Software Commoditization and Open Source Strategies

WORKING DRAFT

Cedric Thomas
OW2 Consortium
July 2010

The opinions expressed in this paper are those of the author and do not necessarily reflect the views of OW2 Consortium.

Software Commoditization and Open Source Strategies

Introduction

This paper looks at the background of open source and the tectonic drift that has positioned it center stage in the enterprise software industry. After essays on business ecosystems and platforms, I now explore commoditization, another key concept in the open source business environment.

In this paper[1], I contend that open source is one of three forms of software commoditization. I show that commoditization mechanisms provide the main explanation for the growth of open source in the enterprise software industry and point to two other commoditization forms: offshore outsourcing and, emerging today, cloud computing.

The scope of the paper is the enterprise software segment; however, I believe that most of the concepts developed here are applicable, with some adjustment, to other segments such as embedded software, telecommunications, gaming, high performance computing, etc. The analysis focuses on the relation between open source and commoditization and what it means for business strategies.

In this attempt to better understand open source's underlying business drivers, I have drawn extensively upon the work of many scholars to whom I am indebted. The paper also benefits from working daily with the dozens of organizations who are members of the OW2 Consortium; I hope it will help us all in charting our courses.

The paper has four parts. First, I briefly examine the foundation of the open source momentum, from the early days of the Free Software Foundation to the aspirations of today's open source software vendors on mainstream software markets. Then, I define what a commodity is and summarize different analytical frameworks describing the commoditization process. In the third part, I apply the commoditization framework to the software industry to determine the nature of open source. Fourth and last, I highlight several strategic options available to governments, software vendors, developers, customers and systems integrators.

A. The emergence and evolution of open source software

1. Definition and Origin

It all started with Richard Stallman's e-mail on September 17, 1983[2]. He announced his intention to develop a free operating system to be compatible with UNIX, the operating system, developed by AT&T, which was at the time popular with universities before becoming, a few years later, a de facto industry standard. He named it GNU, which stood for *GNU's Not Unix*. The key point in his message

1 This paper is an extended version of a presentation I gave about a year ago at the Exaptation 2009 International Workshop, http://users.unimi.it/exaptation2009/presentation.html .

2 Source: http://www.gnu.org/gnu/gnu-history.html

was that he said he would "give it away free to everyone who can use it." He then went on to add something that would prove to be equally important: "Contribution of time, money, programs and equipment are greatly needed."

Two years later, in October 1985, Richard Stallman established the Free Software Foundation (FSF) with the initial aim of raising funds to help develop GNU. Through the FSF, he proposed the rules that would define the free software movement. These rules are, in fact, four freedoms: a) the freedom to run the software for any purpose, b) the freedom to study how the software works and to adapt it, c) the freedom to redistribute copies of the software, and d) the freedom to improve the software and distribute the improvements. The FSF defined a license embodying these freedoms, the GNU Public License, now also known as GPL.

These freedoms, use the software without constraint, look at the code, change it and redistribute an improved version, are not allowed with proprietary software. These simple rules have had the power to change the very structure of the software industry. Today, the FSF promotes the universal freedom to distribute and modify computer software.

The GNU project's initial goal was to create a free operating system. By 1990, all the major components of an operating system were available except the kernel.

The next defining moment took place six years later, on August 25, 1991[3]: Linus Torvald sent an e-mail, quite comparable to Richard Stallman's, saying: "Hello everybody out there (...) I'm doing a (free) operating system (...)". Linus Torvald immediately adds "just a hobby, won't be big and professional like Gnu". By the end of the year, nearly 100 people had joined the newsgroup[4] and that same year Linux, the core, or kernel, of a free operating system was released. In 1992, Linus Torvald adopted the GNU GPL license which permitted the combination of GNU and Linux software thereby resulting in a full operating system now known as Linux but which really should be called GNU/Linux.

Another defining moment occurred in 1994[5] when Brian Behlendorf and some of the web's early system administrators, created a mailing list to coordinate and track the changes and improvements each of them were making to their http servers. On March 18, Rob McCool released Apache 0.2 based on NCSA httpd 1.3, and few years later, in 1998, the Apache developers decided to create the Apache Software Foundation. The rest is history.

2. Achievements

What started as a way to share the burden of code development quickly became a force in the software industry.

Apache has been the leader in web server platforms for years. For the last ten years, its market share

3 Source: http://en.wikipedia.org/wiki/History_of_Linux

4 Steven Weber, The Success of Open Source, Harvard University Press, Calbridge, Massachusetts and London, England, 2004, p. 55

5 Source: Text: http://www.unc.edu/~mohrmana/apache.pdf, http://www.linux.com/archive/feature/42466?theme=print and http://www.mail-archive.com/apache-docs@apache.org/msg01612.html

has been either consistently close to or well above 50%[6]. The world wide web would not be what it is today without the Apache server.

Adoption of standalone open-source software is accelerating. According to independent market research[7], the total market will be worth US$5.8 billion in vendor revenues in 2011, an annual growth of 26% over the 2006-2011 period. However, given that the code is usually distributed free of charge, it has been estimated that open source software successfully replacing acquisition of proprietary applications would save users more than US$60 billion annually[8].

Open source is more and more synonym with real money: companies "allocated up to 24 percent of their budgets to open-source software in 2008, up from 10 percent in 2007"[9]. No wonder investors have identified open source as a valuable investment opportunity. Between 1997 and 2009, since the first venture investment in an open source start-up, US$3.2 billion has been raised by 163 open source vendors through 378 separate funding deals[10].

On July 17, 2009[11], Red Hat was chosen by Standard and Poor's for inclusion in their stock index. Along with the Dow Jones Industrial Average, the S&P 500 is a benchmark stock index of major publicly-held American companies. It is also viewed as a reference for the American economy. The inclusion of Red Hat in the S&P 500 is a true recognition, first of the company's achievement and second, of the significance of open source in the economy.

3. Commercial open source

What started with Richard Stallman and the FSF as "a legal construct for cooperation and trade in intellectual property"[12] blossomed to become method for collective technology development with the Apache Foundation and, through start-ups and investors, evolved into both a market entry and competition strategy and an investment opportunity.

Just a quick point on definitions: there are important ethical differences between "free software" and "open source". The former refers to the four freedoms as defined by Richard Stallman and the FSF. Many software projects can still be considered free software, i.e. strictly abiding by these freedoms. The latter points toward a more business-friendly alternative which was gradually extended, complemented or wrapped into commercial software where "open" takes a different meaning than "free"[13].

---

6 Source: http://www.chotocheeta.com/2009/03/06/apache-vs-litespeed-time-to-switch/

7 Source: IDC, http://www.cio.com/article/116201/IDC_Open_Source_Market_to_Be_Worth_._B_by_

8 Source: Standish Group, Trends in Open Source, Study, April 2008, quoted in: http://www.zdnetasia.com/proprietary-vendors-lose-us-60b-to-open-source-62040484.htm

9 Source: http://news.cnet.com/8300-1001_3-92-1.html?keyword=IDC&tag=mncol

10 Source: Source: The 541 Group, 2008, http://blogs.the451group.com/opensource/2009/04/08/the-past-present-and-future-of-vc-investment-in-open-source/

11 See: http://press.redhat.com/2009/07/27/red-hat-included-in-sampp-500-index/

12 Van Lindberg, Intellectual Property and Open Source, A Practical Guide to Protecting Code, 2008, O'Reilly, Sebastopol, p.155

13 A difference clearly outlined in this comment: "I hate to keep asking this but are we talking here about Open Source Software (like in you can look but don't touch) or about Free and Open Source Software (like in our software will not enslave you) ?" Anonymous comment on

Although it can be argued that free software has always had a commercial dimension, the term *commercial open source software* (COS) would be better suited to specifically describe the offerings of those companies who are competing against conventional proprietary vendors and thus contributing to changing the structure of the software industry by taking market share away from them.

