

Giving It Away:

A capitalist entrepreneur's view of Open Source

Robert Young, CEO, Lulu.com

Here's an updated, 3rd, edition of an explanation of my view of the open source phenomenon and how to build a company involved in open source software. The 1st edition was originally published in 1997. The second edition was published in 1999 as part of O'Reilly's Open Sources collection of essays: <http://www.oreilly.com/catalog/opensources/>.

This paper is published without any official or unofficial support from my friends at Red Hat, but if they object I suspect this will disappear faster than a magician's rabbit. This version continues to be written in the first person: I, we, our. But please keep in mind it was originally written in 1997. I am not attempting to claim, nor do I deserve credit for all the brilliant work that has made Red Hat into one of the great Internet era success stories. I'm lost in admiration for the work the team at Red Hat has done over the last three years.

My personal focus today is on the success of Lulu, whose opportunity is as large but whose challenges are every bit as daunting as Red Hat's in 1994. But this is not about Lulu except in the exploitive sense of providing more content to Lulu's content exchange.

Stay tuned to www.lulu.com for more on Lulu.

Bob Young, April 2003

Where Did Red Hat Come From?

In the early days of the Linux kernel (1993) Red Hat, formerly known as Red Hat Software, Inc., formerly known as ACC Corp Inc publishers of the PC Unix and Linux Catalog, was a small software distribution company. We offered Unix applications, books, and low-cost CD-ROMs from vendors like Walnut Creek and Infomagic. In addition to conventional Unix offerings, these vendors were beginning to offer a new line: Linux CD-ROMs. The Linux CDs were becoming bestsellers for us. When we'd ask where this Linux stuff was coming from, we'd get answers like, "It's from the programmers according to their skill to the users according to their needs."

If the collapse of the Berlin Wall had taught us anything, it was that socialism alone was not a sustainable economic model. Hopeful slogans aside, human activities did not replicate themselves without a good economic model driving the effort. Linux seemed to lack such a model. We reasoned, therefore, that the whole Linux thing was a big fluke. A fluke that was generating enough cash to keep our little business and a number of other small businesses in the black, but a fluke nonetheless.

However, we found that instead of this bizarre free software (as Open Source was known prior to 1997) effort collapsing, it continued to improve. The number of users continued to grow and the applications they were putting it to were growing in sophistication.

So we began to study free software development more carefully. We spoke to the key developers and the largest users. The more we studied, the more of a solid, albeit unusual, economic model we saw.

This economic model was effective. More importantly, our sales of Linux-based OSes compared to our sales of other Unixes were sufficient to convince us that this was a real technology with a real future. At this point (fall of '94) we were looking for Linux products that we could sell into CompUSA and other leading retail distribution outlets. Marc Ewing and I partnered to create Red Hat Software, Inc. in January of 1995, and the rest of this chapter is devoted to the trials and errors of developing a business plan that was compatible with the bizarre economic model. Bizarre as it was, this model was producing a remarkable OS, providing value to our customers, and providing profit for our shareholders.

At Red Hat, our role is to work with all the development teams across the Internet to take some four hundred software packages and assemble them into a useful operating system. We operate much like a car assembly plant taking parts from many suppliers and building useful products from those parts. Very few people build their own cars. The same with the hundreds of programs that make up the collection of open Source Linux technologies: few people build their own Linux-based OS. Red Hat designs, assembles, and tests the finished operating system product and offers support and services for the users of the Red Hat Linux OS.

The "unique value proposition" of our business plan was, and continues to be, to cater to our customers' need to gain control over the operating system they were using by

delivering the technical benefits of freely-redistributable software (source code and a free license) to technically-oriented OS consumers.

All the other operating systems, such as Windows, MacOS, or AIX-Unix, available at that time were proprietary binary-only products, where the customers had no control over the technology they were building their corporate information systems and networks upon.

How Do You Make Money in Free Software?

That question assumes that it is easy, or at least easier, to make money selling proprietary binary-only software. This is a mistake.

Most software ventures, whether based on free or proprietary software, fail. Given that until very recently all software ventures were of the proprietary binary-only kind, it is therefore safe to say that the IP (Intellectual Property) model of software development and marketing is a very difficult way to make a living. Of course, so was panning for gold during the gold rushes of the 19th century. But when software companies strike it rich they generate a lot of money, just like past gold rushes, so lots of people are willing to assume the risks in order to have an opportunity to “strike gold.”

