为什么 XP

作者：Robert C. Martin（President, Object Mentor, Inc.）
翻译：lijin_bo[AKA]

为什么 Object Mentor 公司对 XP（eXtreme Programming：极限编程）情有独钟？因为，我们专注于如下两个基本理念：

1 开发软件要迅速
2 开发软件要正确

经验告诉我们，要同时实现这两个相互冲突的目标，一个核心的准则是：采取简练的、结构驱动的开发流程，如：XP。

Object Mentor 注意到：工业界正越来越倾向于采取重量级的开发流程。很显然，那些采取这种流程的公司和项目组认为：他们的早期软件中的问题与不完备的开发流程有关。也许这是对的。然而，不完备不等于“太小”。

在我们看来，多数流程，不论庞大与否，都缺少对软件架构的足够重视。可以相信，如果采取专注于软件架构的轻量级开发流程，而不是那些重量级流程，那些公司可以做得更好。我们也相信，这样的流程能够带来更快的市场推出，更高的软件质量，和更富有激情的工程师团队。

XP 是一种十分注重软件架构的轻量级开发流程。它并不缺少时间规划；实际上，XP 在时间估计与规划上使用了大量的项目管理技术。但是，XP 的核心是开发有强壮的内部结构的软件；使用 XP，开发软件容易，同时软件也易于修改，易于扩展。即使用于相对较小的项目，也能够节省很多的费用和时间。

工程师们的时间是项目经理所拥有的最重要资源。在重量级的开发流程中，许多时间被无关紧要的工作浪费了。通过消除不必要的会议、文档和讨论等，XP 引导项目组把时间花在应该的地方：软件开发。

以逐渐细化的方式确定工程实现规范，关注越来越多的细节，制造和阅读近乎无穷无尽的文档，这是重量级流程的典型特征。与此不同，使用 XP，工程师们能够协作开发功能不断增加的软件，每一次功能增加都是可以演示的。股东们可以看到最近开发的软件；每过一周左右，他们就可以使用带有新特征的新版软件，并对此发表评论。

在这种功能增长中，XP 并未摈弃分析和设计。实际上，分析和设计只是 XP 附带提供的；只不过不必制造许多外在的分析设计文档而已。在 XP 中，分析文档做得尽可能简单，它使用描述性的自然语言，由股东随时审查。设计文档则内置于代码中，由工程师维护；这样，代码本身就是设计文档。

听起来这有些离经叛道；但是，XP 极其重视代码结构。如果代码本身不能作为设计文档，那么，它就必须不断修改，直到合格为止。在 XP 中，模糊代码、重复代码、需要大量注释才可以了解的代码，都是不合格的。

这是否意味着 XP 不再需要诸如 UML 之类的符号系统了呢？不全是。XP 本身并没有集成这类符号系统，但是，工程师们可以使用它来进行设计方面的交流，以此制定协作开发计划。这正是 UML 所擅长的。

简短的说，Object Mentor 投入 XP 的应用是因为：我们相信，它正确的把握了软件开发的过程与核心，

它可以使软件做得既快又对！

自由、协作、创造 — 为了明天
“来自大雪山的大雁阿卡”

更多精彩文章，请访问：http://www.AKA.org.cn **精彩文章 栏目**
本文如有翻译错误或不妥，请 Email **至** AKAMagazine@yahoo.com

附件：英文原文

Why XP?

Robert C. Martin
President, Object Mentor, Inc.

Why is Object Mentor interested in XP (eXtreme Programming)? Because we are dedicated to two fundamental principles:
1. Getting the software done quickly.
2. Getting the software done right.

In our experience, one of the key ingredients in the formula for meeting those two conflicting goals is to employ a small, architecture driven, process -- like XP.

We, at Object Mentor, have watched with growing concern as the industry has turned further and further towards heavyweight processes. Apparently those companies and project teams that are deploying these heavyweight processes are under the opinion that their previous software problems have to do with insufficient process. This may be so. However, insufficient is not a synonym for "too small".

In our opinion, the element that is lacking from most processes, heavyweight or otherwise, is a strong emphasis on software architecture. We believe that those companies who are currently deploying heavyweight processes would do better to employ lightweight processes that are focused on producing software with a strong architecture. We believe that such processes result in faster time to market, higher quality software, and highly motivated engineering teams.

XP is a lightweight process that has a very strong architecture focus. It is not divorced from schedule; indeed, it places a lot of emphasis on project management techniques for estimation and schedule management. But its primary concern is to produce software that has strong internal structure. As a result, the software remains easy to modify, easy to extend, and easy to develop. Even for relatively short projects, this leads to significant savings in the cost and time of development.

The most important resource that any engineering manager has is the time of his engineers. Heavyweight processes consume that time with work that is tangential to the project. XP directs and focuses that time onto the software, where it belongs. XP does this by eliminating unnecessary meetings, documents, reviews, etc.

Rather than having engineers produce and review a nearly endless stream of documents that specify the project to be built in gradual increments of ever increasing detail, XP has the engineers work together to produce software in a steady stream of demonstrable increments. Stakeholders see the software come to life before their eyes. Every week or so a new release with more features is available for them to use and comment upon.

But XP does not abandon analysis and design while producing these increments. Indeed, XP puts a premium on both analysis and design. However, it does this without producing lots of *external* analysis and design documentation. The analysis documentation is kept as simple as possible, in the form of user stories written in plain language and continuously reviewed by the stakeholders. The design documentation is maintained by the engineers within the code. Indeed, the code itself *is* the design documentation.

This may seem heretical, however in XP the structure of the code is considered to be of extreme importance. Code that does not qualify as its own design documentation is refactored until it does qualify. Obscure code is not tolerated. Duplicate code is not tolerated. Code that requires massive comments to understand is not tolerated.

Does this mean that XP abandons the use of notations like UML? Not exactly. XP does not incorporate notations into itself. However, it allows such notations to be used by the engineers as a means to communicate to one another about the design; and to make plans for coordinating the development. Just what UML is best at.

In short, Object Mentor is committed to the use of XP because we feel it puts the right emphasis on the activities and artifacts of software development in order to get software done quickly, and done right.