Ganesh Prasad: Open Source-onomics: Examining some pseudo-economic arguments about Open Source

(*Apr 12th, 16:00:03*)

By Ganesh Prasad

Synopsis:
While the technical arguments against Linux and Open Source are being gradually silenced, several unrefuted myths about the economics of Open Source continue to float about, confusing and scaring off people considering these alternative products. Worse, the Open Source community is itself divided on such issues, and is unable to provide a cogent rebuttal. This article is an attempt to set the record straight.

Contents

"The poor performance of Linux stocks proves that Linux is a failure"

What's the relationship between the performance of Linux stocks and Linux's own prospects of success? When stocks of companies like Red Hat and VA Linux Systems skyrocketed in the wake of their IPOs, that was taken as an indication that Linux had arrived, and Linux advocates said nothing to counter the impression. Indeed, many

gleefully used the stockmarket to show their peers that Linux was to be taken seriously. So now that the same stocks are trading far below those prices, doesn't it indicate that the Linux shine has worn off? Look at the number of Linux companies in the doldrums, that have laid off employees or closed down. That certainly seems to indicate the end of Linux. It was a great idea that failed to deliver on its promises, and we should now go back to software and companies that are more firmly grounded in economic realities, right?

Well, first of all, Linux is quite independent of Linux companies in a way that the market has never seen before. Windows means Microsoft, Netware means Novell, OS/390 means IBM. The fortunes of operating system and company are usually heavily intertwined. That's simply not the case with Linux. If Novell closes down, that pretty much means the end of Netware, unless another company sees fit to buy the product and keep it alive (On the other hand, Microsoft may simply choose to buy Netware and kill it!). Such things can't happen to Linux. As an Open Source operating system, Linux is teflon-coated against the commercial failures of the companies that try to build business models around it. Commercial entities are Johnnies-come-lately to Linux anyway. Linux managed without them for years, and will continue to exist even if they should all disappear.

In fact, companies that claim to support Linux are wrong -- Linux supports *them*!

"Open Source is not economically viable"

OK, so Linux as a technical product may continue to exist, but if companies cannot make money from it (as is seemingly evidenced by the woes of the Linux companies today), then it's another great technical success that is a commercial failure. History is littered with such examples. Linux will never go anywhere unless people can make money off it.

Now here's an argument even Open Source sympathisers have trouble with, -- the assumption that money must be made for Open Source to succeed. However, the argument is incomplete because it chooses to concentrate on the supply side alone, without regard to the demand side.

While it may well be true that no one can make money from Open Source, that should only serve to discourage *suppliers* of software. On the demand side, however, *consumers* are saving tons of money by using Open Source. Since a penny saved is a penny earned, there is a strong economic basis for the success of Open Source after all. Someone is saving money, and they will fight to keep those savings.

The demand side is the one that should drag the rest of the market, kicking and screaming, to a regime of drastically lower prices. Vendors will see their margins shrink, many will close down, newer, leaner ones will spring up, vendors in other market segments will provide software, and eventually, the market will adjust itself to the new reality. Dollar volumes will go down even as unit volumes go up. The transition could be quite painful for suppliers of software, but no law of economics says it cannot happen. It is not a law of nature that vendors must continue to make the revenues and profits they are used to.

"Not paying for software will ultimately kill the industry"

There are the long-term worriers who don't like this scenario at all, even as they accept that it may happen. Yes, they say, customers will save money in the short term, but they're eating their seed corn. Customers need financially healthy vendors to be around to support them and continually improve their offerings. A herd of gnu may be happy at the disappearance of the local lion population, but the herd needs predators to cull its ranks of the weak and the sick, and to keep its gene pool healthy. Saving money by starving your suppliers is not in your own long-term self-interest.