While making money with Open Source software is a challenge, the challenge is not necessarily greater than with proprietary software. In fact, you make money in Open Source software exactly the same way you do it in proprietary software: by building a great product, marketing it with skill and imagination, looking after your customers, and thereby building a brand that stands for quality and customer service.

Marketing with skill and imagination, particularly in highly competitive markets, requires that you offer solutions to your customers that others cannot or will not match. To that end Open Source is not a liability but a competitive advantage. This development model produces software that is stable, flexible, and highly customizable. So the vendor of Open Source software starts with a quality product. The trick is to devise an effective way to make money delivering the benefits of Open Source software to your clients.

Inventing new economic models is not a trivial task, and the innovations that Red Hat has stumbled upon certainly do not apply to everyone or every product. But there are some principles that should apply to many software ventures, and to many Open Source ventures.

And remember, we’re in the very early days of the deployment and growth of market share for free software. If you aren’t making money today it may be simply because the market for your product is still small. While we are pleased with the growth of the Linux OS, estimates being as high as 10 million users today (1998), you need to remember that there are some 230 million DOS/Windows users.

We Are in the Commodity Product Business

If we do not own intellectual property the way almost all of today's software companies do, and if those companies insist that their most valuable asset is the intellectual property represented by the source code to the software they own, then it is safe to say that Red Hat is not in the Software Business. Red Hat is not licensing intellectual property over which it has ownership. That's not the economic model that will support our customers, staff, and shareholders. So the question became: What business are we in?

The answer was to look around at other industries and try to find one that matched. We wanted an industry where the basic ingredients were free, or at least freely available. We looked at the legal industry; you cannot trademark or patent legal arguments. If a lawyer wins a case in front of the Supreme Court, other lawyers are allowed to use those arguments without requesting permission. In effect, the arguments have become public domain.

We looked at the commodity industries and began to recognize some ideas. All leading companies selling commodity products, including bottled water (Perrier or Evian), the soap business (Tide), or the tomato paste business (Heinz), base their marketing strategies on the building strong brands. These brands must stand for quality, consistency, and reliability. We saw something in the brand management of these commodity products that we thought we could emulate.

Ketchup is nothing more than flavored tomato paste. Something that looks and tastes a lot like Heinz Ketchup can be made in your kitchen sink without so much as bending a copyright rule. It is effectively all freely-redistributable objects: tomatoes, vinegar, salt, and spices. So why don't we, as consumers, make ketchup in our kitchen sink, and how does Heinz have 60% of the ketchup market?

We don't make ketchup because it is cheaper and much more convenient to buy ketchup from Heinz, Hunts or Del Monte than it is to make it. But convenience is only part of the story. Convenience alone would suggest that Heinz, Hunts, and Del Monte share the market equally because they offer roughly equivalent convenience. In fact, Heinz dominates the market.

Heinz does not own 60% of the market because Heinz tastes better. If you go to the Third World and find 100 people who have never tasted ketchup before, you find out two things: one is that people don't actually like tomato ketchup, the other is that they dislike all ketchups equally.

Heinz dominates the ketchup market because they have been able to define the taste of ketchup in the mind of ketchup consumers. Now the Heinz Ketchup brand is so effective that as consumers we think that ketchup that will not come out of the bottle is somehow better than ketchup that pours easily!

This was Red Hat's opportunity: to offer convenience, to offer quality, and most importantly to help define, in the minds of our customers, what an operating system can

be. At Red Hat, if we do a good job of supplying and supporting a consistently high-quality product, we have a great opportunity to establish a brand that Linux OS customers simply prefer.

But how do we reconcile our need to create more Linux users with our need to ensure that those Linux users use Red Hat? We looked at industries where the participants benefit because of, not despite, the activities of other participants.

Drinking water can be had in most industrial countries simply by turning on the nearest tap, so why does Evian sell millions of dollars of French tap water into those markets? It boils down to concerns the water coming from your tap is not to be trusted.

This is the same reason that many people prefer to purchase “Official” Red Hat Linux in a box for \$50 when they could download it for free or buy unofficial CD-ROM copies of Red Hat for as little as \$2. Evian does have the advantage that most of humanity drinks water - we still have to create a lot of Linux consumers in order to have a market to sell our brand into.

