From: Jim Gray
Sent: Mon, 5/11/1998 2:30 PM
To: Nathan Myhrvold; Bill Gates; Eric Rudder, Jim Gray; Gordon Bell; Rick Rashid; Chuck Thacker; Roger Needham; Paul Maritz; Jim Allchin (Exchange); Gregory Faust; Dan Rosen; Greg Maffei; Charles Simonyi; Mike Murray
Subject: RE: Free software economics: The DollarOS

Nathan:
A stimulating and well-reasoned argument for priced-software.

Two things trouble me about the situation you describe:

(1) **Linix is a cult that captures the best-and-brightest kids.**
This is anecdotal, but I see it regularly in 10-year-olds, high schoolers,
and University faculty comments on inbound students.
The Linix cult views Solaris as bad and Windows as evil or stupid.
SUN has put the Solaris source on public view
(500$ gets you the CD, and you can apply that towards the purchase of SUN gear.)
The contract is draconian (SUN owns the rights to derivative works) but no one seems to notice that.
All of this is simply bad for us.
We have dramatically relaxed the rules for access to NT Source,
but we are still a long way from our competitors.
Solaris is "scale" player in the OS space.
Linix is a huge training ground and experimental laboratory for Solaris.
Suggestion: We need to find an analog to create a "cult" of core windows developers.

(2) **I assume Windows is heading for a on dollar OS: the DollarOS.**
Currently we sell about 30 million W95, NTs, WinCE per year.
I assume that W98 and NT will converge
Following Moore's law, we will be selling 100 M WinNTs/year in 5 years.
I also hope that WinCE takes over the PDA space and so it will sell 100 M/year in 5 years.
These do not seem radical numbers to me.
But, I also expect that each disk drive and NIC will want to run either NT or WinCE.
(right now Wind River is the OS of choice here and it is quite pricey and not very good).
All these "peripherals" will have controllers that are supercomputers and will have
128MB of DRAM in that time frame.
So, all those microprocessors are going to want an OS, a network stack, security, management, and TOOLS.
This trend could drive our volumes up 2x more to about 400 M units/year in 2003 (a wild and optimistic guess).
But the trend requires CheapOS (say one dollar for a disk controller that costs 30$ to make).
I assume WinCE is our move towards CheapOS and LiteOS but I also guess that we will face a WinCE - NT
convergence in that timeframe.
All these numbers are HUGE volumes.
If the volumes go up 100x then prices could rationally drop 100x
(which is about right for the dollar OS rather than the 25$ OS).
So, I think we might be a VERY high volume and low-cost OS company in five years.
Jim

Jim Gray, Microsoft Research, 301 Howard St #830, SF CA 94105
tel. 415-778-8222 fax -8210 Gray@Microsoft.com
http://research.microsoft.com/barc/gray (Intranet. http://msr/groups/barc)

-----Original Message-----

From: Nathan Myhrvold
Sent: Sunday, May 10, 1998 9:55 PM
To: Bill Gates (billg), Eric Rudder; Jim Gray (gray), Gordon Bell (gbell), Rick Rashid (rashid); Chuck Thacker; Roger Needham; 'paulma', Jim Allchin (jimall), Gregory Faust; 'Dan Rosen', Greg Maffei, 'Charles Simonyi (charless)'; 'Mike Murray'
Subject: Free software economics

Free software, or even software distributed in source form is a current cause celebre. Netscape, Linux and others are doing it. Is this some sort of trend?

There are two answers - one immediate and pragmatic and the other philosophical.

The pragmatic answer is that much of the "trend" towards free software is very likely due to the novelty of the Internet. In the early days of the PC industry there was a period when "Shareware". People like Jim Button and others developed word processors, communications programs and all sorts of other software on a shareware basis. It was distributed by BBS systems (which were themselves powered by shareware). Richard Stallman started the epic saga of Gnu. This early stage of free software had its adherents - and it too was a hot topic in the trade rags. Back then you could write a pretty decent word processor with one or just a couple people so it could be supported on a shareware basis. Over time this became less and less true, and shareware diminished in importance to just a few areas. In any area of consequence a for-profit, paid for software companies using normal channels took over the bulk of the market.