This is a strange suggestion from people who probably describe themselves as market capitalists. When customers make a purchase, should they think about their own savings or should they worry about the supplier or the economy? Is it reasonable to ask them to choose costlier products because that will ultimately and indirectly serve their own interests? The argument smacks more of Marx than Adam Smith, -- The State above the individual.

This was the convenient argument of horse-buggy manufacturers when the locomotive arrived, and of the railroad companies when the aeroplane appeared. We've seen this dozens of times in our history. A generation of suppliers is threatened, and they try to convince the rest that society as a whole is threatened. If history is any guide, consumers will make the decisions that suit their immediate interests, and vendors will have no choice but to adapt as best as they can. Those decisions may decimate them, but civilisation will survive, as it always has. *L'Etat, c'est moi*.

"Why will programmers continue to contribute code if they can't make money from it?"

Right. Given a choice between a free software product and a competitor with a price tag, it is understandable if customers choose the free one. But why should anyone write it for free in the first place? What would they gain?

The assumption behind this question is that there are only three types of transactions between parties: win-win, win-lose and lose-lose (Lose-lose transactions should never occur under conditions of rational decision-making). Win-lose transactions occur when the winning party is stronger than the other and can force a transaction through. All other transactions are willingly entered into by two parties and are win-win.

In the case of Open Source, the recipients of the software are obviously winners, but the writers of the software don't seem to be winning anything because the recipients don't have to pay them for it. Therefore, our assumption tells us that this not a win-win situation, and that there is no economic incentive for a programmer to write Open Source software.

For the moment, let us go along with the assumption that the only motivation for writing software is economic (which is not true). Even with such an assumption, the reasoning is flawed because there are other types of transactions which are not so obvious and which have not been considered: win-neutral, lose-neutral and neutral-neutral.

Under conditions of rational decision-making, lose-neutral and neutral-neutral transactions have no incentive to occur, but win-neutral transactions can and do occur quite frequently. Everyday examples include someone asking for directions, or asking for

change. Here, the person asking certainly gains something from the transaction, but the other party neither gains nor loses from it. Therefore, the transaction can still take place.

Most Open Source programmers would probably not write software and give it away if it cost them something to do so. However, they don't perceive the effort of writing it to be a cost. Most of them write software to solve a specific problem that they happen to be facing, or to "scratch their personal itch", as Eric Raymond points out. The process of developing such software is actually quite pleasurable and energising to most good programmers. Once the software has been written, giving away copies of it does not deprive the programmer of the ability to continue to use it, and it costs them nothing extra to do so. It is a win-neutral transaction, and therefore there is no economic reason to prevent it from taking place.

(Economics purists would point out that there is indeed a cost to giving away the software -- the *opportunity cost* of not selling the software instead. However, for many programmers, the process of selling their software is more trouble than it is worth, so the *effective* opportunity cost is actually zero, and it is a win-neutral situation after all.)

If that was not sufficient reason, Open Source programmers also tend to work with others who share their interest and contribute code. They enjoy a multiplier effect from such cooperation. Metaphorically speaking, each programmer contributes a brick and *each* gets back a complete house in return. In software, unlike with physical goods, one person's gain does not come at the expense of another because a copy does not deplete the original in any way. Sharing software is not a zero-sum game, and there are tremendous efficiencies from participating in such a cooperative endeavour.

No, the absence of direct monetary reward does not really constitute a disincentive to writing Open Source software.

"Even Open Source development involves effort, so there has to be payment for that effort"

OK, Open Source programmers lose nothing by giving away the software that they have already written (and they may even gain in non-monetary terms). But some effort *has* gone into their products. Shouldn't such effort be compensated in cash as well? Programmers have families to support, and they need to put bread on the table. They can't live on software and satisfaction alone. To be viable in the long term, Open Source needs to evolve a mechanism to support its contributors financially. Without remuneration, over time, most of these volunteer programmers will simply wander away in search of food.