The difference between "free software" and "open source" is not really relevant in this paper, and I could use the technocratic "FLOSS" – Free/Libre Open Source Software – acronym which was crafted indeed precisely to be inclusive. However I will use all terms almost indifferently but certainly more often the widely accepted "open source" as the generic term for "software that can be modified and redistributed by everyone, sometimes—but not always—with the requirement that derivative works be freely redistributable under the same terms "[14]

4. Intrinsic advantages and challenges

Before trying to understand the industry dynamics that fuel the success of open source, it is important to acknowledge that the popularity of the open source way of developing software grew initially on the basis of its intrinsic advantages.

First of all, free software makes sense. Of course there is a lot of profit to be made from selling software but, from the perspective of an IT project, software itself represents only a fraction of the cost. Independant market analysts[15] show that software represented only 4.6% of the cost of an IT project from the point of view of the end-user or systems integrators. A very low percentage, it is well within the negotiation margin. Removing the cost of software altogether is a way to both surprise a customer and start working on what really brings value.

We could argue that open source is efficient in a number of ways: economically, technically, strategically and socially. Open source is economically efficient because the value of the software code given away is balanced by the value of the contribution to the project made by outsiders in terms of free code, free expertise and experience. Moreover, freely accessible downloads help build both awareness of the software and market share. From the technical point of view, a long-time argument in favor of open source is its technical efficiency. The open source collaborative development model provides quality to the code through peer reviews and extensive testing, and it ensures that the code is state of the art because it relies on both global knowledge sharing and frequent debugging. Open source is socially efficient: access to the source code empowers local communities, it helps grow local expertise, promotes value creation by local software and service companies and generally reduces technology lock-ins. Last but not least, open source is strategically efficient: it enables vendors to improve their time to market and leverage open source business ecosystems, and it gives end-users control over their code base by allowing them to modify the software as they see fit and helping them avoid vendor lock-in.

---

http://blogs.computerworld.com/15167/is_open_source_dead_as_a_business_model

14 Karl Fogel, Producing Open Source Software: How to Run a Successful Free Software Project, O'Reilly Media, Inc., 2006, p. 233 Available on-line at: http://producingoss.com/en/legal.html

15 IDC (REF???)

To sum up, the promise of open source is to achieve *more* by leveraging resources from the developers' community, *better* building on others' ideas and experience and *faster* thanks to quick responses from the community.

The potential benefits and efficiencies of open source are not, however, always achieved, indeed, far from it. Publishing code is not enough to develop a community of developers and only a minority of open source projects actually make it to stardom. This is certainly why open source remains a challenge for many IT professionals who still prefer to see the downside of open source. Arguments against include quality that is not guaranteed because the code is provided "as is" (in fact, this is not much different to most proprietary code which a provided with comprehensive disclaimers); the perception of a latent legal risk because many open source licenses are unproven (not that legal battles in the proprietary world are easier to fight); the lack of responsibility because the code is collective and there is no-one to sue; the investment risk because open source solutions are often supported by small companies; the lack of stability because of frequent releases; the lack of long-term visibility and a roadmap as there is no strategic owner of the software; and, lastly, the lack of internal competency because self-training is not usually recognized by mainstream end-users.

B. The commoditization process

In the first part, I have tried to show that the growth of open source is fueled by its own momentum and by its own characteristics. However, open source as an industry phenomenon must be analyzed also from outside, from the point of view of advanced business concepts such as network effects, competition, business ecosystems and commoditization. In this part, I look at different evolution mechanisms which help to understand how commoditization occurs.

I contend that commoditization trends in the software industry provide an external explanation to the success of open source.

1. Defining a commodity

First of all, what exactly is a commodity? Why are some goods called commodities? Does the term apply only to products or is it also valid for services? The term commodity is generally taken for granted[16] and good definitions are almost non-existent. A workable approach to defining a commodity is as follows: "The word commodity is used today to represent fodder for industrial processes: things or substances that are found to be valuable as basic building blocks for many different purposes. Because of their very general value, they are typically used in large quantities and in many different ways. Commodities are always sourced by more than one producer, and consumers may substitute one producer's product for another's with impunity. Because commodities are fungible in this way, they are defined by uniform quality standards to which they must conform. These quality standards help to avoid adulteration, and also facilitate quick and easy valuation, which in turn fosters productivity gains."[17]

16 For instance, Andrew Holmes, in Commoditization and the Strategic Response, 2008, Gower Publishing, 2008, does not offer a definition of a commodity.

17 David Stutz, http://www.synthesist.net/writing/commodity_software.html.

Using this approach as a springboard, I would like to offer a formal definition of commodities through four key characteristics.

Common specifications: standards and high substituability
Commodities are "objects of utility" for which there is a general consensus among users, customers, experts, etc. on their physical and functional characteristics. They are expected to meet standards of quality which are shared by all products available on the market without any significant variations. Because of these *de facto* standards, goods present a *high degree of substituability* and competitors have only marginal opportunities for product differentiation. Professional organizations and standard bodies, in many sectors such as agriculture, mining and chemical, government agencies, outline *expected standards of quality* based on measurable attributes describing the value and utility of a commodity[18] in order to further facilitate comparison, substituability and commerce. A fundamental consequence: price becomes the only criteria for selecting between two competing offerings.

Not process specific: economies of scope
The second key characteristic of commodities is that they are *fodder* for a great variety of usages. The opposite of niche products, commodities are generic goods or services which themselves are inputs into many different end-products or services. This grants commodities the *greatest level of economies of scope* among all products or services, i.e. the cost for using a generic commodity across different usages is less than the cost of using a specifically designed and produced good or service for each specific usage. A direct consequence -and benefit - is that incorporating a commodity into a product or service minimizes the addition to the end-user cost and provides greater flexibility for creating value (or building margin into the selling price) downstream in the value chain.

Mature products: minor innovations and multiple alternative providers
From the stand point of their production, commodities are mature goods (or services) which have already been through several cycles of technical adjustment and optimization. The general perception is that there will be no significant improvements in the production process or in terms of functionality. Only *minor innovations* are being made and no major breakthrough is expected in the different technical combinations available to implement the production process, at least within an accessible time frame. The process is stabilized and is shared knowledge. Thus replicable, there are *multiple alternative providers*.

Volume trading: economies of scale
Because of high substituability between competing products, variations in the selling price of a commodity are contained within a limited range: when the price is above this range, the offering is not competitive and when it is below, the activity is not profitable. In conventional commodities industry-wide price leveling is often determined by specialized market places. Since they are subjected to external price and quality conditions (see *standards* above), companies in commodity markets are left with leveraging economies of scale in order to maximize their profit. Additional margin can only be derived from additional volume and this is why commodity value chains are characterized by significant economies of scale in production, transaction, transportation and storage costs.

