O'REILLY
ONLamp.com
LAMP: THE OPEN SOURCE WEB PLATFORM

Calculating the True Price of Software

by Robert Lefkowitz
07/21/2005

The first financial trading application I worked on exposed me to an interesting financial engineering technique. An investment bank was taking AT&T stock (footnote: AT&T used to be a telephone company) and selling two synthetic derivative instruments. You could buy what it called the *prime*--the AT&T stock stripped of the dividend. You could also buy the *score*--just the dividend stream, without the stock.

Now, in reality, it isn't possible to have one without the other, but this bank inventoried the real stock and sold these synthetics, which were really contracts to pay you as if these things really existed. These synthetics were traded independently. It isn't too difficult to see that, if you looked at the price for buying the prime plus the price for buying the score, it should equal the price of buying AT&T stock. That is to say, if you bought the stock, you would get the stock with the dividend. If you bought the "stock without the dividend" and the "dividend without the stock," the sum should be the "stock with the dividend." Because they traded separately, sometimes they would get out of sync, and you could make money buy buying the cheap one and selling the expensive one. This activity--looking for price differences between two things that ought to be exactly the same--is called *arbitrage*.

The reason "rocket scientists" (physics majors) were in such demand on Wall Street in those days is because physicists spent their time breaking down what appeared to be elementary particles (such as neutrons) into subcomponents (protons, electrons, and neutrinos) that might not even exist as independent particles (quarks)--except they did it theoretically. Financial engineers like to look at the price of something that appears to be an elementary particle and break it down into possibly imaginary components, which they can price separately and then see whether it all adds up.

Now might that work with, say, an Apple computer? A 17-inch iMac was $1,499 at the Apple store the last time I checked. You can purchase an extended warranty, AppleCare, for $169. That warranty is for years two and three; year one is included. AppleCare also includes extended telephone support, but I'm going to ignore that for now to simplify things.

After a quick trip to Wikipedia's page on failure rates (leavened with anecdotal rumors), it is not unreasonable to suppose that computers experience more failures in their first year than in the subsequent two years. The overall failure rate for computers runs about 15 percent--Macs do better than average. Still, it is not unreasonable to suggest that the curve looks roughly something like 8 percent failure in the first year, 4 percent in the

second, and 2 percent in the third. That means the first-year warranty is worth about $225. So really, that 17-inch iMac costs $1,274 for the computer and $225 for the first-year warranty.

But wait, there's more. Because consumer sites tend to be leery of extended warranties, I initially balked at buying AppleCare, but I learned that I had the option to buy it anytime during the year. However, once the warranty expired, so did the option.

Apparently, along with my computer and its warranty, I was also buying a one-year call option on AppleCare. What's that worth?

In 1973, Myron Scholes and Fischer Black developed a model for how to price options, which revolutionized financial markets. The trickiest input into the formula, and one that has a significant impact on the result, is the volatility of the price of the underlying asset. If there is no volatility (that is, the price doesn't ever change), then the price of the option is really just the interest rate discount for not having to buy it right away. Things get more interesting when the price fluctuates.

In the case of AppleCare, the price doesn't change. For a 3 percent interest rate, the value of that option should be around $5.

Still, logically, that iMac pricing is really a $1,269 computer, a $225 one-year maintenance contract, and a $5 call option on an extended warranty. (For the record, 360 days after I bought my Mac, I exercised the option and bought the extended warranty.)

Let's try to apply similar thinking to software licenses--in particular, enterprise software licensing. We can break down what appears to be a price for a single "asset" (the software license) into its component "quarks."

The conventional wisdom is that you buy a software license (the value of the actual software bits), and then you buy maintenance and support separately, which usually costs 20 percent of the original license cost annually. For a $1,000 software license, you'll pay $200 per year for maintenance and support.

What happens if we decide to separate the "stock" from the "dividend"? Could we price the "software without the maintenance" separately from the "maintenance without the software"? It's much the same as with that AT&T stock--even if logically the dividend always comes with the stock, a middleman might be able to sell them separately.

Now, in this case, the maintenance and support is already priced separately from the license. Doesn't that mean we're already done? Perhaps. Let me suggest that even though you're buying and paying for the maintenance separately, there is an option embedded in the license. When you buy the software license, it includes "options" to buy maintenance. (I use *options* in the plural, because I'll look at an option for each year of maintenance.) Let's separate that out. That is, we want to price the license without the options to buy maintenance and the options to buy maintenance.

When I informally polled enterprise software buyers about what they would pay for software given that they wouldn't be able to buy any maintenance for it (as a middleman,

I'd be selling that to somebody else), the universal response was that they would pay much less than the license--implying that the option to buy maintenance was clearly a significant fraction of the price. It is also the case that people expect software maintenance prices to be subject to change. Certainly, it has been the historical record that large software companies do change their pricing on maintenance occasionally--sometimes substantially.

