# Observed effects of free software on software development and requirements management

David Callele[1] and Krzysztof Wnuk[2]

[1] Experience First Design Inc.
Saskatoon, Canada
dcallele@experiencefirstdesign.com
[2] Software Engineering Lund University,
Sweden
Krzysztof.Wnuk@cs.lth.se

**Abstract.** [Context & motivation] Free software is often declared to be a positive movement that makes technology accessible to those who might not otherwise have access.

[Problem] While the positive effects, to one degree or another, have often been discussed there has been relatively little discussion of the possibly negative effects of the free software movements. In general, these approaches have led to ever-increasing concentration of economic power in a smaller number of entities, the reduction of margins to the point where there is little economic incentive to investment and a general casino-like approach to software development: build it, then build a customer base and then try to find a way to monetize the customer base rather than the product.

[Contribution] The resulting conditioning of the customer base has led to a significant reduction in the availability of venture capital for software products and a corresponding increase in the availability of venture capital for software as a means to make the customer the product through data analytic approaches. This short paper discusses these observed effects of free software on the software economy and possible effects on requirements engineering practice.

**Keywords:** free software, requirements management, negative effects, software product management

## 1 Free software

The early experiments at Berkeley and MIT [4] in the 1960s and the first public release of the Linux kernel source in 1991 [1], free[1] and open source software have deeply penetrated and shaped the software industry [4]. The scale of these sociological movements has far exceeded the founding expectations. Today, many large proprietary software products have a free and open source counterpart, some of which

[1] This work uses free in the context of "without fee or consideration". This work does not address free in the context of "free as in freedom" as is often used by organizations like the Free Software Foundation.

have been shown to deliver equal or greater quality customer experience than the original[2].

The Internet has reshaped many industries by enabling direct-to-consumer communication and distribution. We posit that the continued practice of making software and software-based services available for free could have Internet-like effects upon the practice of requirements engineering, the software industry as a whole and upon the consumers of the software themselves.

In this paper, we present and discuss our observations of the effects of free software on investments in software development and the potential ramifications for requirements engineering.

## 2 Related Work

For the purposes of this section, we shall assume that the participants being discussed are rational in the economic and financial sense. In other words, some form of cost-benefit analysis is performed and the necessary investment in developing software delivers a return that is acceptable to the investor. The Return On Investment (ROI) is not necessarily monetary; for example, making a donation to an open source effort is not necessarily economically rational, at least from the perspective of some observers. But, to the individual making the donation of their time and effort there exists some form of return on that investment that makes the investment worthwhile to them.

Lerner and Tirole investigated the open source movement from an economist's perspective [3, 4] and found many aspects that were "economically puzzling". They were able to identify that contributor ROI is dominated by career-based motivations and ego gratification, observing that open source projects may be more 'cool' to contributors than their routine development tasks. On the other hand, Scacchi [5] focused on understanding how the development of requirements for open source software differs from the development of requirements for commercial software, stressing that open source software initiatives do not adopt modern software and requirements engineering practices with the absence of formal requirements elicitation, analysis and specification activities. Lerner and Tirole [3] confirmed these observations, noting that expected project and product management practices were less stringent in non-commercial projects.

We are unable to identify further requirements engineering specific research literature that studies the impacts of free and open source software on requirements management and software product management.

## 3 Cost of Production and Reproduction

Determining the ROI for free software requires an understanding of the cost of production and reproduction of software. We generally model the reproduction cost of software at zero since the incremental energy cost of transmitting a copy is very small. Typical production cost calculations are dominated by the developer time invested in the project. However, these are not the only costs that should be considered

and we look more closely at two factors that are not often taken into consideration by software developers[2].

Lost Opportunity Cost (LOC) is often ignored in cost analyses for software development. In the context of a single developer, what else could that developer have been doing with the time that they spent working on the software? Let us assume that the developer is employed by a third-party that develops commercial software (software that must be paid for by the user). Outside of the normal work week, the developer could choose to develop software that will be freely distributed when completed. Or, the developer could choose to work extra hours for their employer and the employer's product could enter the market more quickly. Which effort should the developer support? Developer time is typically a zero-sum game and the decision to work on one project means another project does not get those resources.

