

Cholera is one of those 19th-century ills that, like consumption or gout, at first seems almost quaint, a malady from an age when people suffered from maladies. But in the developing world, the disease is still widespread and can be gruesomely lethal. When cholera strikes an unprepared community, people get violently sick immediately. On day two,

severe dehydration sets in. By day seven, half of a village might be dead.

Since cholera kills by driving fluids from the body, the treatment is to pump liquid back in, as fast as possible. The one proven technology, an intravenous saline drip, has a few drawbacks. An easy-to-use, computer-regulated IV can cost \$2,000 – far too expensive to deploy against a large outbreak. Other systems cost as little as 35 cents, but they're too complicated for unskilled caregivers.

The result: People die unnecessarily.

"It's a health problem, but it's also a design problem," says Timothy Prestero, a onetime Peace Corps volunteer who cofounded a group called Design That Matters. Leading a team of MIT engineering students, Prestero, who has master's degrees in mechanical and oceanographic engineering, focused on the drip chamber and pinch valve controlling the saline flow rate.

But the team needed more medical expertise. So Prestero turned to ThinkCycle, a Web-based industrial-design project that brings together engineers, designers, academics, and professionals from a variety of disciplines. Soon, some physicians and engineers were pitching in – vetting designs and recommending new paths. Within a few months, Prestero's team had turned the suggestions into an ingenious solution. Taking inspiration from a tool called a rotameter used in chemical engineering, the group crafted a new IV system that's intuitive to

use, even for untrained workers. Remarkably, it costs about \$1.25 to manufacture, making it ideal for mass deployment. Prestero is now in talks with a medical devices company; the new IV could be in the field a year from now.

ThinkCycle's collaborative approach is modeled on a method that for more than a decade has been closely associated with software development: open source. It's called that because the collaboration is open to all and the source code is freely shared. Open source harnesses the distributive powers of the Internet, parcels the work out to thousands, and uses their piecework to build a better whole – putting informal networks of volunteer coders in direct competition with big corporations. It works like an ant colony, where the collective intelligence of the network supersedes any single contributor.

Open source, of course, is the magic behind Linux, the operating system that is transforming the software industry. Linux commands a growing share of the server market worldwide and even has Microsoft CEO Steve Ballmer warning of its "competitive challenge for us and for our entire industry." And open source software transcends Linux. Altogether, more than 65,000 collaborative software projects click along at Sourceforge.net, a clearinghouse for the open source community. The success of Linux alone has stunned the business world.

But software is just the beginning. Open

source has spread to other disciplines, from the hard sciences to the liberal arts. Biologists have embraced open source methods in genomics and informatics, building massive databases to genetically sequence *E. coli*, yeast, and other workhorses of lab research. NASA has adopted open source principles as part of its Mars mission, calling on volunteer "clickworkers" to identify millions of craters and help draw a map of the Red Planet. There is open source publishing: With Bruce Perens, who helped define open source software in the '90s, Prentice Hall is publishing a series of computer books open to any use, modification, or redistribution, with readers' improvements considered for succeeding editions. There are library efforts like Project Gutenberg, which has already digitized more than 6,000 books, with hundreds of volunteers typing in, page by page, classics from Shakespeare to Stendhal; at the same time, a related project, Distributed Proofreading, deploys legions of copy editors to make sure the Gutenberg texts are correct. There are open source projects in law and religion. There's even an open source cookbook.

In 2003, the method is proving to be as broadly effective – and, yes, as revolutionary – a means of production as the assembly line was a century ago.

Thomas Goetz (thomas@wiredmag.com) is Wired's articles editor.

THE IDEALS OF OPEN SOURCE

SHARE THE GOAL

Open source projects succeed when a broad group of contributors recognize the same need and agree on how to meet it. Linux gave programmers a way to build a better, leaner operating system; Woochi gives wine lovers an encyclopedia as refined as they are.

SHARE THE WORK

Projects can be broken down into smaller tasks and distributed among armies of volunteers for execution. Tim O'Reilly, whose namesake company runs the Open Source Convention, calls this the "architecture of participation," and it is the irresistible genius of open source, a tool that no corporate model can match for the sheer brainpower it yokes. But architecture demands structure: a review process that screens for the best contributions and avoids the "fork" – that horrible prospect that a project will split into a multitude of side projects.

SHARE THE RESULT

Open source etiquette mandates that the code be available for anyone to tweak and that improvements to the code be shared with all. Substitute *creation* for *code* and the same goes outside of software. Think of it as the triumph of participation by the many over ownership by the few. – T.G.

In the Beginning

Message-ID:
1991Aug25.205708.9541@klaava
.helsinki.fi
From:
torvalds@klaava.helsinki.fi
(Linus Benedict Torvalds)
To: Newsgroups: comp.os.inix
Subject: What would you like to
see most in minix?
Summary: small poll for my new
operating system

Hello everybody out there using minix-I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386 (486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat...

Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus

Thousands of coders, hackers, and developers answered Linus Torvalds' call – and helped him build a robust system that continues to pick up steam. Yet what's amazing about Linux isn't its success in the market. The revolution is in the method, not the result. Open source involves a broad body of collaborators, typically volunteers, whose every contribution builds on those before. Just as important, the product of this collaboration is freely available to all comers. Of course, there are plenty of things that are collaborative and free but aren't really open source (Amazon.com's book reviews, for instance). And many projects aren't widely collaborative, or are somewhat proprietary, yet still in the spirit of open source (such as the music available from Opsound, an online record label). Not to mention that, as with any term newly in vogue, *open source* is often invoked on tenuous grounds. So think of it as a spectrum or – better still – a rising diagonal line on a graph, where openness lies on one axis and collaboration on the other. The higher an effort registers both concepts, the more fully it can be considered open source.

Of course, for all its novelty, open source

isn't new. Dust off your Isaac Newton and you'll recognize the same ideals of sharing scientific methods and results in the late 1600s (dig deeper and you can follow the vein all the way back to Ptolemy, circa AD 150). Or roll up your sleeves and see the same ethic in Amish barn raising, a tradition that dates to the early 18th century. Or read its roots, as many have, in the creation of the Oxford English Dictionary, the 19th-century project where a network of far-flung etymologists built the world's greatest dictionary by mail. Or trace its outline in the Human Genome Project, the distributed gene-mapping effort that began just a year before Torvalds planted the seeds of his OS.

If the ideas behind it are so familiar and simple, why has open source only now become such a powerful force? Two reasons: the rise of the Internet and the excesses of intellectual property. The Internet is open source's great enabler, the communications tool that makes massive decentralized projects possible. Intellectual property, on the other hand, is open source's nemesis: a legal regime that has become so stifling and restrictive that thousands of free-thinking programmers, scientists, designers, engineers, and scholars are desperate to find new ways to create.

We are at a convergent moment, when a philosophy, a strategy, and a technology have aligned to unleash great innovation. Open source is powerful because it's an alternative to the status quo, another way to produce things or solve problems. And in many cases, it's a better way. Better because current methods are not fast enough, not ambitious enough, or don't take advantage of our collective creative potential.

Open source has flourished in software because programming, for all the romance of guerrilla geeks and hacker ethics, is a fairly precise discipline; you're only as good as your code. It's relatively easy to run an open source software project as a meritocracy, a level playing field that encourages participation. But those virtues aren't exclusive to software. Coders, it could be argued, got to open source first only because they were closest to the tool that made it a feasible means of production: the Internet.

The Internet excels at facilitating the exchange of large chunks of information, fast. From distributed computation projects such as SETI@home to file-swapping systems