On the other hand Evian water is not used to build valuable corporate information systems. The need in major global corporations for support services to reduce the cost of deploying and maintaining the rapidly evolving Open Source software in Red Hat Linux, makes this an exciting business opportunity.

The challenge is to focus on market size, not just market share. When consumer demand for bottled water grows, Evian benefits, even though many of those consumers start with a bottle other than Evian. Red Hat, like Evian, benefits when other Linux suppliers do a great job building a taste for the product. The more Linux users there are overall, the more potential customers Red Hat has for our flavor.

The power of brands translates very effectively into the technology business. We have evidence of this in the Venture Capital investors who continue to invest in Open Source software companies. The one common denominator between all of the investments to date has been that the companies or their products have great name recognition, and are recognized as being quality products. In other words, they have successfully established a brand.

The Strategic Appeal of This Model to the Corporate Computing Industry

Much of brand management comes down to market positioning. Consider the challenges that a new OS faces in trying to gain significant marketshare. The current OS market is crowded, and dominated by definite market favorites from a brilliant marketing organization. Positioning a competing product correctly is crucial to competitive success.

Linux fills this role naturally and extremely well. The primary complaint about the market leader is the control that vendor has over the industry. A new OS must deliver control over the OS platform to its user and not become just another proprietary binary-only OS whose owner would then gain the same dominant market position that consumers are currently complaining about.

Consider that Linux is not really an OS. It has come to describe a whole collection of collection of open-source components much like the term “car” describes an industry better than the thing we drive on the highway. We don’t drive cars- we drive Ford Taurus or Honda Accords. Red Hat is the equivalent of an OS assembly plant of the Free Software operating system industry. Red Hat succeeds when customers perceive themselves not as purchasing an operating system, or even purchasing Linux, but purchasing Red Hat first and foremost.

Honda buys tires from Michelin, airbags from TRW, and paint from Dupont and assembles these diverse pieces into an Accord that comes with certification, warranties, and a network of Honda and independent repair shops.

Red Hat takes compilers from Cygnus, web servers from Apache, an X Window System from the X Consortium (who built it with support from Digital, HP, IBM, Sun, and others), and assembles these into a certifiable, warranted, and award-winning Red Hat Linux OS.

Much like the car industry, it is Red Hat’s job to take what it considers the best of the available open-source components to build the best OS we can. But control over the OS is not held by Red Hat or anyone else. If a Red Hat customer disagrees with our choice of Sendmail and wants to use Qmail or some other solution, they continue to have the control that enables them to do this. In much the same way, someone buying a Ford Taurus may want a higher performance manifold installed on the engine in place of the one that was shipped from the factory. Because the Taurus owner can open the hood of the car, they have control over the car. Similarly, Red Hat users have control over the Linux OS they use, because they have license to open and modify the source code.

You can’t compete with a monopoly by playing the game by the monopolist’s rules. The monopoly has the resources, the distribution channels, and the R&D resources; in short, they just have too many strengths. You compete with a monopoly by changing the rules of the game into a set that favors your strengths.

At the end of the 19th century, the big American monopoly concern was not operating systems, but railroads. The major railroads held effective monopolies on transportation between major cities. Indeed, major American cities, like Chicago, had grown up around the central railway terminals owned by the railroad companies.

These monopolies were not overcome by building new railroads and charging several fewer dollars. They were overcome with the building of the interstate highway system and the benefit of door-to-door delivery that the trucking companies could offer over the more limited point-to-point delivery that the railroad model previously offered.

Today, the owners of the existing proprietary OSes own a technology that is much like owning the railway system. The APIs of a proprietary OS are much like the routes and timetables of a railroad. The OS vendors can charge whatever toll they like. They can also control and change the “route” the APIs take through the OS to suit the needs of the applications they sell, without regard to the needs of the applications that their competitors sell. These OS vendors’ biggest competitive advantage is that they control access to the source code that both their applications and the Independent Software Vendors (ISVs) applications must run on.

To escape the confines of this model, ISVs need an OS model where the vendor of that OS (Linux) does not control the OS; where the supplier of the OS is responsible for the maintenance of the OS only and where the ISV can sell his application secure in the knowledge that the OS vendor is not his biggest competitive threat. The appeal of this OS model has begun to take hold in the software world. This is a big part of the reasoning behind Oracle’s port of their database software to Linux, and behind IBM’s support for Apache.

