MN4267 Creative Industries Essay

How useful is a framework based on an analysis of capitals in evaluating the dynamics of a creative industry?

Niall Douglas (040007747)

One of the most economically important industries in recent decades has been the information technology industry which is comprised of semiconductor based electrical hardware being operated by software, a set of contextualised and embedded mathematics. In 2007 spending on IT was 2.5% of world GDP (US$1.24 trillion – IDC, 2007) with 21% of that upon software (US$260 billion – IDC, 2007). World Economic growth 1995-2003 was comprised of 45% increased capital inputs, 26% increased labour inputs (mostly through extra hours worked) and 29% increased productivity; one third of the increased capital inputs was spending on IT infrastructure – indeed, that makes 15% of world GDP growth being purely on IT capital investment and up to half total world GDP growth being due to IT (Jorgenson & Vu, 2005). One can literally say that IT has redefined the world over the last two decades, and of that computer software is the most important as it's much harder to implement – already, over half the employment of the IT industry is purely for software despite it only taking 21% of the spending (IDC, 2007), and 60% of the additional 0.3% of GDP growth in IT spending projected to 2011 is projected to go to software, not hardware (IDC, 2007).

Yet despite that IT has utterly transformed all the other creative industries – which make up around 5% of UK GDP (Department for Culture, Media and Sport, 2001) and is growing rapidly – I have seen very few analyses of the computer software industry itself from a creative industry standpoint. Most commentators seem to prefer the soft creative industries such as media (print, radio, film or television), art (crafts, music or videogames) or even a hard one like architecture. Yet where is the study of the creative industry which is a far bigger contributor to world GDP growth[1] than all other creative industries put together?

In fact, of all the computer software industry, about 42% is directly related to one company: Microsoft (IDC, 2007). I have found absolutely no analyses in terms of cultural capitals of this whatsoever despite that it must surely be an astonishing cultural phenomenon given that no one had heard of Microsoft only twenty years ago. I have found just **one** study applying Bourdieu's framework to open source software development which centres around the "gift economy" idea (Zeitlyn, 2003). I therefore thought it would be interesting to give it a try myself.

The Cultural Dynamics of Computer Software

I think it fair to say that the cultural dynamics of computer software can be split into two main philosophical camps: (i) proprietary and (ii) open source. Proprietary means that the software's source code and interfaces are kept mostly private and held internally by the company as a form of competitively advantageous trade secret, whereas open source believes that full distribution of source code and interfaces are more competitively advantageous. The former has the advantage that economic rents can be more easily earned as control is centralised with the owner, whereas the latter has the advantage that a much bigger army of bug testers exist who can fix the bugs themselves without any extra assistance, plus that code from other projects can be simply copy & pasted rather than having to be reverse engineered and thus debugging all over again – both greatly reduce development costs as some 80% of

[1] The other creative industries have a bigger slice of GDP, but then they have been around for much longer. If you work out the contributions to current world economic growth, IT and product design are the overwhelming contributors with media and art barely making a few percentage points – though, I daresay they underestimate the usefulness of information dissemination by the media (which isn't a final goods or services transaction and thus is ineligible for GDP, it's more of a public good).

software's development cost is bug fixing. Put in a greatly simplified nutshell, this difference is one of Money vs. Artistic Integrity as is almost always found in every creative industry – the "art for art's sake" principle as noted by Caves (2000). Another equally useful way of looking at the difference is as one of between corporations and cooperatives – both seek profit maximisation, but both go about it in very different ways.

Needless to say, both philosophies have their cultural icons. For proprietary, it used to be IBM but since the mid-1990's it has become Microsoft. For open source, it began with BSD (an open source implementation of Unix, an operating system) but transformed itself also around the mid-1990's into Linux (also an open source implementation of Unix) mainly due to a proviso in its copyright agreement that none of its source code can be used by proprietary software. Famously, large chunks of BSD (which still exists and is in active development) were simply copied as-is into Microsoft Windows and Apple Mac OS X and both Microsoft and Apple have derived massive profits while those who wrote BSD got nothing. This kind of selfishness and exploitiveness has generated great hatred of proprietary software, and I think it fair to say that a majority of computer software programmers are philosophically firmly in the open source camp even if their day jobs require putting on a proprietary hat (some 63% of open source programmers work for proprietary company – David, Waterman, & Arora, 2003).

Bourdieu has four kinds of culture capital: (i) Economic (ii) Social (iii) Cultural and (iv) Symbolic [(Bourdieu, Outline of a Theory of Practice, 1977) and (Bourdieu, The Field of Cultural Production, 1993)]. I shall now treat each in turn.

