Delivering Material Shareholder Value with Eclipse and Open Source Software

Brent C. Williams
(temporarily) Independent Equity Research Analyst
brent_williams@hotmail.com
March 5, 2007

Agenda

- What defines shareholder value?
- How to measure shareholder value?
- How can companies affect shareholder value, in general?
- How can open source play to the multiple ways to drive shareholder value?
- What's the one step everyone forgets when trying to get open source to enhance shareholder value?

But first, we'll have a frank discussion about the big elephant in the center of the room that we can't ignore any longer…

What is a Commodity?

- Definition
  - A product where the buyer is unable to distinguish between products from different producers.
  - A black-and-white, universal definition of the product exists.
- Examples
  - Agricultural products: wheat, corn, pork bellies, orange juice, milk
  - Metals: gold, silver, steel
  - Energy: crude oil, electricity, natural gas.
  - Industrial products: nails, plumbing supplies, electrical wire, plywood. Sometimes even DRAM chips.

Structure of a commodity marketplace

- Large number of producers and consumers.
- Relative ease for a new producer to enter the market (capital investment, knowledge, licensing)
- Market structure can't be dictated or controlled by any one set of market participants.
- Well-defined product allows risk-free trade with multiple intermediaries between producer and consumer (market structure is flexible and adaptable).
- All acquisition costs are measurable and known: direct, indirect, contingent.

Behavior of a Commodity Marketplace

- No switching costs to buy from different producer
  - Standard product definition,
  - All acquisition costs known
- Market prices are a function of changes in supply and demand.
- Producers can't affect demand, only supply.
  - Farmer Bob can't increase demand for wheat by having a "season end sale."
- Pricing moves quickly to find a point of supply/demand equilibrium.
- "Excess" profits quickly disappear and producer profits revert to the mean of the economy as a whole.
- Lowest-cost producer wins in a commodity marketplace, because he can sustain "excess" profits longer than all other producers.
  - So economies of scale are key to winning.

Software Market vs. Commodity Market

Generic Commodity Marketplace	***Enterprise Software Marketplace***
• No switching costs to buy from different producer	• How many customers alternate database license deals between Microsoft and Oracle, just because?
• Market prices are a function of changes in supply and demand.	• Software supply is always infinite, since it costs zero to print the next copy.
• Producers can't affect demand, only supply.	• Producers in software *create* entire markets (nobody knew they needed a relational database until somebody invented one). • Producers drive their own demand by differentiating their products from competition.
• Pricing moves quickly to find a point of supply/demand equilibrium.	• How often do software companies update their price books?
• "Excess" profits quickly disappear and producer profits revert to the mean of the economy as a whole.	• Operating margins for software companies have always been above the level of the economy as a whole, regardless of industry growth rate. • Software industry operating profits are at highest level ever, relative to the industry's past.
• Lowest-cost producer wins in a commodity marketplace, because he can sustain "excess" profits longer than all other producers.	• Microsoft spends 25% of revenue on R&D. They are *not* the lowest-cost producer. • Everybody is making "excess" profits in the software industry.

Conclusion

If the software market does not behave like a commodity market, then software cannot be a commodity product.

Structure of Software Marketplace

- Producers control structure of the market -- you have to buy direct unless producer creates a tightly controlled indirect channel.
- Industry standards control only interfaces, not implementations
  - Interfaces are simple in commodity products, complex in software
  - Standards rarely if ever change in commodity markets because they don't need to. Definition of "8d framing nail" hasn't changed in 300+ years, because the "architecture" of a nail hasn't changed in 3,000+.
  - Standards frequently change in technology markets because underlying technology changes even faster.
- Non-conformance (i.e., "embrace & extend" strategy) with industry standards in commodity product makes a non-conforming product useless (ex: pipe fittings)
- Deep relationship between producer and end consumer benefits consumer.
  - Exceptions (such as third-party Microsoft Windows support) prove the rule.

So What Kind of Market is the Software Market?

- High-Risk Industrial Capital Equipment, a variant on standard Capital Equipment
- Standard Industrial Capital Equipment
  - High direct purchase cost (big printing presses, locomotives, etc.)
  - High indirect acquisition cost (personnel training, installation time, complex dependencies on other software in environment, extensive hardware purchases)
  - High lifetime operating cost, emphasis on reliability.
- Added dimension of extreme failure risk (i.e., high “contingent cost”)
  - Planes crash (can’t buy generic jet engine parts on eBay)
  - People die (radiation therapy machines can’t run amok)
  - Stocks plummet (last week’s software bug in computing the Dow)
  - Companies go out of business (any large Web site failure or airline reservation site crash)
- Added dimension of not being able to even measure contingent costs accurately
  - Fear replaces numbers as decision making criterion – “safe buy” perception

If It's Not a Commodity Market, Why Do People Still Believe In Commoditization?