The benefit of an open-source OS offers over the proprietary binary-only OSes is the control the users gain over the technology they are using. The proprietary OS vendors, with their huge investment in the proprietary software that their products consist of, would be crazy to try and match the benefit we are offering their customers, as we generate a fraction of the revenue per user that the current proprietary OS vendors rely on.

Of course if our technology model becomes accepted by a large enough group of computer users, the existing OS vendors are going to have to react somehow. But that’s still several years in the future. If they do react by “freeing” their code the way Netscape “freed” the code to the Navigator browser, it would result in better products at dramatically lower cost. The industry at large will be well served if that were the only result of our efforts. Of course, it is not Red Hat’s goal to stop there.

As an illustration of the importance of the “control” benefit of the Linux OS, it is interesting to note Fermilab’s experience. Fermilab is the big particle accelerator research laboratory outside Chicago. They employ over a thousand high-level physics engineers who need state-of-the-art technology that they can customize to the needs of the projects they are working on. An example of the benefit of Linux is its ability to be used in cluster farms to build massively parallel super-computers. Fermilab needs this feature, as they are proposing to increase the performance of their accelerator. As a result of this performance increase, they expect to need to analyze almost 10 times more data

per second than they have been. Their budgets simply will not enable them to acquire the computing power they need from the existing super-computer suppliers.

For this and other reasons, Fermilab wanted something Open Source. They recognized that Red Hat Linux was one of the more popular Open Source choices, so they called us. In fact they called us six times in the four months during the system selection phase of the project, and we did not respond even once to their inquiries. Nonetheless, the result of their study was to select Red Hat Linux as an officially supported OS at Fermilab. The moral here is that (a) we needed to learn how to answer our phones better (we did), and (b) that Fermilab was able to recognize that our business model was delivering them the control over the Red Hat Linux OS they were intending to use- whether or not Red Hat was in position to support them.

So whether it is the large computer consuming organizations, or the large computer technology suppliers (ISVs), the Linux OS provides benefits and is free from the major limitations of all the proprietary binary-only OSes available today. Careful brand management of Red Hat Linux among Linux distributions, and careful market position of Linux among OS alternatives, enables Red Hat to enjoy the growth and success we have today.

Licensing, Open Source, or Free Software

The benefit to using Linux is not the high reliability, ease of use, robustness, or the tools included with the Linux OS. It is the benefit of *control* that results from the two distinctive features of this OS; namely, that it ships with complete source code, and that you can use this source code for whatever you chose- without so much as asking our permission.

NASA, the outfit that rockets people off into outer space for a living, has an expression: “Software is not software without source code.”

To the engineers at NASA, high reliability is not good enough. Extremely high reliability is not good enough. NASA needs perfect reliability. They cannot afford to suffer the “blue screen of death” with twelve trusting souls rocketing at a thousand miles an hour around the earth, depending on their systems to keep them alive.

NASA needs access to the source code of the software they are using to build these systems. And they need that software to come with a license that allows them to modify it to meet their needs. Now, I’ll admit that the average dental office billing system does not need the standards of reliability that NASA astronauts depend on to bill patients for their annual teeth cleaning, but the principle remains the same.

And unlike proprietary binary-only OSes, with Linux our users can modify the product to meet the needs of the application they are building. This is the *unique value proposition* that Red Hat offers our customers. This is the proposition that none of our much bigger competitors are willing or able to offer.

This is a value proposition that overturns usual notions of intellectual property. Rather than using a license to lock customers in and wall them off from the source code, Red Hat needs a license that embodies the very idea of access to and control over source code. So what is an acceptable license for the purpose of delivering this unique value proposition? Reasonable people in the Open Source community can and do differ in how they answer this question. But here’s mine:

The General Public License, from the Free Software Foundation www.fsf.org, ensures the modifications and improvements made to the OS remain public, is effective for managing a cooperative development project.

My definition of “effective” goes back to the old days of Unix development. Prior to 1984, AT&T used to share the source code to the Unix OS with any team who could help them improve it. When AT&T was broken up, the resulting AT&T was no longer restricted to being a telephone company. It decided to try and make money selling licenses to the Unix OS. All the universities and research groups who had helped build Unix suddenly found themselves having to pay for licenses for an OS that they had helped build. They were not happy, but could not do much about it- after all, AT&T owned the copyright to Unix. The other development teams had been helping AT&T at AT&T’s discretion.