Economic Capital

Proprietary software commands tremendous economic power – Microsoft alone is well known to have a *cash* war chest exceeding US$40 billion which is bigger than that of General Electric, the world's largest corporation[2]. Partially this is because it is extremely expensive to develop software and despite that the barriers to entry are extremely low (the tools and hardware are very readily available), coordinating computer programmers is extremely tough as they tend to be very difficult to manage and weirdnesses exist such as adding manpower to a late project makes it much later (Brooks, 1975). In fact, as Brooks points out, most of the standard management rules of thumb must be inverted for computer software projects unless you want to lose a few hundred million dollars.

Despite this, Linux somehow managed to write itself, which for the first ten years was almost entirely by unpaid volunteers with no management structure whatsoever (unlike BSD, which has always had a control board). At standard programmer time cost estimates, development of Linux up until 1998 had consumed 40,000 man years at an effective cost of US$4 billion (The Register, 1998) – though, as it was volunteered time, it counted to GDP as tens of thousands of dollars. To develop a year 2007 version of Linux from scratch knowing what we know now would cost over US$10 billion (Amor, Robles, González-Barahona, & Peña, 2007) – which vastly underestimates the cost of all the code which has been improved and rewritten since 1998.

No one knows how much open source software contributes to world GDP. We know it powers around 60% of the internet (Netcraft, 2008), forms the base operating system for most consumer electronics goods containing a processor (e.g.; set top boxes, satellite decoders, DVD players, WiFi – with the big exception of mobile phones which is only around 10%) and is certainly the main software platform for BRIC (Brazil, Russia, India, China) development as shown by the recent trend towards low cost (<£200) laptops such as the Eee-PC running Linux, not Windows. We do know that not only BRIC countries see it as vital to their economic futures (The Register, 2007), so does IBM, Sun, Apple and Google who invest several billion dollars a year – even Microsoft has begun to significantly invest in open source this year with the release of their Linux & Windows management tools (see http://www.microsoft.com/opensource/).

[2] For reference, GE's cash balance was around US$30 billion in 2006 (General Electric, 2006) and Microsoft's was around US$40 billion (Microsoft, 2006).

Microsoft's about face has probably resulted from the failure of its latest Windows Vista operating system which was reputed to have cost more than US$10 billion to develop (The Inquirer, 2006) and yet by anyone's standards, it doesn't work very well[3]. Apple completely rewrote the famous Apple Mac OS X, in the process saving the company, in just two years by copy & pasting open source code and for tens of millions of dollars[4]. Those sorts of cost savings made possible by cooperative effort are surely looking attractive to Microsoft right now.

Social Capital

Proprietary software has its main base around Silicon Valley in California – and the Universities such as Berkeley and Stanford which generate its engineering talent. This poses a significant disadvantage for non-Americans, or even those not from around California in general as building up social capital is much harder. Equally however, this provides a significant barrier to entry which protects those workers from that area and gives them significant competitive advantage against anyone outside the geographical region.

Open source software has no specific base nor headquarters, though its centre of power and organisation definitely lies distributed mostly around the northern countries of Europe and to some extent the eastern seaboard of the US[5]. Its main coordination mechanism is email and bugtrackers (databases of problems with software) with contributors submitting "patches" (a set of changes) to the current main distribution. Each patch is reviewed by the group, the idea discussed at length, then an elected set of core stewards decides what the group's mind is and accepts or rejects the patch. Contributors live literally in every part of the planet and almost never meet one another and thus have absolutely no idea what the other looks like.

This is a tremendous difference. Proprietary social capital is much more traditional with much greater emphasis on financial management – there is a clear hierarchy of powerful companies, each with powerful commanders in chief and as is usual in most industries, a significant proportion of top brass get head hunted between the top companies and well paid lobbyists pound the corridors of Washington D.C. Open source social capital is vastly more diffuse, with each project having its own leaders many of whom are fundamentally ideologically at odds with one another. For example, the inventor of Linux, Linus Torvalds, has rejected an improved version of the copyright licence written by Richard Stallman which applies to most open source software – despite that this obviates much of the point of having an improved version in the first place.

Nevertheless, one should not underestimate the social capital of open source software. The recent defeat of EU attempts to introduce software patents was entirely due to an unprecedented grass-roots lobbying effort which I think surprised every MEP with its ferocity (I should know, I extensively lobbied the Irish MEP's and know well the fully funded (through donations) open source software lobbying group in Brussels, most of whom are Irish).

