原文链接：<https://hbswk.hbs.edu/item/the-simple-economics-of-open-source>

原文文本放在最后

翻译：

开源的简单经济学

作者：Martha Lagace

是什么促使成千上万的计算机程序员，甚至是雇用他们的公司，来与世界分享他们的代码？初看，所谓的“开源”软件的日益普及似乎没有多大的经济意义。但根据哈佛商学院教授 Josh Lerner和他的同事Jean Tirole的研究，经济学实际上可能有助于解释为什么开源能像它一样有效。

为什么有人会花时间免费编写复杂的软件程序呢？

这是个好问题，激起了许多经济学家的好奇心，如果可以的话，他们想知道在技术领域蓬勃发展的“开源”运动背后有什么好处。

毕竟，从表面上看，这种情况很像经济无政府状态。当成千上万有天赋的的程序员，甚至许多商业公司，花费大量时间编写和共享计算机源代码时，市场力量在哪里：这种活动显然没有给参与其中的个人和公司带来任何回报、或者说没有回报？

它是否像一些媒体报道所暗示的那样，纯粹是由程序员的知识和热情驱动，也许再加上分享和传播知识以造福人类的崇高愿望？

哈佛商学院教Josh Lerner和他在图卢兹大和麻省理工学的经济学家的同事Jean Tirole说，没那么快。在他们的新论文《开源的简单经济学》中，Lerner和Tirole指出，程序员利他主义的理想主义概念只能到此为止。毕竟，他们认为，慷慨和知识承载并不是其他行业的真正指导因素，那为什么它们会在计算机领域占据主导地位呢？

相反，他们认为，在开源方面工作会给开发人员和公司带来具体的、有形的和非常有利的经济利益：这些好处是明智的，可能相当有利可图的，而且一句话，很简单。利他主义只是一个很好的副产品。

寻找驱动力

开源现象植根于软件开发中共享和合作的悠久传统，但Lerner和Tirole的研究集中在三个特殊案例上：Apache、Perl和Sendmail的案例。除了翻阅印刷的采访、材料、与开源开发中的关键参与者进行面对面的讨论外，他们还询问了知识渊博的观察者。

Lerner和Tirole学到的东西使他们提出了一些方法，当人们从事开源项目而不是“封闭”项目时，程序员普遍支持的动机可能会有所不同。

他们写道，经济理论告诉我们，当程序员从工作中获得净收益时，他们就会参与一个项目，净收益基于即时和延迟的回报。即时奖励包括金钱补偿和修复错误或为自己的利益定制程序的机会。

延迟奖励，即Lerner和Tirole所说的“信号激励”，包括“职业关注激励”，指的是未来的工作机会、商业开源公司的股份或未来获得风险投资的机会，以及“自我满足激励”，主要是渴望获得同行认可。尽管在某些方面有所不同，但当程序员想要让那些他们想要留下深刻印象的人（同事、风险投资家、整个就业市场）看到时，两者都会变得更强大。

凭借即时回报，商业项目在资金方面具有优势，因为代码的专有性质会产生收入，使公司有可能用薪水奖励程序员。但开源项目具有商业项目无法比拟的两个优势。一个是“校友效应”：程序员已经习惯了在学校和大学时就使用开放代码，在那里它是免费提供的；他们能够利用自己已经拥有的知识进行构建。另一个则是，程序员欢迎开源提供的机会，他们可以自定义和消除项目错误，供个人使用或让他们在公司的工作更轻松。

加强信号激励

但Lerner和Tirole发现，开源的真正优势在于延迟或信号激励，其中程序员贡献的可见性最重要。

开源项目可以更好地衡量个人绩效。在一个商业创建的程序中，局外人无法真正分辨谁做了什么。开源则不同。正如Lerner和Tirole所写的那样，“局外人不仅能够看到每个人的贡献是什么，以及其贡献是否‘起作用‘，还能够看到任务是否困难，问题是否以巧妙的方式解决，代码是否对未来的其他编程任务有用”，诸如此类。

在开源中，程序员是他或她自己的老板，可以对任务的成败承担全部责任。相比之下，典型商业项目中的程序员需要与他们的主管合作（或围绕他们）；个人贡献更难去衡量。

最后，在开源中，人们在从一个项目转移到另一个项目时具有更大的灵活性，在过程中积累知识和“工具”。相比之下，在商业公司中，人们受到特定于该公司的专有代码的限制。所以从某种意义上说，当他们换工作时，他们必须重新开始。

在他们的工作论文中，Lerner和Tirole还指出，开源人士可以将他们的项目用作“入口”。例如，一所小学院的系统管理员（他可能是开源的用户，也可能是开源的贡献者）可以向许多能够影响她未来职业生涯的人“发出”她的才能：如同事、潜在雇主，尤其是风险投资家。