18 See, for example, the United States Department of Agriculture (USDA) Quality Standards at http://www.ams.usda.gov/AMSv1.0/standards

These four characteristics: high substituability, economies of scope, technical maturity and economies of scale all define a commodity. And it is important to highlight that, combined, they provide a structural framework allowing permanent downward pressure on the selling price of commodities which explains why commodities are often associated with low prices.

2. The commoditization process: life cycle approach

Goods (and services) are not born as commodities: coffee, chocolate, tobacco, for instance, were well sought-after luxury goods before becoming commodities: oil was polution for agriculture before becoming a commodity. Having defined what a commodity is, we must now understand how commodities appear on the market. Commodities are the result of a process. The analysis of the commoditization[19] process falls within the broader theme of industry evolution. The underlying logic is that products and industries go through the different stages of a maturation process and, as this process unfolds, the conditions for conducting business also change. Here is a quick overview of three analytical tools available to describe the changes that take place during this maturation process: product life cycle, the technology adoption life cycle, and the buying hierarchy life cycle.

Product life cycle

The *product life cycle* and *product life cycle management* approach provide a broad framework for optimizing marketing strategies[20] and for analyzing industry evolution.[21] It is a simple conceptual framework often used to describe indifferently the evolution of a product, a market or an industry and relevant business growth strategies.

The basic theory postulates that sales grow along an S-shape curve in four different stages.

- In the first stage, or introduction stage, sales are slow, customer inertia has to be overcome and the strategic priority is to develop the market.
- The second stage, or growth stage, is characterized by the emergence of a mass demand, sales grow fast as the market penetration rate increases; competition also increases and vendors' strategic priorities are to grow market share through product and channel extension.
- This is followed by the maturity stage and sales growth slow-down: in this stage ,the target market becomes saturated and sales are determined by the need for replacement and the demographic growth of the target market.
- Lastly, in the decline stage, sales and profit decrease as market demands shift to new products.

The product life cycle describes how a product is launched on a market as an innovation and evolves

19 A brief but important note on the concepts used in this paper: although they often seem to be interchangable, commoditization should not be confused with commodification. The latter is the process by which "objects of utility become commodities", see Karl Marx, Capital, Volume 1, Penguin Books, 1976, Harmondsworth, Middlesex, England, page 163, or, in other words, acquire the Marxist commodity-form and become exchangeable whereas the former is the process by which an already exchangeable good becomes available with the four characteristics of a (generic) commodity. Dave Stutz, ibid, for example, immediately after providing a pertinent definition of a commodity, chooses to refer to Marx and makes the confusion.

20 Theodore Levitt, Exploit the product life cycle, Harvard Business Review, 1965, November, page, 81-94

21 Michael E. Porter, Competitive strategy: techniques for analyzing industries and competitors, Free Press, 1980, page 157.

until it becomes mature with certain commodity-like attributes such as frozen features and low profitability. Despite its popularity, the product life cycle approach has been the object of much criticism, for instance, for being ill-defined, not validated by actual industry data and narrowly product oriented.

Technology adoption life cycle
While the product life cycle approach suggests a sequence of stages in the evolution of market demand and the appropriate strategies, the technology adoption life cycle approach identifies segments of users characterized by their attitude toward risk. The model describes "the market penetration of any new technology in terms of a progression in the types of consumers it attracts throughout its useful life."[22] It offers a bell-shaped sales curve and five customer segments:

- A new technology first attracts Innovators: this segment includes technology enthusiasts who would have been able to develop the technology themselves had it been necessary.
- Then it attracts Early Adopters: these are the visionaries who adopt the new technology in order to build a competitive advantage.
- The next segment is that of the Early Majority; these are the pragmatists who follow the example of the Early Adopters but at this point their benefit is uniquely in terms of productivity.
- Then come the Late Majority, or conservative users, who adopt the technology only to maintain the status quo and the Laggards, or skeptics, who would only adopt the technology under duress.

The technology adoption cycle is the basis for an improved version called the High-Tech Marketing Model. This broadly identifies two very different markets for every new technology, one the early market - with technology enthusiasts and visionaries - the other the mainstream market with the statistically dominant risk-averse customer. The behaviors of both markets are so different that success in the early market is no guaranty of success in the mainstream market. The gap between them is described as a *chasm* into which many high-tech companies have disappeared for lack of financial resources and expertise.

One of the main reasons for falling into the chasm is failure to recognize that early and mainstream markets buy different products: the mainstream market does not want to experiment, it requires proven and recognized standard solutions that tend toward commodities "fusing together in an integrated whole what the market has already come to endorse as the standard set of component parts"[23].

Buying Hierarchy life cycle
While the Disruptive Innovation framework describes how vendors' decisions change the structure of an industry, the Buying Hierarchy model explores the point of view of the customer and analyzes how changes in the supply-side of an industry are reflected by changes in the demand-side[24]. As an industry matures and meets customers' expectations for specific attributes that are the determinant selection

22 Geoffrey A. Moore, Crossing the chasm: marketing and selling high-tech products to mainstream customers, HarperBusiness Essentials, 1995, page 11
23 Geoffrey A. Moore, Inside the tornado: marketing strategies from Silicon Valley's cutting edge, HarperBusiness, 1995, page 87
24 In fact the Buying Hierarchy model was developed by a consulting firm, Windermere Associates of San Francisco, California, and popularized by Clayton M. Christensen, The innovator's dilemma, ibid, see page 216-219

criteria, customers move to new selection criteria in their buying behaviors.

The Buying Hierarchy life cycle model identifies four main decision criteria corresponding to four sequential stages: functionality, reliability, convenience, and price:

- In the first stage (new product) of a market, the main differentiating factor between products is functionality; in a market where all competing offerings are unequal in terms of functionality, customers are prepared to pay more for functionality.
- When the features and functionality are comparable across all offerings[25] and meet, or even exceed market demands, they are no longer a differentiating factor and customers shift to evaluating the reliability of a vendor and his product as their key selection criteria;
- Then, when market expectations in terms of functionality and reliability are saturated, customers shift to a new selection criteria: convenience (ease of acquisition, ease of use);
- Lastly, when all market needs are addressed and products are comparable in terms of functionality, reliability and convenience, then price becomes the main selection criteria.

At the end of this maturation cycle, the offerings are differentiated neither by their features nor by their reliability or convenience. And when offerings are highly substituable, the single differentiating factor is the price at which they are sold. They have become commoditized.

3. Commoditization process: disruptive innovation approach

The Disruptive Innovation framework

The Disruptive Innovation approach[26] provides a good framework to understand the structural changes in the software industry and the growth of open source software. The starting point of the analysis is the misalignment between the ability of consumers, or users in general, to absorb new technology and the rate at which established vendors launch innovations on the market. This creates an opportunity gap for new entrants with new offerings. At this point the theory identifies two broad families of innovations: innovations by established vendors who compete for market share are called *sustaining innovations* while innovations by new entrants who take advantage of the opportunity gap *disruptive innovations*.

The disruptive innovation framework works as follows[27]:

- New technologies always start with products and solutions offering a level of performance that does not satisfy the average user: initially, cars were noisy, unreliable and spilled oil, LCD screens were too small, etc.
- Some new technologies will fail and some will find niche users who will allow them to become profitable and improve until they reach a performance level good enough to satisfy the mainstream market.