- Industry standards will lead to "commoditization."
  - Commoditization in database business because of the SQL standard was first predicted in 1987.
  - Standards are *interface standards*, not *implementation standards*.
- Unit prices do not erode because of commoditization
  - Mostly a function of deal sizes increasing as enterprise deployments increase.
  - Also a function of "suite" bundles to deepen per-seat or per-server revenue.
- "We're always losing deals because of larger competitors' aggressive discounting."
  - Better to look at branding or other steps to counter, rather than "blaming" commoditization.

Do Interface Standards Compress Pricing?

- Hyundai & Lamborghini both support same “interface standard” for layout of driver controls (steering wheel, gas, brake, clutch, gearshift, turn signal, speedometer, tach)
- Hyundai has far greater financial muscle and low cost producer status versus Lamborghini.
- ***“Interface standard” is not “implementation standard”*** (horsepower, body styling, sound system, seat fabric, …)

2006 Lamborghini Murcielago. $279,000

1988 Hyundai Excel, $975

What About Interface Standards *Plus* A Really Nifty Reference Implementation?

- Reference implementation delivers enormous competitive value:
  - R&D cost savings (which should probably be reinvested)
  - Ease of gaining market share against “that other” platform.
  - Eclipse reference platform is in the control of Eclipse members.
- Reference implementation changes rules for success *slightly:*
  - Favors move to more “solution oriented” products (deeper core competency, higher unit prices)
  - Can leverage common infrastructures to create products targeted at clusters of verticals (not horizontal, not vertical, but “diagonal” functionality) giving some combination of volume and specialization.
    - This was the original (good) strategy behind Microsoft’s “Project Green.”
  - Integration assurance becomes an opportunity because platform popularity means wider variety of stuff to integrate; open source enables easy bug fixing, and stable reference port enables focus on integration testing of stuff that matters.
- Echoes the trend in applications – broad “platform” applications now adding deeper vertical functionality by acquiring specialized vendors
- Relatively unlikely to crush pricing unless vendors get complacent outside window of opportunity.

How the Lies About Commoditization Are Told

- Senior sales guys tell the CIO about commoditization
  - Reason: to get customers to sign broader strategic enterprise deals
  - Value of enterprise license: lock out competitors, increasing account control, making supplier switching difficult.
  - Another variant on “suites versus best-of-breed” argument
- But software companies never *really* abandon differentiation
  - The technical sales guys are fighting tooth and nail downstairs with technical decision makers to differentiate their products ...
  - …while the senior sales guy is upstairs spinning that story with the CIO
- Account control and recurring maintenance contracts are the real source of higher profit margins in large software companies, not “lowest cost producer” nonsense.

- Rookie Wall Street stock analysts tend to buy the commoditization argument, because they remember their Econ 1 classes in college.

Structure of Open Source Software Marketplace -- Is It a Commodity Market?

Generic Commodity Marketplace	*Enterprise Software Marketplace*	*Open Source Software Marketplace*
• No switching costs to buy from different producer	• Customers loyal to incumbent vendors	• Customers loyal to incumbent ***distribution*** vendors
• Market prices are a function of changes in supply and demand.	• Software supply is infinite (zero duplication costs).	• Software supply is always infinite (zero duplication costs)
• Producers can't affect demand, only supply.	• Producers in software *create* entire markets (nobody knew they needed a relational database until somebody invented one). • Producers drive their own demand by differentiating their products from competition.	• Producers create entire markets. • Producers drive their own demand by differentiating their products from competition ***(maybe in slightly different ways)***
• Pricing moves quickly to find a point of supply/demand equilibrium.	• How often do software companies update their price books?	• How often do **open source** software companies update their price books?
• "Excess" profits quickly disappear and producer profits revert to the mean of the economy as a whole.	• Operating margins for software companies have always been above the level of the economy as a whole, regardless of industry growth rate. • Software industry operating profits are at highest level ever, relative to the industry's past.	• Operating margins for **open source** software companies have always been above the level of the economy as a whole, regardless of industry growth rate. • **Open source** software industry profits are at highest level ever, relative to the industry's past.
• Lowest-cost producer wins in a commodity marketplace, because he can sustain "excess" profits longer than all other producers.	• Microsoft spends 25% of revenue on R&D. They are *not* the lowest-cost producer. • Everybody is making "excess" profits in the software industry.	• **Open source vendors may have slightly lower R&D costs than other companies, and may even see marketing benefits as well, on both open source & proprietary offerings.** • **Not sure yet whether** everybody is making "excess" profits in the open source software industry.

