高效团队最佳实践系列

撰文 / 莫映

条款 2：为小型团队明确设立质量跟踪和产品集成角色，并实行轮值制度

对于一个 5－10 人的小型开发团队——正如我所在的团队——也许每个人都曾经或者正在承担着繁重的开发任务，但这不足以成为可以忽略质量跟踪与产品集成的充分理由。在软件过程中，质量跟踪与产品集成的角色是不可以被轻视的。前者负责全程监督并持续跟踪软件产品的质量状况，确保软件质量不出现问题；而后者则负责将软件各组分集成并统一于一个全局视图。

实践故事

我们不打算设立专职人员，因为高质量专业化的独立测试是中小规模团队所无法承担的，沟通不利和效果不彰的先例屡有所闻。我们也不打算刻意强调过程文档化，包括我在内，过重的文档往往让人产生抵触情绪。并且，过程控制的复杂性也无益于小型团队对快速变化的响应能力。

为了在可调配资源与软件质量之间取得平衡，我们没有对质量跟踪和产品集成做严格区分，而是将其作为一个角色，同时强调其职责的明确性，以及沟通的充分性。

在我们的团队里，质量跟踪人员的日常事务包括：

- 根据软件的规格说明对其进行集成测试，并负责将 Bug 提交至 Issue tracking 系统。随后，一封邮件将会由 Issue tracking 系统自动发送给相关责任人。这里的软件规格说明，除了简明的专有文档外，更细粒度的软件特性，事先都已经转化为 Issue tracking 中的 New Feature 或者是 Improvement 了。
- 监控和跟踪由持续集成平台所反映的有关产品集成与构建的问题。持续集成平台每天午夜会有一次完整的自动构建，并生成相应的测试报告和构建日志，构建结果将会作为邮件发送给全部组员。质量跟踪人员在次日早上会根据构建日志的记录情况，就具体问题寻找相关人员进行沟通。
- 我们的质量跟踪人员还需要负责搭建和维护软件产品的相关集成环境和测试环境，包括对自动化脚本的维护工作。充分利用自动化脚本工具（比如：Ant、Maven），可以简化许多与质量跟踪、产品集成相关的工作，比如：自动准备测试数据，自动运行测试用例（包括单元测试和集成测试），根据需要自动发布质量报告，自动生成可供演示的运行系统，以及多个子项目的并行管理，等等。

另一项值得一提的辅助实践是轮值制度：在团队中每隔一定时间，我们将会对质量跟踪的角色做一次轮换，通常是 1 至 2 个月。根据迭代周期的具体情况，我们还会要求团队成员不定期的轮换进行集成测试。因而，前任者还负有一项将“实践经验”传授给继任者的附加职责。

道理分析

首先，我们认为保证软件产品质量是开发人员的本职工作，因而由开发人员担当质量跟踪的角色是可以理解的。并且，通过测试驱动、持续集成等相关措施的保障，我们可以利用较低的成本，可控而稳定的达到可见的软件质量水平。

其次，开发人员需要有必要的“打扰”，否则他们未必了解自己所负责的模块中存有缺陷和隐患，也未必意识到自己在某项任务上花费的时间是否对开发进度构成风险。成员天生的“惰性”是需要坦然承认的，这没什么好避讳的。相反，在接受这一事实之后，我们就有理由将该项因素充分考虑，并制订出更为务实的措施。所以，我们需要有一个对软件质量和开发进展有全局认识的人，他会视情况不厌其烦的“打扰”其他成员，而当其他成员就项目当前状况提出疑问时，他也应该马上能够做出反馈。

轮值的目的在于培养和复制共同经验，使团队成员在质量跟踪方面有更为深刻的切实体认。在实践过程中，我们时常发现，开发人员在负责局部模块时，主观上总会缺乏对软件整体视图，或他人负责模块的了解和认识的热情。这对于一个小型敏捷团队而言是十分致命的，而轮值可以为团队成员们习惯于从全局角度把握软件本身以及开发过程创造契机。实际上，轮值制度一定程度上还受到了极限编程思想的启发，这看起来很符合“共同所有权（collective ownership）”的精神。

最后一点，轮值制度也可以避免因为人员变更而导致软件过程失去控制的危险。

一段时间之前，我将持续集成的维护工作转交给了继任者。就在几天前的早上，当我打开邮箱收到一封来自持续集成服务器的“build failure”报告之后没过多久，新的负责人正在关切的向另一位同事询问 failure 的原因……半个小时后，我再次收到了来自服务器的自动邮件，不过这次是一个让人欣喜的结果。

注意事项

不要以为集成环境的搭建与维护是轻而易举的。根据我们在上一次开发周期中的统计数据显示，在质量跟踪与产品集成上的精力投入大约花去了单人有效工作时间总和的 2/3，而这其中有一半的时间是花在持续集成平台的搭建与维护上的（包括自动脚本的维护）。但是，这样的投入是值得的，并且随着团队成员对工具与环境的逐步熟识，所花成本将会相应递减。

其他

虽然工具的使用并非主导因素，但是我也很乐意与大家一起分享我们目前围绕质量跟踪与软件集成所使用的相关工具。以下是一个工具清单，仅供参考。

工 具	说 明
Jira	Issue tracking 系统
Anthill Pro	持续集成平台

Ant	自动构建脚本
Cvs	版本控制与管理
Wiki/Blog	统一的文档与内容发布

最后要强调的一点是，这里所谓的“最佳实践”并非是彼此孤立的“金科玉律”，正如极限编程所倡导的，应该充分相信其“加和效应”。我们的团队也依然处在持续改进的过程之中，远非理想状态。但是在这里，并不会有管理制度自上而下的全盘性贯彻或指派，因为我们相信，这很容易流于形式，且无法长久。我们所选择的，是一条循序渐进的道路：在某些适度实践得到巩固并保持良好延续之后，在成员们切实体会到某些关联实践的必要性之后，再依次引入那些关联性实践。一个典型的例子是：当我们意识到小版本发布的反馈情况，对过程控制有着十分重要的意义时，我们引入了持续集成，而当我们深入实践持续集成的时候，我们又逐步意识到，为了使持续集成更有价值，需要加强单元测试的力度。我很高兴的看到，目前大家编写单元测试的自觉性有了显著的提高。