25 "By 1998, Ray Lane, then the president of Oracle, was confessing that "customers can't find percent difference among SAP, PeopleSoft and us", quoted by Nicholas G. Carr, Does IT Matter: Information Technology and the Corrosion of Competitive Advantage, Harvard Business School Press, 2004, page 47

26 Clayton M. Christensen, The innovator's dilemma: when new technologies cause great firms to fail, HarperBusiness, 2000, New York

27 For a quick summary of the Disruptive Innovation framework, see: Clayton M. Christensen, Michael E. Raynor, The innovator's solution: creating and sustaining successful growth, Harvard Business School Press, 2003, page 32-35.

- They then become established technologies. Established vendors do not want to sacrifice their profits by competing on price; instead they compete by incorporating functionality and innovations into their products and solutions.
- As users become more sophisticated and better acquainted with new technologies, they are capable of absorbing higher levels of performance.
- Motivations being different, the rate at which vendors incorporate improvements into their offering is faster than the growth of the user's absorption capacity: at some point, offerings exceed the average user's needs.
- Meanwhile, as the functionality scope of the product category stabilizes, technology standards appear to enable the constitution of business ecosystems where some companies concentrate on developing modules or sub-parts of end-user solutions which are integrated and marketed by other companies.
- This gap creates an opportunity for a new wave of competitors entering the market with disruptive innovations, typically "simpler, more convenient and less expensive (...) that appeal to new or less-demanding customers"[28].

At this stage it is important to understand and to highlight that "disruptive innovations usually do not entail technological breakthroughs. Rather they package available technologies in a disruptive business model."[29] In other words "a key characteristic of a disruptive (innovation) is that it heralds a change in the basis of competition".[30]

Disruptive innovation and open architectures
A technology industry that is well advanced in its maturation process is one where leading, established vendors offer products and services with attributes in terms of functionality, reliability and convenience overshooting market expectations. In such an industry, established vendors have developed their positions by investing in proprietary innovations and integrated architectures.

The relationship between standards and innovation is ambivalent: "On the one hand, innovation and technical change challenge the existing stock of standards. On the other hand, standards may restrain innovation activities by freezing the technical state of the art."[31]

Moreover, while established vendors who dominate their markets with proprietary technologies set their prices at premium levels, new entrants following disruptive innovation strategies, based on open standards not owned by anyone, are able to bring éthe price as low as the marginal cost.[32]

Entrants leveraging open standards and business ecosystems are able to compete favorably against incumbent integrated vendors and their proprietary architectures: "Modularity enables the dis-integration of the industry. A population of non-integrated firms can now outcompete the integrated firms that had dominated the industry. Whereas integration at one point was a competitive necessity, it

28 Clayton M. Christensen, Michael E. Raynor, ibid, page 34
29 Clayton M. Christensen, Michael E. Raynor, ibid, page 143, note 5.
30 Clayton M. Christensen, ibid, page 219. The wording has evolved, the authors now prefer the term innovation which is broader than technology.
31 Knut Blind, The economics of standards: theory, evidence, policy , Edward Elgar Publishing, 2004, page
32 Suzanne Scotchmer, Innovation and Incentives, MIT Press, 2004, page 294

later becomes a competitive disadvantage."[33]

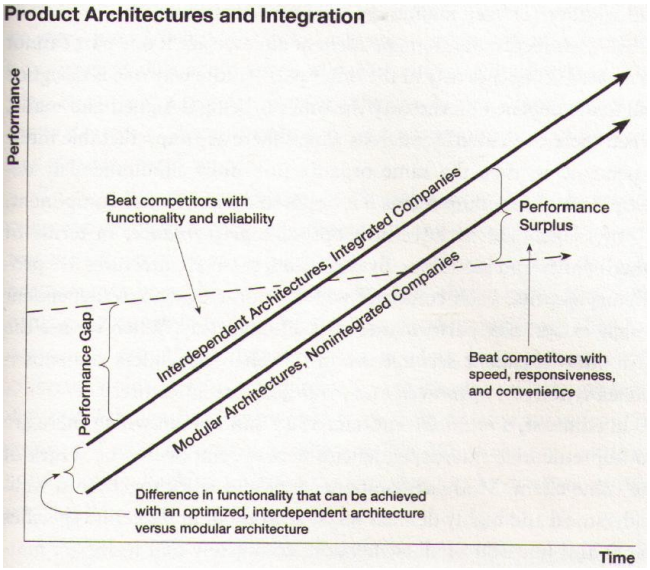

Figure 1: The Disruptive Innovation Model: Product Architecture and Innovation[34]

C. The nature of open source software

1. Commoditization of software

I define software commoditization as a process of transformation in the way software is developed, marketed and consumed under the influence of changing industry conditions. These changes include: slowing pace (asymptoting?) of technology innovation, increasing availability of production expertise, reduced uncertainty (i.e., increasing stability, convergence, growing atavism) in user needs.

The main indicators of commoditization in the software industry are, first, a lower rate of market growth and vendor consolidation due to maturation; second, the increasing significance of open standards in determining technologies and third, the emergence of disruptive alternatives to conventional offerings.

Signs of industry maturation illustrate life-cycle theories
Despite the software industry's ability to reinvent and revitalize itself on a regular basis, the growing trend in vendor consolidation and a slow-down in overall market growth[35] illustrate the life cycle

33 Clayton M. Christensen, Michael E. Raynor, ibid, page 134
34 Clayton M. Christensen, Michael E. Raynor, ibid, page 125
35 For example see: S. Sadagopan, 2005, at http://www.sandhill.com/opinion/daily_blog.php?id=6&post=20 and Jim Holinscek, 2009, at http://blogs.gartner.com/jim_holincheck/2009/12/23/predictions-business-applications-with-a-focus-

analysis and reflect a "maturation of the software industry just as it has been seen with “heavy” industries in the past century."[36] Enterprise application software, once the largest segment in the software market, is approaching maturity at least with large companies and early adopters, the key market sectors which usually help sustain growth in innovative applications[37].

What happened? Information Technology is now so widely used that few company can claim to use it to gain a strategic advantage. Most companies use the same standard software packages with little room for differentiation. Users do not have the choice, they are involved in a supply-side rationale where "the rise of packaged, generic programs supplied by vendors and shared by many companies"[38] is explained by the maximization of economies of scale in a capital intensive network industry. To some extent, IT has made the transition from strategic asset to standard, interchangeable commodity input which does not provide competitive advantage or differentiation – in the same way that the steam engine, railroads, electricity and highways all did before it. IT vendors are no longer able to control their customers and command premium prices that generate above-average margins. As a consequence, in many segments of the software industry "the traditional business model of enterprise software companies has matured. There will be very little further growth in perpetual license sales of proprietary software products."[39]

Rise of open standards

The software industry is a technology business fundamentally driven by innovation. Innovation, by nature, is not "standard" and most strategic innovations occur in areas where standards are not yet defined. More precisely, the software industry being essentially a network industry, strategic innovations are made of non-standard proprietary technologies leveraging existing standards in order to realize the necessary positive network effects. [40]

Established vendors in software market segments develop and defend their market share with proprietary technologies and architectures and intellectual property strategies based on the promotion of *de facto* proprietary standards. In this situation proprietary innovators in a given product category are able to set their prices at a level that maximizes the return on their R&D investments.