This argument appeals to equity as well as economic commonsense, and finds sympathisers even in the Open Source community. Certainly, we would all like to see programmers being compensated for their contributions. There are several business models that are being attempted. The SourceForge and Collab.Net method of raising contributions from users to pay developers is an innovative one, but its success is as yet unproven. Programmers could also try and make money by supporting their creations, maybe selling copies of it as well, providing consultancy and professional services, etc. But we still don't know of a foolproof business model for this sort of thing. There may not

even be one. In the absence of a good system coming along pretty soon, Open Source will perhaps continue to be written by volunteer programmers who have day jobs writing commercial software. It could also expect contributions from hardware or services companies with a stake in its success.

But even in this worst case, does it mean that Open Source will stop being written? As long as Open Source programmers have alternative sources of income (i.e. day jobs), they lose nothing by working on Open Source projects in their spare time (a win-neutral transaction). With the increasing number of people being exposed to Open Source, the pool of contributors is in fact growing larger by the day.

"Are Open Source programmers writing themselves out of their jobs?"

But that leads to what may seem the ultimate argument against the economics of Open Source: How long can programmers work day jobs at commercial software companies and write software at night that puts those same companies out of business? Writing Open Source software is not just irrational, it is positively suicidal. 'Tis an ill bird that fouls its own nest, not to mention an extremely foolish one.

Indeed, this appears to be a very powerful argument. However, Eric Raymond comes to our rescue with this statistical nugget: Only 5% of all programmers are actually engaged in writing "for sale" commercial software. The other 95% actually write and maintain custom-built software for in-house use. Open Source doesn't threaten custom-built software at all. It only competes with packaged software that is sold as a product. And so, in the worst case, Open Source programmers are only going to put 5% of their own kind out of work. That's an acceptable level of collateral damage, as the generals might say.

"But free isn't natural. There's no such thing as a free lunch."

But this entire idea is crazy, somewhat like producing something out of nothing! How can one seriously expect an entire economy to be based on something that is absolutely free? Doesn't it violate some fundamental economic law, just as producing something out of nothing violates the Law of Conservation of Mass in physics?

Let's examine whether it does.

We realise it is not possible for any supplier to charge less for a product than it cost them to produce it. That would mean a loss. At the same time, if all products in a category are roughly alike in function, and there are plenty of suppliers for those products, it is not possible for any of them to charge significantly more than their competitors without pricing themselves out of the market. So they should all end up charging just slightly more than it cost them to make the product, making only modest profits in the process. The underlying assumption here, though, is that we have "pure competition".

"Pure competition" in economics means a buyers' market. Consumers love it and suppliers hate it (though, curiously, all suppliers claim to welcome it). A competitive market means that consumers can easily find any number of alternative suppliers for a product. It also means the product is a *commodity*.

A "commodity" product means that there is very little differentiation between the various versions of a product. They all do the same thing, with only minor, insignificant differences. Consumers don't bother about brands when buying commodities. Suppliers hate commoditisation for the same reason and try their best to create artificial differentiation. (The best example is the Vodka Paradox: Vodka, by definition, is a colourless, odourless and flavourless drink of a specific composition, so all vodkas should be the same! But we know of both premium and downmarket brands of vodka, so at least some of them are, by definition, not vodka at all!)

Look at the software market from these angles. Is it competitive? Is it a commodity market? Think about whether it would be easy for you to replace Windows on your PC with another operating system. Think about whether such a system would work the same way. Such an analysis may suggest that this is neither a competitive nor a commodity market. However, these aren't very straightforward questions to answer because some recent developments have impacted the market a great deal, but we'll come back to them a bit later.

The important point to note is, if the software market becomes a competitive commodity market, the price of software should be close to the cost of producing it. That's what economic commonsense says should happen.

"Is software a commodity?"

But why would the software market suddenly turn competitive and into a commodity market? The answers are standards, the Internet, and Open Source software itself.