Our concern is the same. If Red Hat builds an innovation that our competitors are able to use, the least we can demand is that the innovations our competitors build are available to our engineering teams as well. And the GPL is the most effective license for ensuring that this forced cooperation among the various team members continues to occur regardless of the competitive environment at the time.

Keep in mind that one of the great strengths of the Linux OS is that it is a highly modular technology. When we ship a version of Red Hat Linux, we are shipping over 800 separate packages. So licensing also has a practical dimension to it. A license that enables Red Hat to ship the software but not make modifications to it creates problems because users cannot correct or modify the software to their needs. A less restrictive license that requires that the user ask the permission of the original author before making changes still burdens Red Hat and our users with too many restrictions. Having to ask possibly 800 different authors or development teams for permission to make modifications is simply not practical.

But Red Hat is not ideological about licenses. We are comfortable with any license that provides us with control over the software we are using, because that in turn enables us to deliver the benefit of control to our customers and users, whether they are NASA engineers or application programmers working on a dental office billing system.

The Economic Engine Behind Development of Open Source Software

The interesting stories of where Linux comes from help illustrate the strong economic model that is driving the development of this OS.

The Open Source community has had to overcome the stereotype of the hobbyist hacker. According to this stereotype, Linux, for example, is built by fourteen-year-old hackers in their bedrooms. We see here an example of the Fear, Uncertainty, and Doubt (FUD) foisted on the software industry by vendors of proprietary systems. After all, who wants to trust their mission-critical enterprise applications to software written by a fourteen-year-old in his spare time?

The reality, of course, is very different from this stereotype. While the “lone hacker” is a valuable and important part of the development process, such programmers account for a minority of the code that makes up the Linux OS. From the head of the kernel team, Linus Torvalds, on down most of the code in the Linux OS is built by professional software developers at major software, engineering, and research organizations.

A few examples include the GNU C and C++ compilers that come from Cygnus Solutions Inc. of Sunnyvale, California. The X Window System originally came from the X Consortium (made up of support from IBM, HP, Digital, and Sun). A number of Ethernet drivers are now largely the responsibility of engineers at NASA. Device drivers are now coming frequently from the device manufacturers themselves. In short, building new open-source software is often not so different from building conventional software, and the talent being Open Source is by and large the same talent that is behind conventional software.

Grant Guenther, at the time a member of Empress Software’s database development team, wanted to enable his co-workers to work on projects from home. They needed a secure method of moving large files from their office to home and back. They were using Linux on PCs and using Zip drives. The only problem was that at the time (1996), good Zip drive support was not available in Linux.

So Grant had a choice: throw out the Linux solution and purchase a much more expensive proprietary solution, or stop what he was doing and spend a couple of days writing a decent Zip drive driver. He wrote one, and worked with other Zip drive users across the Internet to test and refine the driver.

Consider the cost to Red Hat, or any other software company, of having to pay Empress and Grant to develop that driver. Safe to say the cost would have been in the tens of thousands of dollars, and yet Grant chose to “give away” his work. In return, instead of money he received the use of a great solution for his problem of enabling Empress programmers to work from home, at a fraction of the cost of the alternatives. This is the kind of win-win proposition offered by cooperative models like the Open Source development model.

The Great Unix Flaw

The best example I know of to illustrate that the Linux model is a profoundly different approach to building OSes is to look at what many people are convinced is the ultimate outcome of this OS effort, namely that Linux will balkanize the same way all the Unixes have. There are apparently thirty different, largely incompatible, versions of the Unix OS available today.

But the forces that drive the various Unixes apart are working to unify the various Linuxes.

The primary difference between Unix and Linux is not the kernel, or the Apache server, or any other set of features. The primary difference between the two is that Unix is just another proprietary binary-only or IP-based OS. The problem with a proprietary binary-only OS that is available from multiple suppliers is that those suppliers have short-term marketing pressures to keep whatever innovations they make to the OS to themselves for the benefit of their customers exclusively. Over time, these “proprietary innovations” to each version of the Unix OS cause the various Unixes to differ substantially from each other. This occurs when the other vendors do not have access to the source code of the innovation and the license the Unix vendors use prohibit the use of that innovation even if everyone else involved in Unix were willing to use the same innovation.

In Linux the pressures are the reverse. If one Linux supplier adopts an innovation that becomes popular in the market, the other Linux vendors will immediately adopt that innovation. This is because they have access to the source code of that innovation and it comes under a license that allows them to use it.