Despite the attractiveness of such product categories, innovation and IP-control represent significant barriers to entry, but only to the point when the industry matures: the functional profile of a given product category then stabilizes and *de facto* proprietary standards are in competition with open standards implemented by multiple vendors. These open standards focus "less on individual firm competitive advantage and more on defining rules for interoperability for a common architecture"[41] and "enable a modular subdivision of labor and thus a decentralized production of innovation."[42]

---

on-hcm-in-2020/

36 Report: "Towards a European Software Strategy", Working Group #2, page 22

37 Source: IDC, 2007, quoted by http://justinche.wordpress.com/2007/10/19/worldwide-software-market-drivers-2007%C2%962011/

38 Nicholas G. Carr, ibid, page 50

39 Source: http://www.sandhill.com/opinion/editorial.php?id=74

40 Ibid. An example: Adobe's "AIR" technology is a proprietary innovation leveraging the standards of the Internet.

41 Henry William Chesbrough, Wim Vanhaverbeke, Joel West, Open innovation: researching a new paradigm, Oxford University Press, 2006, page 122

42 Ibid.

OSS rivals in all product categories illustrate the disruptive innovation theory
Empirical observation tells us that commoditization and modularization are a permanent phenomenon within the IT industry. Hardware entered the commoditization cycle in the early 80s with the personal computer; from the mainframe to the PC, it is not the best product that dominates the market but simply the most standard one. And today, from desktop to mainframe, we have the ubiquitous Intel architecture.

Software entered the commoditization cycle in 1969 when Ken Thompson and Dennis Ritchie wrote a new file system called UNICS (Uniplexed Information and Computing Service) which went on to become UNIX, the main rival of proprietary operating systems. AT&T first chose to use the product for internal purposes, and then distributed it to computer research institutions such as Universities, ultimately leading to GNU-Linux.

The spread of free and open source software has been such over the last ten years that there is now no software product category, particularly in the enterprise computing area, where proprietary offerings are not challenged by open source offerings. Application servers, Business Intelligence, Enterprise Resource Planning (ERP), Portals, Enterprise Service Bus, Workflow, etc., open source market share is growing in every product category.

Open source is becoming the leading offering in several segments to such an extent that conventional proprietary vendors are disappearing. In the Content Management Systems segment, for instance, innovators Vignette and Documentum who created the product category and were highly regarded industry leaders, have both been acquired and in the Business Intelligence segment, leader Business Object has also been acquired.

2. OSS, one of three forms of software commoditization

The maturation course in the software industry leads to at least three forms of commoditization: offshore, open-source and cloud computing.

I define the commoditization process as the evolution by which an already exchangeable good (or service) becomes available to the market with the four characteristics of a generic commodity[43]. In the software industry, this process reveals itself in at least three ways: offshore, open-source and cloud computing. These are the three dominant forms of software commoditization.

Offshore outsourcing as the first form of software commoditization
The growth of offshore outsourcing is a direct outcome of commoditization trends in the software industry. Until the mid-80s, offshore firms had specialized in migration contracts, converting users' applications to new operating systems and hardware, mainly IBM's as a result of its ever growing market share. However, in the mid-80s, with the market shift from proprietary computing environments to open systems these offshore firms could develop into all other IT sectors. What happened is that the growing popularity of standard software development tools and the availability of sophisticated workstations for developers, meant that "the skills required by the market become less specialized and

43 See note 16, above

hence undifferentiated"[44]

As a result, many tasks, usually well-defined and non critical, such as data migration, software maintenance, application conversion (think of Y2K), documentation, etc., could all be outsourced offshore. The trends continue since "organizations no longer perceive IT to be adding strategic advantage and therefore are unwilling to pay such high fees or salaries for the services IT experts provide."[45]

Interestingly, offshore outsourcing is going through its own commoditization process. For several years India was virtually the only supplier of IT outsourcing services and Indian companies charged relatively high prices. Now, more than thirty countries compete for a piece of the offshore outsourcing pie.[46]

The continuing improvement of remote software development management practices and methodologies such as CMMI ensure that despite lower costs, the quality and reliability of offshore software is state-of-the-art. Offshore sourcing is today a standard input in any complex software system.

Open source as the current form of software commoditization
Enterprise computing segments in which open source is successful were first established some ten or twenty years ago. In these categories, leading open source software has the characteristics of commoditized software.

First, they thrive on established standards, particularly open standards, on substituability with incumbent proprietary software or with other open source software. Second, they are mainly available as components and platforms used to build end-user solutions: open source is to be found in infrastructure software (including middleware and generic applications) and in development tools. Third, few, if any, incorporate real innovations (the Apache web server is an exception although it itself started "as a fork of the NCSA httpd, a Web server created by Rob McCool at the National Center for Supercomputing Applications"[47]). Generally, they are open source interpretations, some highly creative, of existing offerings or functionality sets offered by established proprietary solutions. Fourth, being freely available for download and their value being proportional to volume just like any other commodity, their selling price is low. Indeed, the price can be as low as zero, an extreme position as far as low prices are concerned, when vendors do not monetize the software itself but the the size of its users' and developers' communities.

Moreover, open source software is a typical disruptive innovation in the software industry. The open source offerings that compete against proprietary offerings were typically introduced as "simpler, more convenient, and less expensive products that appeal to new or less-demanding customers"[48] including either those who could simply not afford the high cost of proprietary licenses or those who were willing

44 Andrew Holmes, ibid, page 35
45 ibid
46 Source: Gartner, http://www.gartner.com/it/page.jsp?id=565107
47 The Apache Software Foundation, Apache HTTP Server 2.2 Official Documentation - Volume I. Server Administration, Fultus Corporation, 2010, p. 14
48 Clayton M. Christensen, Michael E. Raynor, ibid, page 34

to support training, product integration and maintenance.

As illustrated by the following quotes from their annual analysis of operations, many incumbent companies find open source disruptive: "*If the open source community expands into enterprise application and supply chain software, our license fee revenues may decline.*"[49] "*Open source vendors' actions could also materially adversely affect our ability to generate revenue and maintain acceptable profit margins.*"[50] *"As the use of open source software becomes more widespread, certain open source technology could become competitive with our proprietary technology, which could cause sales of our products to decline or force us to reduce the fees we charge for our products(...)".*[51]

Open source software now is essentially the result of the conjunction of two elements: the first, "Free software", as established by Richard Stallman who had the revelation that software should be shared and formalized the rules (or *freedoms* as he calls them), the second, the leverage of the commoditization process in action within the software industry by opportunistic entrepreneurs who use the free software approach to establish a foothold in the market. This is the nature of the open source software market as we know it.

Cloud computing is the newest form of commodity provisioning

Cloud computing[52] is the newest form by which the software commoditization process reveals itself. Compared to conventional ways of deploying software, cloud computing's main innovation is that "the IT infrastructure no longer lies with the user, meaning that even inexperienced users can access these services. (...) Using clouds is more efficient and more flexible than the maintenance of internal IT departments."[53]

Cloud computing has become the generic term for a model where IT capabilities are provided as a service by specialized organizations to external customers using Internet technologies. It serves as a generic term for today's state-of-the-art paradigme for the provisioning of commoditized IT resources. These are usually described in three categories (or "layers"): commoditized computing resources, including storage, processing power and networks are provided as Infrastructure-as-a-Service (IaaS), commoditized software architecture and development environments as Platform-as-a-Service (PaaS) and commoditized applications as Software-as-a-Service (SaaS). Although significant innovation is still required for the implementation of a complete cloud service delivery architecture, it remains that cloud computing is a model of IT commodity provisioning.