The correct way to build an application using 1990s thinking is to grab a copy of Visual Basic or PowerBuilder, develop a Windows executable and install it on every user's PC. The larger the number of PCs to install the software on, the more you walk around. When you need to upgrade the software, you put on your sneakers again and take another walk.

Now fast-forward to today. The correct way to build an application using millennial thinking is to put the application on a website and get users to point their browsers at it. When you need to upgrade the software, you modify it once on the server and your users hit the Refresh button on their browsers.

Web technology, if you stop to think about it, is a predominantly server-side technology. True, there's the Java applet, but hardly anyone uses it. There's Javascript, but ever since the browser wars, when you couldn't be sure which browser would break your code, developers have been wary of coding a lot of Javascript. That leaves virtually all development on the server side. All that the user needs is a lowly browser.

At one stroke, the web has commoditised the server, because all a server needs to do is talk some standard "protocols". If it knows HTTP (HyperText Transfer Protocol), it can talk to a browser. If it spits out some HTML (HyperText Markup Language), the browser can actually render it for the user to read. Whither brand? Neither the browser nor the user sees the brand of the server software. The same goes for other Internet standards such as SMTP (Simple Mail Transfer Protocol) and LDAP (Lightweight Directory Access Protocol). They have completely commoditised the servers that implement them.

Suddenly, standards are king, and anyone can play.

The favourite vendor tactic, -- differentiation, -- doesn't work very well in this situation. Differentiation breaks standards. *More is less*. A product that fails to comply with a standard is automatically incapable of surviving in the Internet ecosystem. Any superfluous features it boasts simply wither away through disuse. It's a self-perpetuating discipline. Internet protocols and standards rule with an iron fist. So it looks as if commoditisation is here to stay, however much vendors may hate it.

What's more, every such standard and protocol is faithfully implemented in at least one Open Source product. That keeps commercial implementers honest, too. No oligopoly is possible in the Internet-era software market, because any Open Source implementation pre-emptively breaks the cartel! (As an example, the nascent oligopoly among web application server vendors is coming under severe pressure from the Open Source JBoss and Enhydra. Expect to see prices tumble in this market).

And so, here we are, in a competitive and commodity market after all. We know that in such a market, the price of software will be close to the cost of producing it. So if we can show that the cost of producing software is zero, then the price tag of zero is justified.

"Who will invest in software development if it doesn't yield a return?"

It sounds a preposterous argument on the face of it. How can the cost of software ever be zero? Doesn't it take significant effort to develop software? Even Open Source software is not miraculously produced. Programmers spend many man-months of effort on it. So how can the price of software ever be zero?

Selling below cost is considered predatory pricing in many countries. In international trade, it's called "dumping". Is Open Source guilty of "dumping" or predatory pricing? If unchecked, this could destroy the commercial software industry. Who will invest in developing software if they cannot recoup their development costs?

The answer to this question may be surprising, because it overturns many of our fundamental assumptions about the way the world is run.

Let's start by observing that if Linux had been developed by a commercial organisation, it could never have been free. Commercial organisations, whether funded by debt or by equity, need to show a return on their investment. They cannot waste that investment by giving away their products. Therefore, even if it costs nothing to create *additional* copies of software (what's called the "marginal cost" of software), the initial costs of development must be spread over many copies, they must be priced in such a way that those costs can be recouped, and a positive return must be shown on the initial investment.

Of course, for this to work, *software must be shoehorned into the mould of a physical product*. Copying of software by anyone other than the producer must be made a crime. The infinite replicability inherent in software must be artificially curtailed through legislation. Only then can the model work. This is precisely what we have with commercial software today. It is important to understand that the commercial model works by imposing a system of artificial scarcity. It is physically possible and economically feasible to produce as many copies of software as the world needs, but that

is however, legally punishable. That means that many people who need software but cannot pay the asking price must go without it. That is the only possible (legal) outcome. There are people who need a software product, and the product can be replicated at little cost, yet the transaction cannot take place. From society's viewpoint, this inefficiency is the price it pays for choosing a commercial vehicle for software development.

