# The Emerging Economic Paradigm of Open Source

The economics of Open Source can be explained entirely within the context of conventional open-market economics. Open Source has much stronger ties to the phenomenon of capitalism than many have appreciated.

## Timeline

1981
Joins the New York Institute of Technology, becomes an Unix kernel programmer

1986
Works at Matrix Instruments for 6 months due to non-compete clause

1987 to 1999
Joins Pixar Animation Studios as a senior systems programmer

1987
Initial release of Electric Fence

1994
Became a major Linux developer

1998
Starts the Open Source Initiative with Eric Raymond

1999
Linux Capital Group

2000
Senior strategist for Linux and open source at Hewlett-Packard

2002
Starts Perens LLC

2004

Senior Scientist, George Washington University

## Boards

Software in the Public Interest
Elara Fusion
Open Source Risk Management

## Background

Bruce Perens helped coin the term "Open Source" and wrote the original Open Source definition. He has been instrumental in many open source initiatives, such as the Linux Standard Base.

Bruce is series editor of the Bruce Perens' Open Source Series line of books with Prentice Hall PTR publishers, with 15 titles published so far. The books sell well in retail stores, even though their texts are released under an Open Source license.

He was recently named Senior Scientist for Open Source by the Cyber Security Policy Research Laboratory of George Washington University.

Perens is a member of the board of directors of Open Source Risk Management Inc. and Elara Fusion and a major stockholder of Progeny Linux. He is an elected director of Software in the Public Interest.

## Introduction

Bruce Perens was SDForum's distinguished speaker at PARC on April 21, 2005. Andrew Aitken, founder and managing partner, Olliance Group introduced Bruce as being a founder and driver of the Open Source movement. He observed that Bruce has a tremendous sincerity and focus on Open Source and has consistently focused on what is the next step to take in moving Open Source forward.

Andrew started selling Open Source software, 8 or 9 years ago. In attending many of the same conferences as Bruce, since they both liked to drink beer, they often ended up at the same bars after a conference. When Andrew was in a cab without cash, Bruce handed him $20 without a question, hardly knowing him.

## A Brief History and Background

In Andrew's introduction, he noted that Bruce loaned him $20, without knowing him, when needing to pay a taxicab bill. But in reality, Bruce had seen Andrew at conferences before, and knew he could find him again!

Bruce has done a bit of everything. In being asked to speak tonight, he reminded the audience that another term for distinguished speaker is "old fart."

Bruce's programing career started when he was running a college radio station. Having access to a Xerox Sigma 9 that ran BASIC, he spent six months figuring out how to use it to automatically generate the radio station log. Being particularly good at reading books, his next task was to learn APL on the Sigma 9.

Bruce's involvement with computer graphics started in 1981 when he joined the predecessor to Pixar, the New York Institute of Technology's computer graphics lab as a system operator on its Unix system at $2.10/hour in 1981. They were the first to use computer graphic character animation for story-telling characters.

Having never taken any computer science courses, at the time, there were only two books on Unix. There was a pile of supplemental Unix literature that you had to print out on a Versatec wet toner printer. And there was Dennis and Ritchie's C programming book.

Now the reason that you had to know all this, was that at the time, when your file system got corrupted, you had to use the ADB debugger to fix it. Unix tools like fsck (disk check) didn't exist at that time.

In 1981, a single frame player didn't exist. A frame buffer for a 640 x 480 image cost $70,000. Bruce worked for 6 or 7 years on DEC PDP-11 and VAX computers running Unix. They developed software that let them use a videotape recorder one frame at a time, down to a 30th of a second. This was his first exposure to what you could do in realtime with Unix.

He left the New York Institute of Technology in 1986 and went to Matrix Instruments (now part of Agfa) for six months, before going to Pixar. At the time, there was a non-compete clause that prevented him from going directly to Lucasfilm that owned Pixar at the time.

In 1986, Pixar's primary business was making an image processing computer. George Lucas needed money to make the *Howard the Duck* movie, so he sold Pixar to Steve Jobs. At any rate, the Pixar computer was a SIMD (single instruction, multiple data) computer with a tiled memory architecture. It was implemented using multiple 4-bit AMD 2901 processors. Bruce wrote microcode for the 2nd generation of this hardware.

They used gate arrays for part of Pixar's hardware. The problem was that it took 1 or 2 months to make a gate array and it cost $50,000 each time you made one. They addressed this problem by creating a behavioral model of the gate array that allowed them to simulate it, and get it right the first time.

## Getting Involved with Open Source

Bruce got an email that took him about a half hour to answer a question about software that he had written and made available on an Open Source basis. He got an email back saying that thanks to his advice, the code would be used on the next space shuttle flight.

Being a bit of a loud mouth, Bruce enjoyed talking to the press and became well known. Microsoft started saying some really stupid stuff to the effect that Linux was a cancer, and he responded to their accusations. He started to get invitations to speak at conferences, at desirable locations like Iceland.