Lerner和Tirole发现，风险投资的吸引力也很强。开源工作可能是未来风险投资的重要垫脚石。例如，开源环境使Sun、Netscape和Red Hat的创始人能够向其他人展示他们的构成。

**公司纷纷加入**

Lerner和Tirole说，商业公司并非没有注意到开源项目的成功。他们从中获取利益的策略通常是两种策略之一。

在他们所谓的 “反应式” 策略中，商业公司试图将付费服务和产品捆绑到开源程序中以填补空白。这些服务和产品要么根本不由开源提供，要么没有得到非常有效的处理。“该公司希望......在互补的细分市场上提高利润。“Lerner和Tirole写道。

在第二种策略中，公司通过发布一些自己的专有代码来接受开源运动，希望这将通过新的合作方式在未来带来更大的价值。正如Lerner和Tirole所解释的那样，“这类似于放弃剃须刀（已发布的代码）以销售更多的剃须刀片。

他们指出，这种开放和封闭源代码的混合并非没有风险。用该领域的术语来说，当一个不道德的程序员修改一个模块，然后有效地强加一个专有的新平台，从而带走原始程序的一些主要好处时，开源项目很容易被 “劫持”。

Lerner和Tirole还将其与学术界进行了类比，他们写道，如果程序员过于专注于令人兴奋的短期商业目标，商业利益很容易将创造力和智力探索排除在外。

**未来存在的问题**

Lerner和Tirole说，开源运动给经济学家留下了几个未来需要考虑的问题。例如，将项目分解为多个模块将如何帮助或损害开源？开源项目的成功似乎取决于将项目分解为不同板块的能力；然而，随着项目超越他们Unix的起源，新兴语言是否会继续适应这种模块化？

Lerner和Tirole还想知道开源项目是否能够处理这么多跟风的贡献者。项目负责人将如何筛选所有提交的内容，其中许多提交的内容价值一般或可以忽略不计？

最后，开源项目能否预期比商业项目存在更长的时间？Lerner和Tirole写道，在这个问题上，陪审团仍然没有定论。由于开源代码是免费提供的，因此只要人们被其固有的挑战所吸引，程序就可以存在。但是，像任何其他领域一样，开源领域也突然风靡一时，涌向高知名度项目的开发人员，他们为了获得上述知名度、未来获得风险投资等，很可能会过早放弃那些有价值但不那么光彩的项目。

Lerner和Tirole预测：“我们自信地回答这些问题和相关问题的能力很可能随着开源运动本身的成长和发展而提高。”与此同时，两位教授大胆地希望这样的问题能够激励和激励其他研究人员自己研究这些问题，并以真正的开源方式分享他们的建议。

原文：

The Simple Economics of Open Source

by Martha Lagace

What motivates thousands of computer programmers-and even the companies that employ them-to share their code with the world? The growing use of so-called "open source" software may not seem, at first glance, to make much economic sense. But according to research by HBS Professor Josh Lerner and his colleague Jean Tirole, economics may actually help explain why open source works as well as it does.

Why in the world would anyone take the time to write complicated software programs for free?

It's a good question, one that has piqued the curiosity of a number of economists, who wonder what benefits, if any, lie behind the burgeoning "open source" movement in technology.

After all, outwardly the situation smells of economic anarchy. Where are the market forces, when thousands of talented programmers—and even many commercial firms—spend inordinate amounts of time writing and sharing computer source code: an activity that apparently gives the individuals and companies involved no pay-off, no reward?

Could it be driven, as some media reports have admiringly suggested, purely by intellectual fervor on the part of programmers, perhaps coupled with a noble desire to share and dispense knowledge to benefit mankind?

Not so fast, say HBS Professor [Josh Lerner](https://hbswk.hbs.edu/faculty/jlerner.html) and his colleague Jean Tirole, an economist at the University of Toulouse and the Massachusetts Institute of Technology. In their new working paper, "The Simple Economics of Open Source," Lerner and Tirole make the case that an idealistic notion of programmer altruism only goes so far. After all, the pair argues, generosity and knowledge-bearing have not really been guiding factors in other industries: so why would they dominate the computer field?

Instead, they suggest, laboring on open source brings developers and companies specific, tangible and very favorable economic benefits: benefits that are sensible, potentially quite lucrative and, in a word, simple. Altruism is just a nice by-product.

Looking For Drivers

The phenomenon of open source has roots in a long tradition of sharing and cooperation in software development (see "[A Long Tradition](https://hbswk.hbs.edu/item/the-simple-economics-of-open-source#1)"), but Lerner and Tirole's research focused on three particular cases: those of Apache, Perl and Sendmail (see "[The Fathers of Invention](https://hbswk.hbs.edu/item/the-simple-economics-of-open-source#2)"). In addition to wading through printed interviews and materials, and conducting face-to-face discussions with key players in the development of open source, they also queried knowledgeable observers.