But now, consider an alternative to the investment model. If the cost of software development can somehow be treated as an expense, and simply *written off*, then the software is freed from the requirement to show a return on investment. There will be no need to artificially constrain its natural replicability. The world can have as many copies of it as it needs. There will be no need for restrictive legislation. From society's point of view, what could be more efficient?

Large expenses, however, cannot readily be written off. They need to be "amortised" over a sufficiently large number of units. This is where another property of software becomes invaluable. Software can quite practicably be developed by hundreds, even thousands of programmers. Other intellectual works, such as books, music or movies, while sharing software's trait of infinite replicability, cannot be produced by a cast of thousands. Of all the works of mankind, physical and intellectual, software stands alone in its twin characteristics of infinite replicability and amortisability of effort.

Looked at this way, Open Source seems the more natural and efficient way to build software. Get a large number of interested developers to work on a piece of software. Most of them spend less than a couple of hours a day on it, so they don't mind "writing off" the effort in terms of expecting a monetary return. That is why Open Source operating systems and associated software are free for every man, woman and child on earth to copy. By keeping important software like Linux out of the ambit of commercial interests, society has benefitted handsomely.

Software, like wealth itself, is potentially limitless. Capitalism correctly views wealth as potentially infinite, and fuels global growth to increase the overall size of the economic pie. However, the current structure of the commercial software market is not capitalistic at all, but *mercantile*. It sees software as a limited good that needs to be hoarded and released sparingly. It is therefore incapable of being an engine of growth.

Lest our current "capitalistic" mileu should give anyone the wrong idea, it must be noted as a matter of sociological interest that commercial organisations do not have a divine right to exist. They exist at society's pleasure, because they have hitherto been the most efficient *known* means of producing quality goods and services at reasonable prices. However, it appears that the investment model that underlies all commercial activity is a grossly inefficient vehicle to deliver to society the levels of software that it needs.

So here's a really subversive thought: Perhaps corporations shouldn't develop software at all! Just as free market advocates call for governments to get out of the business of running industries, perhaps we should call for corporations to get out of the business of writing software. They are applying the wrong economic model to software, and it is proving too costly and inefficient for society to bear. We need a model that takes a capitalistic view of software, not a mercantile one.

What we see today with the gradual success of Open Source is perhaps society's

"invisible hand" turning over software development to the more efficient (from its viewpoint) Open Source vehicle, and gradually relegating commercial software to the fringes of economic activity. Adam Smith would have approved.

(Along the way, notice that we have also shown how the cost of software can be effectively reduced to zero, thereby justifying its zero price-tag.)

"Open Source may have a niche, but proprietary commercial products will continue to rule"

We may have shown that Open Source is viable and will most likely continue to survive, maybe even thrive. But isn't it too much to suggest that proprietary, commercial software will go the way of the dodo? Is this really a commodity market? Aren't most leading commercial software products ahead of their Open Source equivalents, anyway? How can Open Source hope to beat commercial software in features? For example, can the Open Source database package PostgreSQL ever hope to match Oracle? Customers won't use inferior products just because they're free! They'd prefer to pay for better products.

This situation is similar to the story of the two men who come upon a tiger in the jungle. One of them starts putting on his running shoes. "Are you crazy?" whispers the other, "You can't hope to outrun a tiger!" "I don't have to outrun the tiger," explains the first, "I only have to outrun you!"

Open Source products don't have to become better than their commercial equivalents. They just have to become good enough to meet user requirements. Why would users pay for features they don't need? Do you really need a webserver inside your database? A Java Virtual Machine, perhaps? Or a whole host of features you're never going to use? No? Then why pay for Oracle 9i? PostgreSQL lets you create tables, fill them with data, fire SQL queries at them, and gives you reasonable performance. Isn't that good enough for you, and for 90% of the market? So PostgreSQL didn't have to outrun the tiger, did it?