Fast forwarding to the present, the appearance of the Internet caused yet another temporary situation where a small number of people could create a very competitive product. Mosaic, Apache and Linux florished during this period, just as shareware did in the earlier era. Once again you could have a software product based on a very simple task - like serving up a file in response to HTTP.

I believe that most of the growth in Linux can be traced directly to these new Internet application areas where commercial software companies have not yet created products so demonstrably superior that they have obviated the niche occupied by free software. Although people claim that Linux is growing, my bet is that if you subtract out web servers and related new niches, the growth is much more modest.

If this is so, then we can expect that over the next several years commercial software companies will displace them because web servers will become far more interesting and complex. Straight HTTP will get ever more complicated and extended. This has happened with a vengence for the browser, and it will occur on the server too. If nobody can beat Linux and Apache with commercial products, then shame on all of us in the industry!

So, the pragmatic answer to the free software trend is to say WHAT TREND? Netscape's gambit to distribute source is too recent to say whether it will work or not. In the case of some other products there is a definite growth trend, but this is explained most simply as a transient effect where the Internet has made simple software viable, and along with it free software has become viable. This explains Apache, Linux and many other free software packages. If history is a guide, over time somebody will find a way to make money from these categories and the "trend" will reverse. However, even if it doesn't there is no evidence to date to suggest that this is something fundamental about software economics. It is more about the historical artifact that HTTP and other protocols are so simplistic.

As much as I like a smug dismissal like this, it does raise the philosophical question. What do we know about the software economics for free or cheap software? I will concentrate on system software, but much of the same arguments apply to applications.

Today, system software is priced between 5% and 10% of the price of the hardware it runs on. On smaller volume platforms that take a proprietary systems software approach (Apple, Sun, SGI and so on), the true percentage is higher, given that they must amoritize some fixed costs over a smaller base. This is compensated by the fact that proprietary hardware margins often subsidize the software.

That level supports the world we know today - which has (my guess) a few tens of thousands of people worldwide writing core operating system software. Several times as many write system-ish software which has a price level that is linked to the core operating system. It also supports the current user base of 150 to 200 million users.

Consider two alternative worlds - one in which the operating system is much cheaper - say 0.5% to 1% of hardware cost, and another in which it is tens times what it is today - at 50% to 100% of the hardware cost.

If you made this switch instantly, there would be some shocks to the system, but instead let's look at the steady state condition - as if the pricing models above had been long standing traditions.

In CheapOS world, many fewer people would be working full time on system software, because there would be no revenue to support them. Features and functionality which support the current user base would consquently be lacking. Which means that the user base would be much smaller.

Thus the total number of people working on system software is nonlinearly smaller - revenue wise there would be 10X fewer systems programmers per PC. There also would be many fewer PCs. Which means fewer variety of peripherals and other aspects of the industry. The number of systems developers would be reduced from our present world by much more than a factor of ten.

CheapOS world is a place which has a tremendously smaller user base,and a tremendously smaller computer industry. These days the tech sector drives the economy, but that wouldn't be the case in CheapOS world.

Linux fans and other supporters of "free" software might have some arguments against this.

First, they might say that there would be millions of developers dorking with the free source. The problem is that those incremental improvements done by small scale developers would not be available to the market as a whole, because there would be nobody to integrate and test the results. As we know, integration and testing does not scale gracefully. You can let a million people hack your code, but gathering the improvements together so that each customer can get the benefit of ALL the work is a mammoth task. We have a ratio of developers to testers of 1:1, so for millions of developers you'd better find millions of testers.

But it's actually worse than that, because the integration and testing needs to coordinate MUCH more than the developers do. You might be able to live with a million distributed developers (albeit with a lot of wasted effort), but integrating all their work is a single task, because any line of code may conflict in some unexpected way with some other line of code. This means you must have a highly coordinated effort of a million integrators and testers. I've made it sound extreme by saying millions, but the same logic works for other numbers - you need highly coordinated effort for integration and testing.