What Lerner and Tirole learned has led them to suggest ways that the commonly espoused motivations of programmers might be different when people work on open source projects as opposed to "closed" projects.

Economic theory, they write, tells us that programmers participate in a project when they derive a net benefit from the work, with net benefit based on both immediate and delayed rewards. Immediate rewards include monetary compensation, as well as the opportunity to fix a bug or customize a program for their own benefit.

Delayed rewards, what Lerner and Tirole call the "signaling incentive," include the "career concern incentive" which refers to future job offers, shares in commercial open source-based companies or future access to venture capital, and the "ego gratification incentive," focused on a desire for peer recognition. Though different in some regards, both have been shown to be stronger when the work is visible to people the programmer wants to impress (colleagues, venture capitalists, the overall job market).

With immediate rewards, commercial projects have an edge as far as money goes—the proprietary nature of the code generates income, making it possible for firms to reward programmers with salaries. But open source projects carry two advantages that commercial projects can't match. One is the "alumni effect": programmers are already used to working with the open code from their time in schools and universities, where it was freely available; they are able to build on knowledge they already possess. And, two, programmers welcome the opportunity, made possible by open source, to customize and de-bug projects, either for personal use or to make their job easier at their firm.

Strengthening The Signaling Incentive

But the real advantage of open source, Lerner and Tirole discovered, is in the delayed or signaling incentives, where the visibility of the programmer's contribution counts most.

Open source projects measure individual performance better. In a commercially created program, outsiders can't really tell who did what. Open source is different. As Lerner and Tirole write, "Outsiders are able to see not only what the contribution of each individual was and whether that component 'worked,' but also whether the task was hard, if the problem was addressed in a clever way, whether the code can be useful for other programming tasks in the future," and so on.

In open source, a programmer is his or her own boss and can take full responsibility for the success or failure of a task. Programmers in typical commercial projects, by contrast, need to work with (or around) their supervisor; the individual contribution is harder to measure.

Finally, in open source people have greater flexibility when moving from project to project, building up knowledge and "tools" as they go. By contrast, in commercial firms people are restricted by proprietary code specific to that firm. So in a sense they have to start all over again when they switch jobs.

In their working paper, Lerner and Tirole also point out that people in open source can use their projects as a "port of entry." For example, a systems administrator at a small college (who might be a user of open source as well as a contributor to it) can "signal" her talent to many people in a position to influence her future career: colleagues, prospective employers and, especially, venture capitalists.

The venture capital attraction is also a strong one, Lerner and Tirole found. Open source work may be a great stepping stone to future venture capital. The open source environment made it possible, for instance, for the founders of Sun, Netscape and Red Hat, to show other people what they were made of.

Companies Jump Aboard

Commercial firms have not failed to notice the success of open source projects. Their strategies for capturing some of this energy usually fall into one of two strategies, say Lerner and Tirole.

In what they call the "reactive" strategy, commercial firms try to bundle paid services and products onto open source programs, to fill a niche. These services and products are either not provided at all by open source or are not handled very efficiently. "The company expects to … boost its profit on a complementary segment," write Lerner and Tirole.

In the second strategy, companies embrace the open source movement by releasing some of their own proprietary code, in the hopes that this will lead to greater value down the road thanks to new kinds of cooperation. As Lerner and Tirole explain, "This is similar to giving away the razor (the released code) to sell more razor blades."

This mixing of open and closed source code is not without risks, they point out. In the lingo of the field, an open source project can easily be "hijacked" when an unscrupulous programmer modifies a module and then effectively imposes a proprietary new platform, whisking away some prime benefits of the original program.

Lerner and Tirole also draw an analogy to academe, writing that commercial interests can easily preclude creativity and intellectual exploration if programmers become too fixated on exciting, short-term commercial goals.

Puzzles For The Future

The open source movement leaves several questions for economists to contemplate in the future, say Lerner and Tirole. How, for instance, will the breaking of projects into modules help or hurt open source? The success of an open source project seems dependent on the ability to break down the project into distinct components; yet as projects move beyond their Unix origins, will emerging languages continue to accommodate this modularization?

Lerner and Tirole also wonder whether open source projects can handle so many contributors jumping on the bandwagon. How will project leaders sift through all the submissions, many of which are only of fair-to-negligible value?

And finally, can open source projects expect to live longer than commercial ones? The jury remains out on that question too, write Lerner and Tirole. Since open source code is freely available, programs can probably live so long as people are attracted to their inherent challenges. But fads erupt in open source as in any other field, and developers who flock to high-profile projects—for the aforementioned visibility, future access to venture capital, etc.—could well abandon worthy but less glamorous projects before their time.

"Our ability to answer confidently these and related questions," predict Lerner and Tirole, "is likely to increase as the open source movement itself grows and evolves." In the meantime, the two professors venture their hope that such puzzles will inspire and stimulate other researchers to look into the issues themselves, and share their suggestions — in true open source style.