Notice that this is an economic argument. It is not a technological argument along the lines of "Open Source products evolve faster and fix bugs quicker, so they'll get better than their commercial rivals one day". Actually, we couldn't care less. At a certain point in time, commercial vendors may be reduced to selling differentiated features that 90% of the market doesn't need, while the most commonly-required features will be available to all, free of charge. Those common features will conform to standards, while proprietary, differentiating features will remain exactly that, -- proprietary and non-standard.

It is such commoditisation of the market that could slaughter proprietary commercial software, driving it into niches and ensuring that the mainstream goes Open Source.

"Customers will never trust something that is free"

All of this sounds pretty convincing in theory, but Open Source should have been growing like gangbusters if all of this is true. But we see very gradual adoption of Open Source in the market. Is it perhaps because people will never respect and trust something that is free...?

In economics, we have two concepts, -- competing products and substitutes. Competing products are other brands in the same category. Substitutes are products in another category that perform much the same function. If I don't like Nescafe, I'll go with Moccona (a competitor), but if I read a medical report finding that coffee is extremely dangerous, I will drink tea rather than coffee when the urge hits me. It's not the same thing, but I could bring myself to settle for tea. That's what a substitute means.

It's more difficult to switch to a substitute than to a competing product, but it can be done when there are compelling reasons. Open Source software is a substitute, not a competitor, to the entire category of proprietary commercial software. It requires a different mindset and a willingness to work with different development and support mechanisms. That's what makes its uptake less than straightforward.

With both substitutes and competitors, "good enough" is a great reason to switch when the price is far lower, and that is what Open Source offers. But with substitutes, there is an extra mental adjustment process that consumers need to go through before full acceptance happens. That takes time. Consumers need time to gain confidence from the positive examples of early adopters. The current situation with Open Source in the marketplace reflects exactly this stage of the proceedings. Potential savings and a greater degree of control over one's destiny are the compelling arguments that will encourage the switch. When the mental adjustment process is complete, the downfall of proprietary software could be swift (put options on Oracle, anyone?).

"Open Source may release value, but it doesn't create value"

New thinking among financial analysts has discovered that most firms reporting improved earnings year after year are doing so by cutting costs rather than by increasing revenue. By reducing waste and improving productivity, companies are "releasing" value that was hitherto "locked up" in inefficient processes. But they aren't creating new value. They're not innovating. True wealth comes from new ideas, and there don't seem to be too many of them. So there are natural limits to how far these companies can go before they hit a plateau.

Isn't Open Source something similar? Sure, it'll help us reduce costs, but is it helping us *create* anything? It's nothing more than a cheaper substitute for our existing software, so its long-term impact will probably be marginal, not revolutionary.

Well asked, and therein lies the difference between a market and a community. To play in a market, you need to have money. That automatically excludes all the people who can't pay. It's a shame that in a world of over 6 billion people, about half are just bystanders watching the global marketplace in action. There are brains ticking away in that half-world of market outcasts that could contribute to making the world better in a myriad little ways that we fortunate few don't bother to think about. There are problems to be solved, living standards to be raised, yes, value to be created, and the "market" isn't doing it fast enough.

God, Government, Market and Community

There are millions who have been waiting for generations for their lot to improve. Religion has promised them a better afterlife, but no god has seen fit to improve their

present one. In a world where socialism has been humiliatingly defeated, governments seem ashamed to spend money on development. Everyone now seems to believe that governments must be self-effacingly small. The market is now the politically correct way to solve all problems. But the market, as we have seen, doesn't recognise the existence of those who have nothing to offer as suppliers and nothing to pay as consumers. They are invisible people.

Therefore it falls to the miserable to improve their lot themselves. Given the tools, they can raise themselves out of their situation. They will then enter the market, which will wholeheartedly welcome them (though it hadn't the foresight to help them enter it in the first place).