How then to quantify the volatility of maintenance prices? Let's try a shortcut. It turns out that over the last year, the implied volatility of the Nasdaq (there are options on the Nasdaq index) has been running about 30 percent. We'll use the implied volatility of the Nasdaq as a proxy for the volatility of software prices--under the theory that the volatility of the Nasdaq captures in some way the volatility of pricing in the tech world. This is as good a place as any to start; we'll come back to the implications of alternative values in a few paragraphs.

Let's normalize the values to a $100 software license and say that a one-year option has a $20 underlying price; a year of maintenance is 20 percent of the license, so we'll assume it's worth $20 today. The *strike price* (what you can buy it for in a year) is also $20--a 5 percent risk-free rate. With all of those inputs, the value of that option is about $2.85. That is to say, for $2.85 you can lock in the price of the maintenance contract so that one year from now, you'll have the right to buy it for $20.

The right to buy the same maintenance for $20 two years from now is about $4.25; three years is $5.35; four years is $6.35. That takes us five years out. Assuming that you've locked in the maintenance over the five years to 20 percent of the purchase price, that set of options is worth $2.85 + $4.25 + $5.35 + $6.35 = $18.80. Five years is not an unreasonable horizon for enterprise software.

But wait, there's more.

Another option included in the license price is the option to upgrade to future versions at some price that will be less than the regular price. Right? That's clearly an option. The same informal poll of enterprise software users asked what they would pay for software when they didn't have the option to upgrade to the next release. The strike price is less standard than regular maintenance. (If you think about it, new versions perform "maintenance" by adding features as opposed to fixing bugs.) Now we're buying an option to upgrade in five years as part of this license.

The underlying price--the price you'd have to pay if you didn't have an option--we'll leave at $100. The next version will be priced the same as this one. Because you're upgrading, you have an option with a strike price of, let's say, $50. That is, you'll be able to upgrade to the new version for only $50. A five-year option for a $100 underlying price with a strike price of $50 and a volatility of 30 percent (with a 5 percent risk-free rate) is about $62.50.

Of course, most software offers new releases more frequently than once every five years--but enterprises don't like to upgrade very often and usually plan on skipping every other release in order to avoid upgrading too often. Then it would be two options. Much like adding additional planetary bodies to a problem in gravitational dynamics, the

complexity mounts rapidly. I'm trying to keep it simple. (I suggest follow-up research problems for interested students.)

At this point, the astute reader will have noticed that the sum of the value of the option for the upgrades plus the options for the maintenance is $18.80 + $62.50 = $81.30. That is to say, our $100 software license consists of $18.70 for the value of the actual software and $81.30 for options on future maintenance and enhancements.

This seems to correlate with the reaction I mentioned earlier that software without the option to get maintenance or upgrades is worth significantly less than the software with those options.

In fact, if we suggest that the maintenance is actually worth $25 instead of $20--but the option strike price of 20 is embedded in the so-called license--then the price of the maintenance options goes to $34.15, and the software itself turns out to be worth about $3.35.

The financially sophisticated may take a few exceptions. My interest rate is wrong. The volatility assumption is wrong. There isn't really an option contract, because those prices aren't guaranteed. Not only that, but they aren't really options, they're warrants. The distinction is that options are bought and sold by third parties--there can be a market in options without the participation of the owner of the underlying asset--whereas a warrant is sold *only* by the owner of the underlying asset. Pricing on warrants, therefore, differs from pricing on options.

Fair enough on all of these objections, but the point I'm trying to make is this. Those who have suggested that open source and free software is somehow not capitalistic, destroying the value of software and other such assertions, have missed this alternative explanation. It is just as likely that the free and open source software folk have stumbled across the financial engineering insight that a significant portion of the value of software is the embedded "derivatives"--options or warrants--on future maintenance and enhancement. Whether one believes that software has intrinsic value is related mostly to one's view on the correct value to use for volatility in calculating the option value. Larger values of volatility mean the software itself is worth less. Smaller values of volatility reduce the option price, and the software is intrinsically worth more.

Therefore, the major difference in worldview between open source advocates and proprietary software license advocates is explainable as a differing opinion on the correct value of the volatility of maintenance and upgrade pricing. People who believe that the pricing on maintenance is stable and unlikely to change see greater intrinsic value in the software. People who fear that the pricing is subject to large fluctuations see no intrinsic value in the up-front license; stripped of the options, the license value approaches $0.

For the open source movement, perhaps a better way to position the change that OSS is making is this: we're converting warrants on future maintenance and enhancements into options, which means that instead of having a sole supplier (warrants), we have created a third-party market (options) of these derivatives.

How capitalistic is that?

Robert Lefkowitz has spent 30 years weaving software for the airline, nuclear power, financial services, and telecommunications industries.

- What about freedom?
  2005-07-27 08:55:50 ddaa

  This essay is an interesting bit of economics, but it's not really talking about OpenSource, even less Free Software, as it only concerns itself with pricing.

  It is applicable to all software that is distributable and usable for free, that is proprietary Freeware, as well Free Software. It totally ignores how copyleft affects the economy of software.

  I will let more economically-versed people expand on how the right to modify, and redistribute modified versions of the software really removes the artificial scarcity imposed on software by proprietary licensing, and how it underlines the real scarcity: skilled programmers.