Cultural Capital

The institutionalised cultural capital of software development is strongly represented by academia through the award of computer science, electrical engineering and software engineering degrees – these are generally held by most programmers, with 75% of open source developers holding a university degree (David, Waterman, & Arora,

[3] Indeed, in January 2008 Infoworld rated Vista as the second worst IT flop in history (Infoworld, 2008).

[4] No one but Apple insiders know for sure, but it can't have been expensive. Apple was oscillating between small annual profits and large annual losses of up to US$1bn between 1995 and 2005 and they also open sourced the Apple kernel and much of the core operating system right from the beginning. Apple have a history of proprietary development even stronger than Microsoft (especially ever since Microsoft stole Apple's software design back in the 80's) which most would describe as extremely paranoid. They only open sourced because they literally couldn't afford the in-house developers.

[5] There are no papers or even web sites saying this. I know though as I am familiar with the leaders of all the major open source projects and I know where they live as I have been part of the open source movement since the early 90's. However, I can tell you that 53% of open source developers come from Western Europe, 27% United States, 8% Russia and Eastern Europe, 5% in East Asia, 3% in Australia and New Zealand, 3% in South America and 1% rest of the World (David, Waterman, & Arora, 2003).

2003). Proprietary institutionalised awards also include Microsoft's Certified Professional title which demand a very exacting and expensive training process (some 6% of open source developers hold one versus 5% of all developers).

Objectified cultural capital is somewhat nebulous in software – software is not something a person can hold in their hands, yet its effects on everyday experience in the world are tremendous. Every time someone uses a phone, iPod, computer or views any kind of media, the objectified effects of software are highly visible. Nevertheless, few make the relation that much of what typifies the current world is due to software – people think that special effects in film and television are the result of its artistic crew, not realising that they were simply not possible only a few decades ago – and furthermore, most special effects nowadays require writing computer programs i.e.; the artists are actually trained computer programmers. This is much more obvious in animated movies such as Shrek or Finding Nemo where a full Physics model animating the characters must be programmed into a computer.

The embodied cultural capital is even more difficult to pin down. No one is entirely sure why most computer programmers are of European cultural descent – partially, no doubt, it is because computer programming languages are exclusively written and perform in English. However, beyond that, within Europe itself most programmers come from Northern Europe and it would appear that there is something in the Saxon-derived European cultures which particularly aids the mentality required. As an example of the contrary, India despite being the recipient of much of the IT outsourcing from the West still has not developed a world-leading software development culture – neither has China, and indeed neither has Japan. These countries are very good at implementing *known* methods cheaply, but very poor at innovating new ones. While the Chinese and Indian software industries claim that they will soon catch up and exceed innovation in the West, the history of Japanese attempts in this area does not bode well. Quite simply, something in the Asian culture makes for programmers who are good at copying but not so good at leading[6]. This phenomenon extends to South America – it has even been claimed by some that a Protestant cultural element is required for a country to excel in software innovation (Mikkonen, Vadén, & Vainio, 2007) and I do personally find it interesting that consumer owned Cooperatives are also the strongest in Northern European cultures (International Cooperative Alliance , 2007).

Symbolic Capital

This is one area in which open source does strongly lead against proprietary – most workers for proprietary software organisations do after all look favourably upon open source. Unlike proprietary which is viewed with distrust, suspicion and even outright hatred, open source is symbolically highly valued within the software world and indeed by many far outside it, some of whom have advocated its mechanism as the future for most if not all major human projects e.g.; http://www.oekonux.de/.

As a result, those who are esteemed within the field are held very highly indeed due to their perceived contributions. Richard Stallman, who originally proposed the idea of a cooperatively developed operating system, and Linus Torvalds who first implemented Linux (and still acts as gatekeeper to its progress today) are both well known. Neither has actually contributed much code greatly in use[7], but they are associated with two of the "big" pillars which define the movement and thus they have a "political" symbolic capital attached to them. Contrastingly, the developers of the most useful code in the world have a more simple "respect" symbolic capital e.g.; Jean-loup Gailly and Mark Adler developed and still maintain a piece of code called zlib which is a simple compression & decompression library. This piece of open source software is so ubiquitous in practically every major piece of software that when a security vulnerability is discovered in it, almost every computer system on the planet becomes

[6] No one ever talks about this in public despite the overwhelming evidence that it is true. It seems to deeply upset a lot of people and certainly generates hacker attacks against you from India and China. I'm not saying that a very small minority over there aren't just as capable as Westerners – I am saying that they don't have the critical mass like we have here.