Where will such tools come from? In a world where intellectual property has such vociferous defenders that people must be forced to pay for software, information technology widens the gap between the haves and the have-nots, a phenomenon known as the digital divide. If producers of software deserve to be paid, then that means hundreds of thousands of people will never have access to that software. That's a fair market, but a lousy community.

Open Source is doing what god, government and market have failed to do. It is putting powerful technology within the reach of cash-poor but idea-rich people. Analysts could quibble about whether that is creating or merely releasing value, but we could do with a bit of either.

And yes, that is revolutionary.

Conclusion

Is it possible to make money off Open Source? In the light of all that we have discussed, this now seems a rather petty and inconsequential question to ask. There is great *wealth* that will be created through Open Source in the coming months and years, and very little of that will have anything to do with money. A lot of it will have to do with people being empowered to help themselves and raise their living standards. No saint, statesman or scholar has ever done this for them, and certainly no merchant. If this increase in the overall size of the economic pie results in proportionately more wealth for all, then that's the grand answer to our petty question.

Economics is all about human achievement. It wasn't aliens from outer space who raised us from our caves to where we are today. It was the way we organised ourselves to create our wealth, rather like the donkey with a carrot dangling before it that pulls a cart a great distance. Open Source gives means to human aspiration. It breaks the artificial mercantilist limits of yesterday's software market and unleashes potentially limitless growth.

When the dust settles, and even the greatest industrial creations of today stand dwarfed by the scale of development that Open Source will bring in its wake, the world will have learnt a thing or two about economics.

References

1. "The Cathedral and the Bazaar" by Eric S. Raymond (http://www.tuxedo.org/~esr/writings/cathedral-bazaar)
2. "The Magic Cauldron" by Eric S. Raymond (http://www.tuxedo.org/~esr/writings/magic-cauldron)
3. "A-Level Economics" by Ray Powell, Letts Educational
4. "The Real Meaning of Money" by Dorothy Rowe, Harper-Collins, 1997

About the Author

Ganesh Prasad has been a Linux user since 1996, and his major fascination with Open Source has been its social and economic impact, though the technical side has its appeal, too. He has been troubled by much of the pseudo-economic bunkum around Open Source, and has decided to shine the brilliant light of his logic to cut through the clutter, making up for lack of rigour with stabs of attempted humour.

Solutions

Whitepapers and eBooks

Microsoft Article: SQL Server 2008 Table and Backup Compression
Intel Article: Save Time and Money by Streamlining Your Budget Process
IBM CXO Whitepaper: The New Information Agenda--Do You Have One?
Internet.com eBook: Becoming a Better Project Manager
Microsoft Partner Portal: Partner Marketing Requires a Game Plan
Internet.com eBook: Web Development Frameworks for Java
Internet.com eBook: Developing a Content Management System Strategy
HP eBook: Easy Storage for the SMB--Has the Time Come?
MORE WHITEPAPERS, EBOOKS, AND ARTICLES

Webcasts

Microsoft Partner Portal Video: Microsoft Gold Certified Partners Build Successful Practices
HP On Demand Webcast: Grid Computing Meets Business
MORE WEBCASTS, PODCASTS, AND VIDEOS

Downloads and eKits

Microsoft Download: Silverlight 2 SDK
Iron Speed Designer Application Generator
Microsoft Download: Silverlight 2 Developer Runtime (Windows)
Amyuni Download: PDF and XPS Engine for Your .NET and ActiveX Applications
MORE DOWNLOADS, EKITS, AND FREE TRIALS

Tutorials and Demos

Microsoft Article: Extending Silverlight to WPF (and Vice-Versa)
Internet.com Hot List: Get the Inside Scoop on the Hottest IT and Developer Products
IBM Cognos 8 BI Reporting Demo
MORE TUTORIALS, DEMOS AND STEP-BY-STEP GUIDES