The cloud computing model is well adapted to commodity software: it helps minimize the cost of both acquiring and running highly standard and substituable software, it allocates resources to fit many

49 Form 10-K for American Software, Inc, 07/14/2008

50 Form 10-K for DoubleTake Software, Inc, 03/13/2009

51 Form 10-K for BMC Software, Inc, 5/18/2009

52 In its definition, Cloud computing is directly related to concepts formerly developed in the iT industry such as Utility computing, On-Demand computing, etc. As for the origin of the term, opinions vary wildly, see, for instance the Wall Street Journal, March 26, 2009 at: http://online.wsj.com/article/SB123802623665542725.html I think it is to be found most probably in the fact that, for over ten years, a simple drawing of a cloud has been used to represent the Internet in millions of graphics and business presentations.

53 EU DG Information Society and Media - Directorate for Converged Networks and Service (2009), Report: "Towards a European Software Strategy", Working Group #2, page 7

different users thus maximizing economies of scope, it mitigates the risk in defining which functionality perimeter to provide by the maturity of the software provided and it allows global market reach to -potentially- maximize providers' margins in a context of minimized price.

Cloud computing customers are typical commodity buyers: they know what they want, they subscribe to a simple, good-enough application, they use the provider's APIs and underlying data models and constructs as inputs in developing their own solutions and they look for convenience and cost (applications are faster to get started, cost less), etc.

3. Commoditized open source software in today's context

The topic is far from being exhausted and further analysis is required today to understand what is going to happen to open source software over the next five to ten years. There are in particular three issues worth exploring: open source and innovation, open source and cloud computing, and the differences between planned and market economies.

Open source and innovation

Having established the nature of open source software in relation to growing commoditization in the software industry, one should not underestimate the relationship between open source and research and development; this would deserve a study in its own right. There is no question that there is a significant similarity between the way academic researchers share knowledge and the open source paradigm; however, the situation become a little bit more complex with regard to private corporate R&D.

A useful structural analytical framework[54] considers two broad innovation systems: one financed by agile venture capital looking for an exit by selling assets embodied in drastic patented innovations, the other, by bank loans and public subsidies where innovations are incremental, standard-based and knowledge is essentially open. Although open source seems to fit naturally with the open knowledge model, empirical evidences show that, at any given time, firms usually stand somewhere between both models, thus making the case for open source less intuitive. Moreover, current shrinkage of R&D public financing across western economies tends to push the cursor toward the first model.

Although many analysis have highlighted the positive relationship between public investment in higher education or public R&D and economic growth[55] (publicly financed open source development would result in positive externalities to the private sector in the form of knowledge spillovers), the current trend is for public funding to go more and more to the private sector R&D where its impact is debatable[56] and where the incentives to open source the results are lesser.

54 Benjamin Coriat, Olivier Weisntein,Coriat, B. & Weinstein, O. (2004) "National institutional frameworks, institutional complementarities and sectoral systems of innovation" in F. Malerba (ed.) Sectoral systems of innovation: Concepts, Issues and analyses of six major sectors in Europe Cambridge University Press, Cambridge

55 See for example, the often-quoted paper by D. Guellec, D. and B. Van Pottelsberge de la Potterie: R&D and Productivity Growth: Panel Data Analysis of 16 OECD Countries, 2001, OECD Economic Studies, 33, p. 103-126 and more recently: Jonathan Haskel, Gavin Wallis, Public Support for Innovation, Intangible Investment and Productivity Growth in the UK Market Sector, Imperial College London Business School, Discussion paper 2010/01, February 2010

56 See N. Serrano-Velarde, La Structure de Financement de la R&D des Entreprises? Une Analyse en Régression Discontinue, Séminaire REPERES, Paris, 21 Mars 2008

The fact that many publicly funded software R&D projects make available some of their results as open source[57] seems to be more the consequence of the de facto open innovation situation in which project consortiums find themselves rather than the outcome of carefully crafted strategies. The timely emergence of cloud computing provides an interesting environment to observe the relationship between open source and innovation.

Open source and cloud computing
Indeed, the relationship between cloud computing and open source will be complex.

First, of all, cloud computing and open source will compete as two different disruptive innovations providing solutions for the commoditization of software. An end-user who needs an application which happens to be commoditized will increasingly have the choice between an open source offering and a SaaS (or cloud-based) application. As both solutions will be economically substituable, users will choose on the basis of qualitative criteria such as security, exit barriers, etc. With regard to using commoditized applications, it is most likely that users will combine both open source and cloud computing solutions.

Second, cloud computing requires new software and will spur a wave of innovations; in fact it is more likely that there will be two waves. The first one, within the next two years, will be characterized by incremental innovations for centralized cloud services. Proprietary offerings already exist for centralized cloud architectures; publicly-funded projects are being set up to provide alternatives, and some of the results will be made available as open source. The second, will appear in three to five years, with rather drastic innovations addressing the requirements of decentralized cloud architectures. Many technologies for decentralized cloud computing are still in research stage and some years away.

It is not certain what role open source, and the open innovation business practices associated with it, will play in the development of decentralized cloud technologies. We can safely predict that many cloud architectures, whether open or closed, will be built upon open source components but we do not know to what extend the components which still need to be created will be developed in an open source way. In any case, open source and cloud computing will most likely interact as both complementers and rivals.

Lastly, it is most probable that cloud computing will have an impact on the very foundation of open source. Providing free usage of the software to users and developers alike is the way open source contributes to IT freedom. With cloud computing what customers get is a service, not software. Whereas IT freedom was broadly defined in terms of *software* accessibility, it must now be defined in terms of *service* accessibility. It is a paradigm shift which might make open source claims irrelevant.

Market economies vs planned economies
The commoditization mechanisms described above essentially reflect the evolution of the vendor/buyer relationship as we see them in market economies where free competition leads to a Darwinian selection process among vendors. In planned economies, however, the shift toward commoditized offerings might occur differently.

57 Source: European Commission, DG INFSO, unpublished mails, 2010

First of all, the evolution toward commoditizaton may be slower because of a combination of two factors: one is the inherent staying power of incumbent vendors whose business with buyers are protected by long-term organizational relationships and the other is the legacy of a business culture emphasizing efficiency and economies of scale over innovation-based entrepreneurship[58].

Another characteristic of planned economies is that the commoditization process tends to result from two movements, one top down, driven by government long-term planning, the other bottom up, determined by concomitant tactical moves of companies open to global industry trends. As a result the commoditization process in planned economies, or formerly planned economies in transition, is certainly not as fluid as in market economies and may hinder the adaptive ability of the national industry to be state-of-the-art. In our experience, it would be interesting to apply this analysis to the open source market in China.

D. Strategies for leveraging open source

1. Use the past to act in the present and change the future

Two drivers of the open source movement are, first, the aspiration for free software based on values inspired by the academic world and, second, the software commoditization process which creates opportunities for open source-based market entry strategies.

Despite evidence of imperfections and ongoing questioning[59], academic knowledge-sharing and peer review, or the process of exposing research, results or ideas to the scrutiny of experts for approval, criticism and elaboration, are key references for free software and open source. Free software originated as the adaptation to software of this academic process and, to some extent, it is not transferable to the average user. The way Richard Stallman explains and actually lives the free software ideal borders on sectarianism and utopia for all those without in-depth knowledge of computer technology.