[7] Linus may have contributed about half the core of the Linux kernel. The *entire* kernel is about 6% of a typical Linux installation. Roughly speaking, Linus' code may make up less than 1% of a typical Linux installation but this is hundreds of times more than Stallman's code contribution.

instantly hackable. Gailly and Adler, like most behind really useful bits of software, are staunchly of BSD politics and do not discriminate between proprietary or open source usage of their code.

One of the main drivers of cooperative development – why all those thousands of man years were volunteered – is almost certainly to have one's ego stroked. As Linus Torvalds said about Linux in 1998:

> *"Originally Linux was just something I had done and making it available was mostly a "look at what I've done — isn't this neat?" kind of thing. Hoping it would be useful to somebody, but certainly there is some element of "showing off" in there too."*

This almost certainly links into psychological characteristics of typical males in Saxon based countries – the desire to show off "cool stuff", and be recognised for doing so.

Conclusion

I have reviewed both of the proprietary and open source ideologies within computer software development as according to Bourdieu's cultural capital framework. We can conclude that neither could exist as we know them without the other – each gives forth to the other, through proprietary's history of ripping off open source code and making profits from it not shared with the authors, to open source's reaction to that by vilifying and setting up in opposition to proprietary. The open source movement has crushed the proprietary efforts of most vendors, including IBM itself[8] and may even have just crushed Microsoft's who since the failure of Windows Vista have begun contributing to open source themselves. The dynamics of software development, after this analysis, appear most clear.

[8] Initially, as Linux was an implementation of Unix, it savaged the sales of proprietary Unix implementations as software could be ported relatively easily to run on Linux. Many proprietary Unix vendors went bust during the 1990's as a result, finally culminating in IBM itself giving up on trying to compete against both Linux and Windows. This marked the end for proprietary Unix implementations – there are only Linux and BSD left after Sun released its proprietary Unix, Solaris, to open source.

Bibliography

Amor, J. J., Robles, G., González-Barahona, J. M., & Peña, J. F.-S. (2007). *'Measuring Etch: the size of Debian 4.0'.* Retrieved from https://penta.debconf.org/~joerg/attachments/33-measuring_etch_slides.pdf

Bourdieu, P. (1977). *Outline of a Theory of Practice.* Cambridge University Press.

Bourdieu, P. (1993). *The Field of Cultural Production.* Polity Press.

Brooks, F. P. (1975). *The Mythical Man-Month: Essays on Software Engineering.* Addison-Wesley.

Caves, R. E. (2000). *Creative Industries: Contracts between Art and Commerce.* Harvard University Press.

David, P. A., Waterman, A., & Arora, S. (2003). *'The Free/Libre/Open Source Software Survey for 2003'*. Retrieved from http://www.stanford.edu/group/floss-us/

Department for Culture, Media and Sport. (2001). *'Creative industries mapping document'.*

General Electric. (2006). *Annual General Report.*

IDC. (2007, October). *'The Economic Impact of IT, Software, and the Microsoft Ecosystem on the Global Economy'.* Retrieved from http://www.microsoft.com/downloads/details.aspx?FamilyId=BB95083E-2BCA-4C60-832C-9B35A2A6BC6D&displaylang=en

Infoworld. (2008). *'Tech's all-time top 25 flops'*. Retrieved from http://www.infoworld.com/archives/emailPrint.jsp?R=printThis&A=/article/08/01/21/03FE-25-tech-failures_1.html

International Cooperative Alliance . (2007). *Annual Report.*

Jorgenson, D. W., & Vu, K. (2005). 'Information Technology and the World Economy'. *Scandinavian Journal of Economics , vol. 107* (no. 4), pp. 631-650.

Microsoft. (2006). *Annual General Report.*

Mikkonen, T., Vadén, T., & Vainio, N. (2007). *'The Protestant ethic strikes back: Open source developers and the ethic of Capitalism'.* Retrieved from http://www.firstmonday.org/issues/issue12_2/mikkonen/index.html

Netcraft. (2008). Retrieved from http://news.netcraft.com/archives/2008/04/14/april_2008_web_server_survey.html

The Inquirer. (2006). *'Vista is the last of the dinosaurs'*. Retrieved from http://www.theinquirer.net/en/inquirer/news/2006/12/04/vista-is-the-last-of-the-dinosaurs

The Register. (2007). *'Sun looks to huddled masses for growth'*. Retrieved from http://www.theregister.co.uk/2007/08/30/sun_bric_growth/

The Register. (1998). *'The true costs of Linux development'*. Retrieved from http://www.theregister.co.uk/1998/10/21/the_true_costs_of_linux/

Zeitlyn, D. (2003). 'Gift economies in the development of open source software: anthropological reflections'. *Research Policy , vol. 32* (no. 7), pp. 1287-1291.