Presumably this is one of the things behind the Netscape free source code move (assuming they think this way). At any rate it is the thinking behind my suggestion a couple years ago at an exec retreat to do something similar with our browser. The slogan was "let a thousand browsers bloom" - in my notion by letting people develop extensions on top of the browser rather than hack the source.

Here's how it works. A thousand browsers spring forth from a thousand developers. However, amidst this field of low flowers would be only one tall tree - the scale player who could afford to do the integration and testing. In my version, where the customization would occur mainly above a set of binary components, the value created by a scale player in those components could flow to many of the customizers. Netscape's source based model is even more harsh - those who hack the source will have no way to integrate new Netscape releases automatically. In this model the free source lets the developers pick up some fringe or niche markets which Netscape wouldn't get around to addressing. However, it comes a a very high cost because changes to the Netscape code base will cause the developers to re-integrate their work. Meanwhile Netscape itself remains preeminent because they are the scale player within the context of their source. If that isn't enough, I suspect that the fine print on the terms of the source license also puts the fix in more directly.

Even if you could get enough coordination to integrate and test all that stuff, there is another problem that an economist would point out - all that work doesn't happen for free. The volunteer army of Linux developers, and the hypothetical integration and testing center, have some value on their time. Calling it "free" software is bogus - instead of paying money to a software vendor, there is a hidden cost in the time of the users, or their organizations. If you account for the total cost (including all the small developers, or the cost of the users making their own mods, or the cost of users finding bugs rather than testing finding them) then "free" software can get pretty costly.

Not only do you have to account for the costs - you must also recall that efficiency in software development depends on scale. Many years ago I wrote a memo about leverage in the software business. If you write software for yourself, then for every dollar you spend, you get a dollar's worth of software (assuming you are competent). If you have a company with N customers, it can afford to spend a lot more on development than the cost of of the software to any one user. So, from the user's perspective you get a lot of leverage. For a typical Microsoft operating system the ratio is over a million to one - for $100 you get software that cost $100 million (or more!) to develop. This miracle in leverage means that you have software much better than you could possibly afford to develop for yourself. It occurs, of course, because of the low marginal cost of producing intellectual property.

If you take the same number of developers and spread them across many small software houses or end user development, you start losing the leverage because each individual dollar spent on development gets distributed to only a small part of the market. The million to one leverage is the flip side of the massive integration and testing effort to create a uniform product. You ONLY get the low marginal cost of distribution to incremental users if all the users get the SAME product. Which means the features for those users must be integrated and tested together.

There are other scale effects as well - management is an example. An organization can concentrate very skilled talent - if every user hacks their own OS, very few of them will manage the work as well as a professional would. Netscape has every economic reason to attract better managers than any of the companies hacking on their browser source.

There are various other defenses one can mount for CheapOS world. Many of today's systems programmers compete with each other because making an operating system is a good business. In CheapOS world you could postulate some sort of socialist ideal where the OS is some Linux like public domain thing. Thus even though CheapOS world has many fewer developers, they are all behind one product.

There are many problems with this. Competition is intense in today's operating systems and that does drive a lot of innovation - losing that competition is not necessarily an efficiency improvement unless something else starts to motivate the developers - for example they could compete in enhancing the public domain OS, or in supplying new features as middleware on top of it. Here again we run into the integration problem - competiting enhancements in effect split the market, even if based on common source (the history of UNIX is a classic example). Competing middleware does too.

Maybe a set of super smart programmers would write operating systems for the good of humanity. Some undoubtedly would (Richard Stallman is an example), but its hard to have confidence in this occurring at the necessary scale The Soviet Union ran the experiment for 70 years and had a rather unambiguous result. I was just in Russia, and I can attest to this.

So, CheapOS world is a nightmare. The system software industry is sucked dry, undermining the foundations of computing. It surely is not an idyllic vision of the future. Since users vote on such actions with their purchases, I think that it is unlikely that we will see it. Only some draconian force - such a misguided government - would put the industry in this sad situation.

What about PriceyOS world where the OS costs 10X as much as it does today?