Open source blossomed in another realm, the business world. It evolved out of the free software idea as companies adapted free software to the world of business in order to leverage opportunities offered by the software commoditization process. Over a period of ten to fifteen years, market leadership in entire software product segments shifted from proprietary to open source offerings. In this perspective, the success of open source is explained by the industry conditions: software commoditization is the driver of open source.

Inevitably, these two "moments" in the development of free software and open source are bound to combine into a third “moment” where open source, instead of being the result of external industry conditions, becomes, in its own right, a change driver of the industry. Open source supporting

58 See I. Grilo and A.R. Thurik, Entrepreneurship in the Old and New Europe, p. 83 in Enrico Santarelli (Ed.), International studies in entrepreneurship, Volume 12: Entrepreneurship, growth, and innovation: the dynamics of firms and industries, Springer, 2006

59 See for example: Arturo Casadevall Ferric C. Fang, Is Peer Review Censorship?, Infection and Immunity, April 2009, p. 1273-1274, Vol. 77, No. 4, Published ahead of print on 17 February 2009, (http://iai.asm.org/cgi/content/full/77/4/1273#FN1)

innovation and recognized as a channel toward market exposure for publicly funded R&D projects may even contribute to altering the structure of the software industry.

Recognizing this new pattern, industry stakeholders such as private companies, governments, software developers and customers may develop appropriate strategies. These include open source innovation, i.e. using open source *modus operandi* to promote (impose?) the result of R&D efforts, and *strategic commoditization*, i.e. voluntarily using open source to accelerate the commoditization of product categories.

2. Governments: strategic commoditization policies

Assuming the perspective that one of the ultimate goals of governments is to maximize social welfare, then public policy makers should adopt open source as one of their strategic levers for several reasons.

First of all, governments being among the largest consumers of information technologies, we can reasonably expect them to optimize the use of tax-payers' money by making a rule to avoid the traps of vendor lock-in strategies and choosing instead to leverage open standards and interoperability in order to keep costs under control[60].

Second, since a large part of research and development investment in information technology is funded by government money, it seems natural to expect that the results of these efforts be refunded to the community in the form of freely accessible open source software.

Third, as governments of most countries are the primary provider of resources to their education systems, a good strategy would be to require that all resources used in education be open source so as to guaranty shared and free access to state-of-the art technologies to all.

By systematically fostering situations where the advantages of software commoditization can be disseminated throughout society, policy makers have the possibility to anticipate and accelerate the software commoditization trends in a given economy: that is what I call *strategic commoditization*. What kind of Internet would we have today if, in the early eighties, TCP/IP, developed by DARPA, hadn't been freely disseminated along with BSD Unix to universities[61]? Examples close to OW2 include Trustie[62], the nationwide software development environement promoted by the Chinese goverment and the Brasilia Protocol[63] adopted by Brazil's public sector.

3. Software vendors: de-commoditization strategies

The *raison d'être* of all companies is to maximize profit and of course commoditization with its consequences is not good news for software entrepreneurs. Who would deliberately enter a market

60 See, for instance: Vivek Kundra: Federal CIO in His Own Words, http://radar.oreilly.com/2009/03/vivek-kundra-federal-cio-in-hi.html, or Kundra advocates open source, http://fcw.com/articles/2009/06/08/feature-open-source.aspx

61 See DARPA and the Internet Revolution: "Since TCP/IP was in the public domain – having been developed at taxpayer expense – other networks were beginning to use it as well, and to link with one another in an expanding Internet." at http://www.darpa.mil/Docs/Internet_Development_200807180909255.pdf

62 http://www.trustie.net

63 http://www.softwarelivre.gov.br/protocolo-brasilia-1

where vanishing differentiation, alternative producers proliferation and raging price competition are the rules of the game? Who would finance such ventures unless they see ways to do what all profitable companies seek to do i.e. maximize profit and share-holder value? And how can they achieve this without fighting off competition and developing sales?

I see three key de-commoditization strategies accessible to the open source software entrepreneur: they all aim at turning the tide and building defensible market positions and growing customer revenue. These strategies are not exclusive from one another.

The first de-commoditization strategy is to create uniqueness by extending a free product with a strong brand and specialist services. What is important here is not to decide if software is a product or a service[64] but to understand that when code is freely available for download and replication, the only way to make it financially viable is to add value through differentiation. The principle here is to consider that the base product can be given away because it represents only one part of the value brought to customers. Offerings that add value and differentiation include selling product-oriented services such as support subscriptions, premium training and integration or business process-oriented services such as product customization that enhance the usefulness of the base product. Among OW2 members, companies such as Ingres, JasperSoft and Red Hat represent this approach.

The second de-commoditization strategy for open source software vendors consists in introducing displacing innovations[65]. In other words, they can ride the commoditization process to their advantage by aggressively investing in open source innovation on sub-systems. As we have seen, the commoditization process generates modularization with standard interfaces and within standard architectures and, as modularization goes on, demand for greater performance shifts from the whole product to the modules. As a consequence, what happened with hardware also happens with software: competition shifts toward sub-systems components which must meet stringent requirements for product quality. "While higher level systems (...) become more modular, lower-level subsystems become increasingly integral. (...) Within each module internal interdependencies become more complex than ever." [66] At OW2, this approach is exemplified by companies such as ActivEon, Eteration, Orbeon, PetalsLink, Requea and Scalagent.

Then there is the broadly called, and these days much debated[67], "open core" strategy; which consists in marketing products, services and add-ons with customer (or partner) lock-in tactics in ways that create proprietary assets and, indirectly, reproprietarize the software. A borderline model between open source and proprietary software, it seems to be the strategy favored by venture capital investors. For reasons that are technically arguable, certain software modules and extensions and certain services,

64 See for instance: Eric S. Raymond The cathedral and the bazaar: musings on Linux and open source by an accidental revolutionary. First edition. Sebastopol, CA: O'Reilly, 1999, p. 145: "In other words, software is largely a service industry operating under the persistent but unfounded delusion that it is a manufacturing industry."

65 Clayton M. Christensen, Scott D. Anthnony, Erik A. Rorth, Seeing what's next: using the theories of innovation to predict industry change, Harvard Business School Presss, Boston, Massachusetts, 2004, p.14

66 Cornelius Herstatt, Hugo Tschirky, Christoph Stockstrom, Management of technology and innovation in Japan, Birkhäuser, 2006, p. 67

67 A good entry point into the "open core debate" is: http://blogs.the451group.com/opensource/2010/06/30/while-lies-beyond-the-open-core-debate/

such as engineer certifications or subscriptions (bug fixes, functionality and security updates[68]), for instance, can be contractually linked to software code and specifically packaged (often with add-ons) by or for the service vendor. Just as with proprietary software, tampering with the code might cancel the right to vital software support. This is the strategy implemented by growth start-ups such as BonitaSoft, Exoplatform and Talend.

In any event, the initial software product, whether a whole product or a sub-system, is made available to the public for free. This is a classic market-development strategy where, on the one hand, "free" is used to generate leads and, on the other, "the object is to get lifetime and multiple value from customers without allowing short-term recovery of cost to get in the way."[69] All open source software vendors share this quite important characteristic along with a genuine interest for cooperative strategies. Personally, I think the only straightforward open source strategy is to make honest money with quality service however, in order to be attractive to investors, many of them tend to end up operating just like conventional proprietary ISVs and develop their own IP-based assets.