An example of how this works is the very example that all the Linux skeptics have been using to predict the downfall of the OS, namely the debate in 1997 between the older libc libraries and the new glibc libraries. Red Hat adopted the newer glibc libraries for strong technical reasons. There were popular versions of Linux that stuck with the older libc libraries. The debate raged for all of six months. Yet as 1998 drew to a close all the popular Linux distributions had either switched or announced plans to switch to the newer, more stable, more secure, and higher performance glibc libraries.

That is part of the power of Open Source: it creates this kind of unifying pressure to conform to a common reference point- in effect, a open standard- and it removes the intellectual property barriers that would otherwise inhabit this convergence.

It's Your Choice

Whenever a revolutionary new practice comes along there are always skeptics who predict its inevitable downfall, pointing out all the obstacles the new model must overcome before it can be called a success. There are also the ideologues who insist that it is only the purest implementation of the new model that can possibly succeed. And then there are the rest of us who are just plugging away, testing, innovating, and using the new technology model for those applications where the new model works better than the old one.

The primary benefit of this new technology model can be seen in the birth of the PC. When IBM published the specs to its PC in 1981, why did the world adopt the PC computing model with such enthusiasm? It was not that the IBM PC was a better mousetrap. The original 8086-based PCs shipped with 64K (yes, K) bytes of main memory. They had an upper memory limit of 640K. No one could imagine that a single user would need more than 640K on their individual machine. A tape cassette recorder was available for data back-up.

What drove the PC revolution was that it provided its users with control over their computing platform. They could buy their first PC from IBM, their second from Compaq, and their third from HP. They could buy memory or hard drives from one of a hundred suppliers, and they could get an almost infinite range of peripheral equipment for almost any purpose or application.

This new model introduced a huge number of inconsistencies, incompatibilities, and confusion, between technologies, products, and suppliers. But as the world now knows, consumers *love* choice. Consumers will put up with a measure of confusion and inconsistency in order to have choice- choice and control.

Notice also that the PC hardware business did not fragment. Specifications have generally remained open, and there is strong pressure to conform to standards to preserve interoperability. No one has a sufficiently better mousetrap with which to entice users and then hold them hostage by going proprietary. Instead innovations- better mousetraps- accrue to the community at large.

The Linux OS gives consumers choice over the technology that comes with their computers at the operating system level. Does it require a whole new level of responsibility and an expertise on the part of the user? Certainly.

Will that user prefer to go back to the old model of being forced to trust his proprietary binary-only OS supplier once he has experienced the choice and freedom of the new model? Not likely.

Critics will continue to look for, and occasionally find, serious problems with Linux technology. But consumers love choice, and the huge Internet-based open-source software development marketplace is going to figure out ways to solve all of them.

"Giving It Away" is copyrighted 2003 by Robert F.Young.. All rights reserved to me.

About the Author

As co-founder and formerly CEO and Chairman of Red Hat from 1993-2000, Bob was responsible for early success of Red Hat. Red Hat is credited with driving the global, industry-wide adoption of open source development practices.

A true open source visionary, Bob's efforts developing Red Hat into a household name have won him prestigious honors such as being named one of *Business Week Magazine's* "Top Entrepreneurs" for 1999. Bob graduated from the University of Toronto in 1976 prior to beginning his high tech career in the computer finance arena.

His 20 years of computer industry finance and marketing experience at the head of the two computer leasing companies he founded (from 1976 to 1993), combined with his innate marketing savvy contributed to Red Hat's success. His book, "Under the Radar", chronicles how Red Hat's open source strategy successfully won wide industry acceptance in a market previously dominated by proprietary binary-only systems.

In addition to Bob's chairmanship of Red Hat, in 1999 he founded The Center for the Public Domain, a non-profit foundation that supports the growth of a healthy and robust public domain of knowledge and the arts. Since its inception as a venture philanthropic enterprise, The Center for the Public Domain has contributed to the founding of several organizations that study, nurture and secure the health of the public domain through major gifts, grants and donations. These efforts have strengthened the public community of shared knowledge and culture across many sectors - technology, law, medicine, academic research, education, media and the arts.

Bob is now actively leading his latest commercial venture, Lulu.com, which is giving artists and other content owners new options for control and distribution of their work, as well as delivering more information at lower cost to the American consumer.