In the short term it would be a windfall for software companies, but the short run is not the relevant topic. The current discussion is limited to the steady state after PriceyOS world had been around a long time. I believe that you could create a stable situation where we could spend 10X more money developing our products than we do today, in return for 10X greater revenues (and the same margins). We would need to do this because our various OS competitors would have access to those kind of revenues too, and over time we would reach some equilibrium where we all spent like crazy developing software.

It may seem wildly unrealistic to say that we could scale up development even if revenues supported it. How could we possibly manage teams 10X as big? Easy is the answer - just look at history. In thje last decade we HAVE increased our development spending by more than this amount, even if you look at a single product. Each step happened incrementally, and we adjusted to it. If somebody had told us then that we'd scale up by 10X we would never have believed it - but we did it all the same.

If we had grown in up in PriceyOS world, over time we would have done the same thing. Indeed if you reject the notion of being able to spend 10X more on development, then you must think that the industry will die soon, because in a few years we will be spending 10X more than today (exponential growth is like that).

We wouldn't be the only ones to adapt; various parts of society would have to change too. Increasing development resources by 10X would put talent at a premium. Which means salaries would rise, creating more incentive for CS grads. The industry would reach out to international developers in India and elsewhere even more than today. Again, this will happen in the near future - as China, Russia, India and other large countries get more software work, and thus more incentive, it is only a matter of time before there are 10X more programmers on earth than today.

Even in the US there would be increases. It might seem crazy to think that there could be 10X as many programmers, until you look at the statistics of how damn many lawyers there are. It might take a while to reach a new equilibrium, but somethnig like PriceyOS world is almost certainly possible.

This amazing increase in development resources would get spent lots of ways. Lots more features. Lots better integration, testing and lots fewer bugs Lots more niche support for features that do not make the priority cut. We might have teams of human code optimizers tweaking assembly language. We might have whole concurrent teams turning releases sooner because they would develop in parallel. There would be many ways we could spend a vastly larger development budget.

Actually, we have an existing example of this. Intel has much higher profit margins than we do on its CPUs, and takes a much larger chunk of the total cost of a PC than operating systems do. Given their amazingly high revenue per PC, in a real sense Intel is living in PriceyCPU world - a close cousin to PriceyOS world. And, as a matter of fact, Intel does ALL of the things mentioned above. They do support for niche features (like MMX and other examples). They do a lot more hand optimization of critical circuits than we do for critical code. They do lots more simulation and testing. They have multiple implementation teams for different chip families. We used to have two main thrusts with Win 9X and Win NT, but increasingly we are moving to one, because we feel that we cannot afford huge parallel efforts. Intel does however, and with Intel style revenues per PC we could, and in think in the long run would, tend to support at least three parallel efforts of comparable complexity. Just as Intel does.

You might say that this is because they are in the hardware business, not because they have much higher revenue. Surely software is different. I think this is backwards. Hardware can be done with low margins - as many chip makers know all too well. In those cases you don't find Intel style spending. There are some obvious differences between hardware and software, such as the capital requirements for building FABs. However, I think that in most cases even these can be handled without breaking the basic analogy that a PriceyOS company would be a lot like Intel.

If you reversed the revenue roles, and gave Intel the same revenue per PC and margins as Microsoft, and Microsoft the Intel revenue and margins, life would go on. Intel would produce fewer new chips, and Moore's law might happen a bit slower. In compensation, some of the development spending at Microsoft would go to size reduction and performance enhancements.

PriceyOS world would have much better software than we have today, just as we have much better software today than we had in the past. It seems pretty good, but there is a major flaw. You can't keep raising the total cost of buying a PC without eventually effecting the market. Demand elasticity is not infinite, and a world where the OS cost 100% of the hardware cost, would in effect double the price of a PC, which would cut the potential market and rob the whole process of some of the scale effects and leverage that is so powerful.

So, it would seem that the optimal pricing for system software is some balance between demand elasticity with respect to price on the part of users, which strives to keep the cost low, versus the benefits of greater development spending. The optimal value is thus somewhere between where we are today (where the system software cost is demonstrably not high enough to impede user demand) and something a bit more expensive. In this view, PriceyOS world is suboptimal because a 1000% increase in the price of the OS would put a damper on demand.