Lost opportunity cost models for open-source projects can be even more complex. Open source projects that are commercially focused must resolve issues of shared intellectual property, especially when the open source license requires the sharing of all modifications. Deriving a custom branch from the main open source branch to deliver significant proprietary innovations with substantial customer value can lead to unexpected maintenance and bug fixing efforts – responsibility for keeping the now proprietary codebase current with the open source code base is no longer a shared effort. Only a proper analysis can determine whether the adoption of the open source code base is financially justified for a given project.

Some models also consider LOC at the society level. For example, the developer could have been working on software for their employer rather than on free software. Given that the employer is a for-profit entity, this activity is expected to lead to greater economic activity and a probable increase in the corresponding tax base. Does the same effort devoted to free software lead to the corresponding increase in the tax base? Perhaps it leads to a reduction in costs, thereby increasing margins and delivering the same net economic benefits? Does the society-level analysis outweigh the personal analysis?

Life Cycle Cost Analysis is another analysis that is often ignored in software cost analyses. Even if the reproduction cost for a software product is zero, the replacement costs for installing a new version of the same product can be very high – particularly when data must be migrated and users must be retrained.

These few examples illustrate that free software is not free, there are non-trivial investments required to conceive, design and implement the software. These resources must come from somewhere and these resources are, of necessity, not applied to alternative projects. Determining the appropriate model for calculating the cost base is subject to interpretation and the resulting values can vary widely.

[2] The work of economists Lerner and Tirole [3,4] does investigate some aspects of this issue.

## 4 Dumping, the WTO, and Free Software

The cost of software production is important in the context of international trade, particularly as economies attempt to diversify into the so-called knowledge economy. International trade is generally governed by the rules of the World Trade Organization (WTO). In the realm of tangible goods, those goods that consume incremental materials in the production or reproduction of that good, international trade is governed in part by dumping rules. Simplistically, dumping is considered a form of *predatory pricing* that occurs when a supplier in one country charges consumers in another country a price that is either below the price charged in their home country or *below the cost of production*. Accusations of predatory pricing have been made against dominant software industry players such as Microsoft[3] and Google[4]. While regulatory powers such as antitrust have been brought to bear upon specific instances, we are unaware of any large-scale consideration being given to whether dumping constraints should also apply to intangible goods such as software. In other words, is there a basis for making the argument that the practice of giving away software for free constitutes dumping?

We only need to look at the mobile telephone industry to see an example of the possible commercial effect of a free and open-source solution, a commercial effect that dumping regulations are intended to preclude in the realm of tangible goods. The introduction of the iPhone transformed the consumer mobile device marketplace, exerting great competitive pressure on traditional handset manufacturers. It also exerted great pressure upon Google because Apple maintained control over consumer data generated by these devices. In response to this threat to its data acquisition business, Google delivered the Android operating system as a free alternative to Apples proprietary operating system. Google's decision to protect their data acquisition market through the release of a free handset operating system eliminated the significant barrier to market entry associated with handset operating system development and rendered the massive operating system investments by handset manufacturers such as Nokia and Sony-Ericsson essentially valueless. Now anyone could become a handset provider – all they had to do was build the hardware, the operating system came for free.

For software to be subject to dumping rule analysis, appropriate pricing models would be required. As we saw in the prior section, simple models (*e.g.* reproduction cost is zero) are unrealistic but determining the appropriate level of complexity could require significant negotiation (*e.g.* an analysis that looked only at the cost of labor would place all developed nations at a significant disadvantage in the sector).

If free software was subject to dumping rules, how would this affect the practice of requirements engineering? Would any additional factors need to be considered or would RE practitioners argue that this was outside of the scope of their responsibility? Would this not be a regulatory compliance requirement just like safety or taxation?

---

[3] http://www.theguardian.com/technology/blog/2006/jun/21/microsoftaccus

[4] https://fsfe.org/activities/policy/eu/20130729.EC.Fairsearch.letter.en.html

## 5 Observations on the Effects of Free

The various app marketplaces for mobile devices contain millions of apps. As the number of available apps has increased, getting the consumers attention has become increasingly difficult. Many developers now give away their apps for free in the hope that they can obtain users, trusting that they will be able to derive revenue from the users once they become tied to using their apps. However, consumers have rapidly become acclimated to the concept of apps being free and there is now significant reluctance to pay for apps when they might soon become available for free – either from the original developer or from someone cloning the concept.