4. Customers and Systems Integrators: non-alignment and best-of-breed strategies

Because commoditized software is available from several different providers, a best of breed approach is.the most efficient technology investment strategy for systems integrators and end-usres. Best of breed procurement consists in selecting the best offering for each need. The advantages of this method are well known: software selection is optimized because the investment is not tied to a single vendor and user utility can be maximized in each project. While vendors' roadmaps for products and features introductions aim at maximizing their own profits, best of breed strategies allow instead independent and timely decisions that maximize end-user benefit. Open source blends well with best of breed strategies - specially when considering commoditized software - and contributes greatly in enhancing their advantages. It helps minimize cost in most mature product categories with state-of-the-art open source offerings.

In this context, customers and systems integrators can be perceived to be in the same position vis-à-vis open source software to the extent that both categories of stakeholders have the same fundamental objective: they use technology to improve business processes and increase end-user competitiveness rather than technology vendors' profits. End-users make choices that will affect their competitiveness and systems integrators work for them and make recommendations that will impact their customers' competitiveness.

Best-of-breed procurement strategies ensure customers remain in control "Customer ownership" is a key concept in marketing[70] but which customer wants to be "owned", what could be the long-term consequences of such ownership over an enterprise ability to make its own decisions? Single vendor procurement, however comfortable this might be hands over some control to the vendor. At some level,

68 See for instance: Savio Rodrigues, Blog post, Proprietary Open Source http://saviorodrigues.wordpress.com/2008/01/30/proprietary-open-source/

69 Sandra Vandermerwe, Customer capitalism: increasing returns in new market spaces, Nicholas Breadley Publishing, London, 1999, p. 197

70 See for example: Howard W. Oden, Transforming the organization: a social-technical approach, Greenwood Publishing Group, 1999, p. 52: "In market terms, the company with the most intimate knowledge of its customer's problems and desires will own the customer relationship and the company which owns the customer relationship wins."

the well-known expression "No one ever got fired for choosing IBM." reflects the "difficulty that insiders once faced when trying to make a non-IBM" product selection[71]." Most companies already contend with multi-vendor information systems, because of structural changes or inherited legacy software. Moreover, it is probable that most organizations are already using open source software either as a "lower-cost good enough alternative to expensive proprietary software" or because "an open source software product may have reached a level of technical superiority that pushed along its adoption."[72] The most advanced customers actively contribute to open source projects they use. OW2 members representative of this approach include France-Telecom and the French Ministry of Interior.

Trustworthy systems integrators have the most to gain by providing assistance in the selection, integration, implementation and management of open source software in enterprise information systems. Thanks to open source, customers do not have to pay simply for the right to use a software product, instead they pay systems integrators for their services and the value they bring to their business processes. OW2 members such as Edifixio, Konsultex and Serli are examples of this positioning.

Open source represents a tremendous opportunity for systems integrators. There is a balancing act between them and end-users; while the former position themselves as trusted advisors to their customers, the latter need to develop their own technology knowledge so as not to become "owned".

5. Developers: community strategies

In commoditized product categories, possibilities for technology differentiation are limited and competitive advantage allowing for profit maximization usually not sustainable. Developers who work on such product categories seek other kind of returns such as maximizing their efficiencies and are driven by other motivations such as peer recognition, technical interest, etc. In this context, they derive maximum satisfaction by pursuing a community strategy.

Communities are voluntary groups of people sharing resources, values and working together to achieve a shared goal. Developers pursuing community strategies aim at developing a collaborative development model and at creating a virtuous circle for continuous improvement of the code. As testified by the experience of the Apache Foundation, community cooperation has proven to be a productive way to develop quality software.

Since "code written for sale is only the tip of the programming iceberg"[73] open source developers do not only work for software vendors, but come from all categories including end-users, systems integrators, universities, public and private research centers, etc. By sharing the code, an open source developer meeting a problem will hope that somebody else has probably encountered and solved it, or at least a similar problem and will provide valuable input. Or put another way: "Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix

71 John Katsaros, Peter Christy, Getting it right the first time: how innovative companies anticipate demand, Greenwood Publishing Group, 2005, p. 108

72 Maria Winslow, The practical Manager's Guuide to Open Source, Open source Migrations, 2004, p.16

73 Eric S. Raymond, The Cathedral and the Bazaar, Musing on Linux and Open Source by an Accidental Revolutionary, 1999, O'Reilly, Sebastopol, p.142. The developer's point of view is the real subject of this book.

obvious to someone[74]."

Another significant incentive for programmers to share their code is to increase their apparent productivity. Despite considerable advances in software engineering, actual programming is like performing a symphony or giving a haircut; productivity does not change over time[75]. Fortunately, reuse of third-party code is made possible by the open architecture of commoditized software and modern programming methods based on standard frameworks, platforms and libraries. Standards and software modularization provide the right context for mutual sharing between programmers. Playing the community game allows programmers to significantly increase their productivity by re-using stable code provided by others.

Possibly counter-intuitive for outsiders, but developers have a great incentive to play the community game. Joining a community may not only provide them with the opportunity to share code repository and development tools but it is also the only way to leverage the intelligence of others and grow third-party contributions: "instead of everybody writing his own version of the same program, the best version would be available to everyone, and everyone would be free to delve into the code and improve on that."[76]

The community game is most often played by developers working for end-user organizations or systems integrators who have either developed software not available on the market or are contributing developments to third-party software they use for their own needs. In any event they are not in the business of selling software. The Apache Foundation provides archetypal examples of community strategies. The submission to the OW2 code base of in-house developments by members such as Bull, EADS, INRIA, Sogeti, etc. and by CHOReOS, a project funded by the European Commission, are also examples of community strategies.

Conclusion

Established from the onset as an unconventional means to share code among developers, open source has evolved into a major structuring factor in the software industry.

This paper explores the nature of open source software as a business phenomenon. The open source model for developing code has its own intrinsic advantages – technical, economic, strategic and social. However, these alone do not explain why open source has become such a force – at least in the enterprise software market. The argument of this paper is that open source software, as we know it today, is essentially the result of the alignment of the long-term commoditization trend in the software industry with the vision delineated in 1984 by Richard Stallman. The emergence of open source in the context of the software commoditization process may be described suitably by refering to the disruptive innovation theory developed by Professor Christensen of Harvard University.

---

74 Ibid, p. 41

75 This is an extremely debatable topic but the number of debugged lines of code programmers are able to deliver seems to be stable at 10-20 lines per day. While modern software engineering increases the functionality delivered, the actual number of lines of code delivered by a programmer in a day remains stable.

76 Steven Levy, Hackers, Heroes of the computer revolution, Dell Publishing Co., New York, 1985, p. 41

The definition of a commodity is key to this paper. Applying this definition to the software industry, I contend there is a long-term commoditization movement which reveals itself through at least three business configurations: the first one is offshore outsourcing, the second open source and the third, emerging now, is cloud computing.

Open source, however, should not be reduced to an expression of software commoditization; it plays a key role in open innovation practices and its interaction with the wave of innovation required by cloud computing provides a new interesting field of anaysis.

The software industry is a system with strong feedback and in constant renewal. A correct understanding of the mechanisms underpinning the growth of open source and its relationship with software commoditization provides a conceptual and strategic framework for governements, software vendors, systems integrators, customers and developers. Hopefully this paper helps stakeholders develop a clear vision of the dynamics of open source in order to pro-actively leverage its benefits in their own strategies.