Software Development, Intellectual Property Rights, and IT Security[*]

Dipl.–Inform. Robert A. Gehring[†]
rag@cs.tu-berlin.de

Technical University of Berlin,
Computers and Society Research Group

Berlin, 2 May 2002

Abstract

Within the last 60 years the process of both software development and deployment has undergone significant structural changes. I present here some of the cornerstones of the way software was freed from its hardware roots and how this had deep impact on the character of the software development process and its inherent values. Instead of focusing on technology issues outside of their context, I attribute many of the security problems we experience with today's software to the described process, the institutional change associated with it, and its unintended results.

[*]Position paper, accepted for the First Workshop on Economics and Information Security, University of California, Berkeley, May 16–17, 2002. Available online: `http://ig.cs.tu-berlin.de/ap/rg/`.
[†]The author is assistant lecturer at the Technical University of Berlin.

1 The Commodification of Software

1.1 The different roots of modern computing

Electro–mechanical computers were first developed in the 1940s for military purposes. The development of the software for these early computers took place under severe control by the state. The goal was mainly to undermine the enemy's security, to know what he knows and thereby to be able to preserve one's own position of power. Quality in terms of usability was paramount to reach the goal.

A little later, a scientific community began to form and started independently the development of universal computing devices, and the appropriate software. It started with John von Neumann's approach to represent the content of calculation independently from the (technical) means of calculation. The interest was directed to making calculations of any kind—as required by nearly all fields of natural sciences and disciplines of engineering—more affordable, i.e. faster, cheaper, and more reliable[1]. The underlying assumption was to hasten the scientific and technical progress, not—at least not in the first place—to create a product in order to derive a profit from its sale.

A few years later, the scientific community splitted. Some of the early computer developers decided to develop and build computers on their own (proprietary hardware together with the necessary proprietary software) and to provide them as goods on the market for calculation and controlling devices. Increasingly, a philosophy of rationalization was adopted in more and more fields of industry and formed a demand basis for such efforts.

At the same time, the state demanded stronger calculation power in order to keep control of the population development. The universality of the new computing machines offered more and better opportunities than the old single–purpose tabular machines.

1.2 A market for software evolves

In the beginning, software–development was driven by the interests of the users, i.e. large companies and the state. They provided the specifications how the machines should behave. Software *as such* was no business but rather came bundled as a part of the calculation machine. This was the logical consequence of the fact that there existed only a handful of computers for more or less specific purposes. The computer manufacturers delivered a *functional entity* to the customers. Quality was more or less a matter of functionality and reliability for a special purpose.

The situation changed in the late 1960s and early 1970s, when independent software vendors established that developed and distributed packaged software for already deployed hardware. This was fostered by the marked domination of IBM and their de facto hardware standard. IBM computers were deployed in completely different environments. The focus of application development shifted from a customer–oriented view of software functionality towards a supplier–oriented model of salability. Independent software vendors derived

[1]As known, complex mathematic calculations made by man are often subject to mistakes. For example, in earlier times logarithm tables contained lots of errors that could only be avoided with the help of computing machines.

their business plans from the anticipated generic demands of a relatively large user base. Software became a product within the now growing software market.

1.3 The emerging property rights regime

But there was a precondition that had to be fulfilled before software could be dealt as a commodity. Knowledge products—or information products in general—cannot simply be sold because they are easily copied and distributed 'for nothing'.[2]

Software as an information product is what is called a public good (OSTROM/HESS 2001: 54). That means the resource—software in our case—, after its creation, is not scarce by nature (no subtractability) and there is no inherent possibility to exclude others from the use of a copy (practically no excludability).

Confronted with public goods, economists usually recommend to create property rights in order to make a public good scarce and thereby creating the opportunity for the creator to derive monopoly profits from his work (for a limited time). Trading such works in the market place should put them to their highest valued use.[3]

The idea behind the property rights (PRs) approach is to give incentives for productive activities. Accordingly, in the realm of intellectual activities there exist intellectual property rights (IPRs). They were introduced *"to promote the progress of science and useful arts"* as it is stated in the Constitution of the United States. Monetary rewards are the most important instrument of the neoclassical economic approach to promote productive activities.

That instrument, too, is applied to creative work such as software development. It takes the form of the copyright, i.e. a state granted monopoly to market one's work thus enabling to derive monopoly profits. And more and more software is additionally covered by patent protection and other special laws[4].