The ability for a second or third party to simply clone the functionality of a software application without consequence is a significant competitive market risk for the innovating developer. Investing in new product development usually requires significant resources and if someone else can just copy or clone the innovation then why invest in that development in the first place? This is a very real threat to ongoing innovation in our industry and has led to a significant reduction in the placement of investment funds in the sector except when the project can demonstrate a significant barrier to competitive market.

Significant investment is currently being placed in projects where the software facilitates the collection of unique data about the users of the software. This unique data has value to third parties such as marketers and entrepreneurs are monetizing this data to ensure their revenue stream in the presence of customer expectations for free software. Essentially, the developer becomes a developer not of a software product that has value in its own right, value that the customer is willing to pay for, but a developer of software that converts data about the user of the software product into the item of value.

In the Angel investment group of which the first author is a member, in the last three years less than 6% of all investment pitches in this sector received investment. In every case where an investment was placed, the investment hinged upon the extent to which the users were monitored and data about them gathered. Anecdotally, this result is typical across North America.

The availability of free software has, therefore, had at least two impacts on the practice of software development. It has led to a chilling effect on investment unless there is some way to reduce the threat of a competitive clone of the product. And, in response, the industry has shifted to monetizing user data instead of monetizing the product itself.

As a result, it may be necessary that the practice of RE expand its scope. Traditional RE that focusses on features is still mandatory for without the right set of features there will be no users. However, RE practices may now need to include RE for user data acquisition to support monetization efforts or there will be little or no revenue to sustain operations. Further, we may even see RE efforts that focus on mechanisms for convincing the user to give up their privacy in exchange for the service.

Requirements engineering for data acquisition can be very technically and legally complex. Effective and minimally intrusive data acquisition mechanisms require significant design of experiment and data science expertise and RE efforts may require

the addition of subject matter experts in these fields. Ensuring compliance with privacy legislation in all operating jurisdictions for a product or service is another significant effort. Work on privacy and RE is in its nascent stages and interested readers should investigate the RELAW series of workshops for further information. While there have been mentions of privacy regulation compliance in published work, we are unaware of any work that focuses on how the general practice of RE may need to change to accommodate these needs.

## 6 Conclusions

Despite widespread adoption of free and open source software, studies that explore the negative effect of this phenomenon on the software business economy are rarely reported in the technical literature. In this paper, we present a number of observations on the consequences of free software. We identify that software is generally exempted from international trade restrictions on predatory pricing (although some antitrust and monopoly actions have been taken against the largest industry members). The advent of a marketplace that expects software to be free has had a chilling effect on investment unless there is a substantial barrier to competitive market entry. Data collection about the users of the software has become this barrier to competitive market entry, turning the users into the revenue stream rather than the product.

We are unaware of any easy answers to the observed dilemmas. Perhaps we could mandate that customers have the right to opt out of this data collection by paying a fee. What would the value of an organization like Facebook or Google be if they had direct paying customers? What would happen if we applied dumping guidelines to software pricing? Would this approach result in a more equitable and privacy-preserving marketplace or would this have a significant negative effect on innovation?

In future work, we would like to further study these issues by empirically evaluating the assumptions, hypotheses and personal experiences brought forward in this paper. Once these observations are better grounded we can return our attention to their effects on the practice of RE and potentially develop new practitioner guidance.

**References**


1. The description of the Linux kernel project is available at http://en.wikipedia.org/wiki/Linux_kernel
2. Stamelos, I., Angelis, L., Oikonomou, A., Georgios A., Bleris, L.: Code quality analysis in open source software development. Inf. Sys. Journal, 34, 43-60 (2002)
3. Lerner, J., Tirole, J.: Some Simple Economics of Open Source. Journal of Ind. Econ. L , 197-232 (2002).
4. Lerner, J., Tirole, J.: The Economics of Technology Sharing: Open Source and Beyond. Journal of Economic Perspectives, 19, 99–120, (2005)
5. Scacchi, W.: Understanding the requirements for developing open source system. IEE Proc.-Soft., 149, (2002).