The Shareholder Value of Open Source

Stock Market 101

- What makes a stock price go up?
  - A growing stream of profits expected in the future.
  - Faster growth in the amount of profits → more value
  - Higher certainty that the company will achieve the expected level of profit → more value

    …ideally, you can do both at the same time.
- What levers are available for you to pull to generate a higher stock price?
  - More revenue at same operating profit margin percentage
  - More operating margin at same revenue (i.e., lower expenses)
    - Software companies tend to be good at doing both at the same time for structural reasons.
  - ***The big wild card: a higher multiple (i.e., P/E multiple or Enterprise Value/Revenue multiple) that investors will pay for a given level of financial performance (i.e., perceived quality of the future revenue or profit stream).***

Open Source and Profit Margins

Open Source is a great boost to profit margins

- Red Hat estimates that there is $1 billion in R&D powering the Linux ecosystem annually. It gets to take advantage of all of that by investing about $60 million in R&D.
- R&D expense reduction is where most composite vendors see initial benefits of open source.
- A thriving open source community can provide significant benefits in cost to find and fix software defects, especially after production release.

… But cost savings tend to be a one-time boost to stock price; once a new cost savings source is “in the stock,” it loses its ability to move the stock further.

How to Generate More Revenue (in General)

- "Economies of scale" type revenue generation
  - Sell product to greater number of customers
    (usually, hiring more sales guys or opening more sales offices)
  - Sell product in larger deals
    (usually, by expanding longtime customer relationships)
- "Agility" type revenue generation
  - Introduce new products faster than competition and win a greater percentage of deals
  - Create and exploit new markets faster than larger competitors, leading to erosion of their value equation

Open Source and Revenue Generation

Open source enables "Revenue by Agility" extremely well:

- Open source development model allows companies to create atomic products in existing categories faster
- Open source allows "innovation by integration," fast creation of new products by integration of piece parts from elsewhere, rather than requiring top-to-bottom development project.
- Innovation-by-integration enables new entrants to enter market easily with specialized niche or customized solutions, enhancing value of overall ecosystem and profits for niche vendors.
- New market entrants win, but ecosystem as a whole also benefits because ecosystem can attack new addressable markets (like healthcare), creating incremental opportunities for all ecosystem players.
- Open source process allows companies to commoditize selected bits of functionality that should be commoditized (IDE editors, class libraries) so they can focus on (higher margin) core competencies with less competition

Open Source and multiple expansion

- Show that you are poised to deliver multiple types of revenue upside at lower cost *on multiple product lines* because of your open source involvement.
- Show that open source gives you an architecture for driving more product innovations faster than competition.
- With a broad enough message of the benefits of open source to your product agility, investors will give you credit for products that you haven't even thought of yet (i.e., the Google juggernaut)
- Show that open source enables you to win market share against competitors that you couldn't previously win market share against.
- Success story: Actuate found that BIRT not only generated direct revenue but accelerated the growth of their high-end proprietary product

… In other words, your Eclipse involvement is about a lot more than just the direct revenue you're getting from Eclipse-based products & services.

The key to it all

You have to close the feedback cycle:

- Companies must communicate the tangible benefits of their open source strategy clearly, consciously, and frequently to investors.
- You need to work through Investor Relations, who has limited bandwidth to deliver the message to investors (short attention span problem on Wall Street).
- Need to be proactive at building a relationship with IR, especially if you're a small part of overall revenue stream. Get a senior management champion to help, if necessary.
- Need to make sure the message "fits" with the overall positioning that Investor Relations is using to talk to investors, or that they would like to be able to use (i.e., don't want to be a "value story," but want to be a "growth story" again).
  - What would happen to HPQ's multiples if the market started to believe that they finally "got it" about software?
- Need to deliver the message consistently over a long period of time before it becomes broadly accepted "market wisdom."
  - Once this takes root, multiples can expand significantly and sustainably.

Conclusions

- Commoditization of software is a lie, a canard, an urban legend, an oversimplification, an ignorance of economics, or all of the above.
- Commoditization of software is structurally impossible given the observable behavior of the software industry.
- Open source software is no different from proprietary software in market behavior, therefore it is the same type of good economically.

- Open source can affect shareholder's perception of the value of the enterprise by positioning the enterprise as increasingly agile and able to exploit as-yet-undiscovered revenue generation opportunities faster than competitors.
- Open source has far greater potential to affect shareholder value than simply as a cost avoidance tactic in parts of the R&D organization.
- The key to realizing that shareholder value is to make sure investors know what you're doing, and how Eclipse and open source enable that to happen.