  Somebody had to chime in about this issue.
  - What about freedom?
    2005-07-27 13:16:33 r0ml

    Free speech is a freedom exercised by the press. The writers and readers both enjoy this freedom. Yet, there is still the necessity of determining economic models around selling newspapers. Or TV news programs. Concerning oneself with the economics of the news business doesn't mean one isn't concerned with freedom of the press. It is a different issue. And having grown up in a country where we didn't have freedom of the press, it turns out that a non-free press has very similar economic issues to a free one. One legislates the freedoms. The economics are less amenable to modification by fiat.

- The Manufacturing Delusion
  2005-07-26 08:36:33 Thomas_Barregren

  Your article clearly demonstrates what Eric S. Raymond called "The Manufacturing Delusion" in his essay The Magic Cauldron (see below). According to Raymond, "software is largely a service industry operating under the persistent but unfounded delusion that it is a manufacturing industry." He argues that this delusion encourages price structures that are pathologically out of line with the actual breakdown of development costs. With your article you have shown how Black-Scholes formula can be used to support Raymond's proposition. Very elegant. Very convincing.

  As Raymond point out in his essay, as well as Gregg Tavares in his comment on our article, the manufacturing delusion don't apply to quite all software. I believe Black-Scholes formula can be used to understand that as well.

The sale value of a software is determined both by factors intrinsic to the software itself and by factors outside the software. Examples on intrinsic factors that affect the price are the software's theoretical use value and the development cost of a functional equivalent. Examples of factors outside the software itself that affect the price are the availability of support, updates, consultants, training and third parties services and add-ons.

The price that most professionals and enterprises are willing to pay for a software is very sensitive to conditions outside the software itself. Consider for instance what happens with the maximum price that professionals and enterprises are willing to pay for a software when its vendor goes out of business or discontinue the development. The price will rapidly fall to near zero regardless of its theoretical use value or development cost of a functional equivalent. This sensitivity gives high volatility, which in turn gives low sale value by Black-Scholes formula.

But this sensitivity doesn't exists for some other software. Consider for instance what happens with the maximum price that consumers are willing to pay for a game when its vendor goes out of business or discontinue the development. The price will probably not be affected as long as the game is worth playing. This sensitivity gives low volatility, which in turn gives high sale value by Black-Scholes formula.

Finally, I want to point out some articles related to what you have written:

- *The Magic Cauldron by Eric S.Raymond*
  (http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/)
- *The Care and Feeding of FOSS (or, The Lifecycle of Software Technology) by Craig A. James*
  (http://www.moonviewscientific.com/essays/software_lifecycle.htm)
- *Open Source Paradigm Shift by Tim O'Reilly*
  (http://tim.oreilly.com/articles/paradigmshift_0504.html)
- *The Emerging Economic Paradigm of Open Software by Bruce Perens*
  (http://perens.com/Articles/Economic.html)
- *Strategy Letter V by Joel Spolsky*
  (http://www.joelonsoftware.com/articles/StrategyLetterV.html)
- *Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers! by David A. Wheeler*
  (http://www.dwheeler.com/oss_fs_why.html)

P.S. I hope this post get better formatted than the previous one. :-)

- free's real price
  2005-07-23 15:37:35 bigtreeman

  I sell free software for a nominal $30,
  + installation and training @ $80/h.
  + support online charged @ $80/h.

+ printed manuals @ 1.5 * printing cost.
(the manuals are installed, viewed through
a browser)

People won't buy it for $0
because they think it's worth $0.

I am selling support and training
and paper manuals.

- The Five Worlds of Software
  2005-07-23 15:06:57 gregg_tavares

  Joel talked about this
  http://www.joelonsoftware.com/articles/FiveWorlds.html

  Your article makes sense if we are talking about enterprise software. It makes less sense if we are talking about end user applications which are not expected to need maintenance. In fact needing maintenance would be a sign of a bad product. Some of them might have the option to upgrade although this is usually not in the contract when purchased, it is more of a discount after the fact to try to get people to spend money again. Finally, in games, specially console games, there is zero maintenance and zero upgrades.

  This is the biggest problem with discussing open source, we are all from different software worlds and we are dealing with our own environments. I can see pretty much 100% that open source is both well suited and commercially viable for enterprise software. Unfortunately, enterprise software is not the entire universe of software.

- Sorry, but I don't fully get your math
  2005-07-22 01:55:51 llogiq

  In your first example, you set the estimated first year failure rate for a mac at 8%, but compute a value of $225 for the first-year guarantee, which is about 15% of the computer price tag.

  Did you use the overall failure rate of 15% for all computers?

  Or is the guarantee worth more than the failure percentage times the price? I would assume it being less, since not all failures are killing the whole machine.

  On the other calculation, you completely lost me. (Ok, that's my fault, too. I was too lazy to look up the math. So be it.) I think the article might be more convincing if it included the steps to arrive at your results, at least in a sidebar.

  Otherwise good article. Thanks.