Or would it? We have been innundated in the last couple years with studies of the Total Cost of Ownership that claim that the actual cost to most users of PCs is vastly higher than the cost of the hardware and software combined by a large factor - say 2100% of the basic hardware cost (i.e. $14K a year for three years for a PC that costs $2K).

You could increase the cost of the OS by 10X and it might not matter if the consequence is that TCO was lowered a sufficient amount. Even a 10% reduction in TCO would let you fund a 10X increase in the price of software and still give the end user a substantial discount versus today. It entirely reasonable to postulate that if you increased the software development budget by 10 TIMES that we could reduce TCO by 10%. Indeed, the NT5 plan of record should reduce TCO by more than 10% for a much more modest increase in development costs. Yet we aren't planing anything like a 10X price increase for NT.

There are many flaws with the TCO studies, but nobody disagrees that owning and using a computer does have hidden costs. So even if the numbers are too high, the principle holds that the cost of system software (or other software for that matter) is a very small percentage of TCO. As a consequence,small improvements in TCO could allow you to price software much higher - and still give customers a BETTER deal than the get today, even if the numbers are not what Gartner claims.

Above I argued that distributing source code to customers for them to hack for themselves was inefficient, because the "free" software just incurred lots more end user costs. These costs were large in aggregate, and furthermore are inefficient because distributed development lacks scale leverage and can't concentrate talent.

Just like TCO. The work that end users, system integrators and other perform as part of their TCO chores is just as distributed and just as ineffecient as if they were all developing the system. In fact, if you count TCO, then our CURRENT software pricing isn fact a CheapOS scenario!

From that perspective, we would be much better off charging a lot more for software, IF (and only if) in return it could solve all the TCO hassles. Those hassles would be better managed, and software which solves them could achieve vast scale leverage, rather than being the domain of the PC guy down the hall.

So, if anything recent experience suggests that the trend should be toward more expensive software, not cheap software!

Of course, there are several objections to this. The first and biggest is that users MIGHT percieve TCO differently than retail cost. Suppose that there were two PCs in the local shop - one costs $2K and has today's level of hassles, the other costs $15K (with a lot going to software annuity) over three years, but has radically lower TCO. Would people buy it? They certainly should if they figure TCO costs, but it does not mean they would. Humans are notorious about subject valuation - which is why nuclear powerplants are much more scary to people than driving to work - even though the commute is demonstrably more dangerous.

Even if in the long run it would be more efficient, the bootstrap period in which the perceptual view would dominate would be hard. Big MIS organizations that track TCO might be the first to switch. It might take a very long time for home users or other parts of the market to catch up. One reason is that TCO average do not apply to many customers. It is much better for them to buy a flaky machine if their pattern of use does not expose the flakyness, or does not do so often enough that it is a problem.

More generally, there is an interesting phenomenon that ANY system software pricing model tends to get stuck in a rut. If we lived in a competitive CheapOS world, it would be hard to raise prices to the current level. Similarly, it would be hard for us to raise prices today, even if you could prove that the benefit (from TCO or other means) was worth it to society Over a long period of time it could be done, but only with a great deal of effort. The world tends to equilibrate around a certain level of software pricing, and resist changes even if they are for the better with respect to the economics of BOTH customers and vendors. Change is not impossible, but it has to be done in a very clever way.

Bill's famous open letter to hobbiests posed this dilemma - saying that he'd love to have people pay for software because then he could afford to hire some more programmers and make the product better. It's time to admit that it worked out pretty well!

However, today you could equally well write the open letter saying that we wished people would be willing to spend a lot more money on software so we could cure the TCO problems, or promote better ease of use, or be able to take on new tasks and advanced features and functionality. Even as far as we have come to date, the main theme of the letter is still valid - software has yet to bottom out in its potential. In the current climate I don't advise posting such a letter, but the amazing truth of the matter is a world that invest substantially more in software would probably be a lot better for everybody.

Nathan