Bruce released his Electric Fence program that enabled the Malloc debugger to stop on the exact line of a bug, as Open Source in 1987. The following things happened:

- When it was first used on the HP-UX system, it helped find 30 bugs, forcing an unscheduled bug fix release.

- Bruce got an email from a person in Ireland saying that the program had saved their job.
- It made money for him, when a person emailed him all the documentation for Electric Fence, meaning that he didn't have to spend time documenting it.

These and other events reinforced Bruce's interest and involvement in Open Source, eventually leading to his formation with Eric Raymond of the Open Source Foundation in 1998.

Subsequently at Pixar, he read a statement by Steve Ballmer of Microsoft stating that "Open Source software is a way of licensing software." At that point, Bruce realized that Steve had read his open source definition, and the influence he was starting to have on the industry.

## Open Source Economics

The tremendous economic success of Microsoft with its $40B in revenues and 55,000 employees, frequently causes us to think of software in a vendor centric mode. And it shapes our thinking, causing us to wonder how people make money doing Open Source? We forget that only 30% of software is sold as software. Most software is written by customers.

Most companies make products and provide services, only a small minority manufacture software. Programmers and web designers are an unavoidable cost of doing business for all but software companies

The economic impact of Microsoft or any software company is in providing the tools that enable modern businesses to operate.

Despite widespread dissatisfaction with its products, the lack of competition due to its being a monopoly, gives Microsoft no incentive to improve its products. Open Source software represents the first serious competition to Microsoft in a decade in that it both democratizes and provides economic incentive to keep software competitive.

Open Source software has much stronger ties to capitalism than many realize. The problem with the book, *The Cathedral and the Bazaar*, written in 1998, is that it describes programmer motivation for doing open source as being non-economic in nature. We now know that this is not correct.

The fundamental economic benefit of Open Source is that it enables you to spend your time and money doing something else — it makes you more productive. As business people get involved with Open Source software, they are increasingly coming to appreciate this.

## Differentiating Software

There are two types of software:

- Differentiate
- Non-differentiating

Differentiating software makes your business better than the competition in some way:

- Amazon's recommendation feature suggests books that other people purchased when you buy a book.
- Pixar's internally developed software used to make their films.

Differentiating software provides an advantage over competitors and thus must remain unique and proprietary.

Most software is non-differentiating. While it is essential to your business, it does not make your business any better than your competitors. Two examples are:

- Apache web server
- GNOME desktop

Since non-differentiating software is common to your competitors, making it open source enables them to become your best collaborators. This is why IBM and HP cooperate on many Open Source programs.

## Open Source Marketing

If you ask someone, "would you liked to work in a planned economy?", the answer invariably is, "no". With Open Source, you don't have the situation of large companies like Microsoft, Oracle or SAP telling the entire IT industry what to do.

You can think of Open Source software as being a massively-parallel drunkard's walk filtered by a Darwinistic process. When you have a large collection of people writing software, some of them are going to have good ideas. The market decides what are good ideas and what are not. This process works as well as marketing.

With Open Source software, if there is a group as small as 50 people worldwide who want the same thing, willing to work on it in their spare time, you have sufficient resources to develop large and complex software products. As the product matures and becomes of more interest to people, the number of people contributing to it will grow substantially.

## How Software is Developed

There are four ways in which software is developed:

- Retail
- In-house / contract
- Consortium
- Open Source

**Retail** software accounts for about 30% of all existing software. You might think that it is a very efficient way of distributing the cost of development among many customers.

But in fact, even for Microsoft, in 2004 it spent only 16.8% of its revenues on development. After the middlemen takes their cut, and figuring that 50% of all software ends up as shelfware (it never gets used), only about 5 cents of your software dollar actually went to developing it.

Why is this? Retail software has a very large overhead, since it is very expensive to locate customers, you have to manufacture boxes and pay for shelf space. There is advertising and points to the distributor and retailer.

The other characteristic of retail software is that it requires a mass market so everyone can buy it. The result is that retail software is non-differentiating.

**In-house and contract software** avoid much of retail software's overhead, with the result that you obtain about 50% efficiency on the dollar, and you gain a differentiating advantage. Typically a single customer pays for everything, and takes all the risk. Historically about 50% of these efforts ends up as shelfware, so the net is that you only get about 25 cents on the dollar for your programming.

**Consortiums** or formal corporate collaborations have ended up as colossal failures, as witnessed by the hundreds of millions of dollars wasted by Taligent and Monterey. The fatal flaw of consortium projects is that there's always the handshake with one hand and a dagger in the other — it is never fair for all of the partners.

**Open Source** has its initial development by a single entity. It is released as soon as it does something useful. For example, Gnome initially used the user interface from a paint program, and could be booted and run from a PC.[1]

So what does Open Source software do economically?

