 文件以包含 Shadowsocks 代理配置。此外，我还创建了另一个脚本，利用 `gsutil` 将客户端的订阅文件上传至 Google 云存储。

我借助了 AI 代码编辑器 Windsurf 来辅助编程。然而，它在处理 Python 单元测试中的模拟依赖时显得力不从心。

回想起尹航分享的测试经验，我记起了他在 Google 的经历，那时他负责 Python 解释器的工作，并为公司代码建立索引以实现搜索功能。他的同事们坚持要求编写测试，这让他感到烦恼。他认为编写优雅的代码比测试更为重要，而同事们只理解了表面，未能触及精髓。

我意识到自己的错误；AI 并未指出这一点。我应当确保库的核心代码稳固后，再专注于测试。这一原则同样适用于概念验证项目。在之前的工作中，比如启动微服务时，应在微服务具备一些 API 或功能后再编写测试。

如果 Windsurf 能妥善处理测试部分，我就不会有此抱怨。但这里涉及两个不同的问题：实施测试的时机与编写测试的正确方法。目前，我们聚焦于前者。这些问题在一定程度上是相互关联的。如果 AI 代码编辑器或人类觉得编写测试代码轻而易举，那么测试的时机可能就显得不那么重要了。然而，编写测试所需努力与编写主代码相当，因此时机选择变得至关重要。

从协作的角度看，测试方法可以有所不同。对于个人项目，我可能会在编写大量代码后再创建测试。但在团队合作中，通常最好为每个代码片段或功能编写测试。但这并非绝对，它取决于团队的协作方式。更准确的说法是，应为团队成员间共享的代码编写测试。目标是确保代码质量，因此在交付代码前，每位团队成员可自由选择测试时机。

在之前的一次工作经历中，我与另外三位后端工程师合作开发一个耗时半年的功能。从测试的角度来看，本文讨论的要点或许能解释为何当时开发进度缓慢。

从协作立场出发，负责主代码的人员也应负责相关测试。任务应尽可能少地交织，每位团队成员职责明确且独立。

回到测试话题，AI 代码编辑器同样缺乏此类优化，显示出改进的空间。这一原则不仅限于软件工程，也适用于硬件及其他领域。测试是一种优化形式，正如俗语所说，“过早优化是万恶之源”。

牢记工作的主要目标至关重要。虽然流程和程序不可避免，但我们必须铭记什么才是真正重要的。

参考资料：

- 测试驱动开发，尹航
- 测试的逻辑，尹航