- At an very early stage, no one bares the entire cost and risk of development.
- If I'm internally writing software that is non-differentiating, than I'm wasting my money since it isn't giving me a competitive edge. If I'm going to write software that is only for my own use, I want to focus on software that differentiates itself.

This is where Sun went astray as a developer of non-differentiating system software. They were

[1] There are some exceptions, it took Mozilla 4 years before its release as Firefox.

fat and happy making money with hardware margins of 70%. But now that they have to compete with Dell (who acts as a front for Chinese manufacturers) over half a margin point; they are in big trouble.

This is the same problem that HP has. In driving by HP's headquarters on Page Mill on the way to this talk, I shouted out the window, "I told you turkeys that buying Compaq would not be a good idea!"

## Capitalism and Open Source

Steve Ballmer of Microsoft has said that, "Open Source software is a threat to the economy." He must have cut his economics class at Stanford.

At the beginning of the 20th century, there was a large industry devoted to the harvesting, storage, and transportation of ice. With the development of refrigeration, this industry collapsed. Despite the failure of this industry, it was a tremendous boost for the overall economy, since it now worked more efficiently.

Although Open Source is not good news for retail software companies, the overall demand for software will not decrease. Instead retail software programmers will be found working internally at companies like Amazon and EBay, developing software that provides them with differentiating advantage.

By enabling things to be run more efficiently, Open Source reduces the overall cost of software, with the result that more people will be able to afford computers.

From a business standpoint, profitability can be increased by spending less on cost centers such as software. Open Source helps a business maximize the money it spends on differentiating software.

With Open Source software, there is no mandate over how people use a product; it provides a level playing surface for competition among software vendors. This in contrast to Microsoft who uses "network effects" to bias the market.

## What Will Stay Proprietary?

There is proprietary software that will probably not become Open Source. For example, consider TurboTax. There is huge liability if you are wrong, and nobody is going to write it for the love of doing it. While it is quite complex, it is also software that is only used once a year.

*(Bruce joked that although his wife uses Windows, he does NOT undergo ritual purification before touching her PC to run TurboTax once a year when doing taxes!)*

Another example is Synoptics, who provides CAD software for designing integrated circuits. While there is a vigorous open source hardware effort, Synoptics sill has a several year lead over these efforts.

But Synoptics decided to make a key part of its software Open Source, software that enables it to interface to multiple chip foundries.

By giving this software to its competitors, it created a common API and defacto standard. The result is that everyone uses it, and Synoptics no longer has to create unique interfaces for each new foundry.

## Economics of Finding Bugs

There are different motivations and economics for finding and fixing bugs. For Microsoft, outside people are continually looking for bugs so they can write viruses and worms. With Open Source software, people are looking to extend and modify it.

When Borland made Interbase Open Source, nine months later, a major backdoor bug was found. Here was software that had been used to implement airline reservation systems, with a bug that could be used to obtain free flights. This illustrates the advantage of having many eyes looking at code. However, many eyes are not disciplined eyes that will inspect code in a systematic manner — you need a mix of the two approaches.

Programmers have a lot in common with artists. Often the creative work that both artists

and programmers do is not specifically connected with their employment. And like an artist, Open Source programmers have a tremendous pride in their reputation. This is why the Linux kernel and other Open Source software keeps getting better and better.

## The Future of Open Source

Open Source software is enabled by the Internet, that has essentially eliminated the cost of manufacturing and distributing software. When a Star Trek replicator becomes available, design will become more important than manufacturing for other products as well.

The closest thing that we have today in Open Source hardware is the Open Source software radio receiver. It lets you receive a signal between 800 MHz to 3 GHz. Using Python, you can connect various DSP blocks to implement a 802.11 radio, a HDTV decoder, or do radio astronomy.

## Open Source Opportunities

Linuxcare was a $20M failed startup. Its fundamental problem was that it was ahead of its time. Early adopters of Linux had no need for support, they wanted complete control. But the second generation of Open Source users want support and are willing to pay for it. This is the opportunity that SpikeSource is seizing, and it also represents an opportunity for both IBM and HP.

The computer should be your invisible friend. We already have software that enables continuous speech recognition and generation. What is needed are the higher levels of software to make these capabilities useful to us.

Fundamentally, it is no longer sufficient to just know computers, you have to have domain expertise in something else.

## Summary

Using Open Source software enables a company to become more profitable, and provides it with more control over its business. Open Source allows a company to focus its finite software development resources on capabilities that differentiate itself from competitors.

## For More Information

Bruce Perens's paper, *The Emerging Economic Paradigm of Open Source* can be found at: www.perens.com/articles/economic.html

## About the Author

Mark Duncan is a marketing consultant who focuses on emerging technologies, assisting companies in entering new markets and developing new business opportunities. He can be contacted at mark@askmar.com.