In the early days of software, when it came bundled with the hardware, it wasn't thought of as something to be legally protected through copyright. It was not earlier than in the late 1970s, that software protection by copyright was broadly accepted.[5] After the enactment of the Computer Software Copyright Act of 1980 it was clear that there are exclusive property rights in software. Software in itself—without connection to a piece of hardware—was now clearly qualified as a matter of possible business activities.[6] Other countries followed closely.

[2]This is one of the commercial problems, the information based industry faces today on the Internet.

[3]According to the standard justifi cation of property rights in order to explain the market economy's effi ciency.

[4]See for example the Digital Millennium Copyright Act (DMCA) in the U.S. In Europe, the directive 2001/29/EC contains the instructions for national legislators to craft laws similar to the DMCA. The DMCA is avaliable online: `http://www.loc.gov/copyright/legislation/dmca.pdf` [02 May 2002]. See `http://directory.google.com/Top/Society/Issues/Intellectual_Property/Copyrights/European_Union_Copyright_Directive/` [02 May 2002] for a number of materials relating to the directive.

[5]According to BENDER (1978: 4–4), *"there appear to be no published decisions on software copyright prior to 1978"* [in the U.S.].

[6]Though, the assignment of copyright protection to software brought along a problem. Copyright contains a 'fi rst sale doctrine' that software vendors liked to avoid. Their solution was licensing software instead of 'selling' it. Cf. LUNNEY (2001: 827), supra note 38.

1.4 The dichotomy of interests

Software became a commodity and this is the point were the classical economic rationality of a firm (producer) that sells a product for profit comes into play. The motivation of a firm to produce software is completely different from the motivation of the before mentioned actors state and scientific community. For the last–named, software is only a means to an end (for a non–profit purpose), whether it be power and/or control or scientific progress. A firm has a different goal: to profit from the sale of goods.

Since all human behavior is motivation–triggered, there must clearly be a different approach behind the way a firm deals with software and the ways other actors do (who don't intend to derive profit from the software itself).

For example, one classical differentiation of the value a certain thing has to people is that between *sales value* and *use value*. *Sales value* gets realized by the seller after sale, while the *use value* depends on the application of a technology. This is one (if not *the*) critical point when it comes to reliability and security considerations: Reliability and security are qualities of use, but not necessary for sale — as long as there exist no binding liability rules as it is the case with packaged software (Cf. KANER 1997b). There evidently exists a conflict of interests that requires resolution in one or another way.[7]

A seller does not necessarily need to be interested in the usability of his goods as long as there is no controlling feedback (i.e. *effective* influence) from the users. There is no such thing as a built–in ethics in the commercial process of developing software and selling it, especially not under conditions of competition.[8] If we have to take into account possible network externalities the competition intensifies. Being first on a lock–in market with strong network externalities can then imply to become the one who controls the market. (SHAPIRO/VARIAN 1999: 167, 173–192) Since quality assurance measures are time–consuming, they are diminished and avoided if possible, for example, due to lack of applicable liability laws.

1.5 Insecurity a market failure?

The different interests of software producers and software users require some process of mediation and effective feedback. Consequently, IT–security, as judged from the perspective of the users, requires a *process, not a product* — as stated by SCHNEIER (2000: 367 ff.).

It is not easy to imagine how big a role software firms could play in this process and what the regulative tasks for the state are, if at all. Obviously, the market can provide a huge amount of highly sophisticated, full featured software. But the experiences with unreliable and insecure software we've made over the last decades raise doubts if the market can

[7]Among the legal system all over the world, a common approach to implement such a controlling feedback is to let courts apply rules of liability. Unfortunately, this approach has failed in the fi eld of software, not least because of the qualifi cation as some kind of literature software experiences by the copyright laws.

[8]This insight is all but a new one. For example, famous pioneer in automobile industry, HENRY FORD, noticed the contradiction between profi t interests and quality demands. He came to the conclusion that it would be better to transform the profi t–based economy into an performance–based economy. His idea did not make him many friends. One interesting point that FORD made was that economy existed to serve *community needs* rather than *selfish interests*. Cf. HANSEN (1992: 124 f.).

provide reliable and secure software—regularly, not only exceptionally. It looks like a market failure.

Issues of asymmetric information concerning product quality between vendors and customers make the problem worse: How to know if a software product will do the job *before* purchase? The customer is always handicapped.[9]

1.6 A solution from outside the market?

But, as distinct from other fields of technology, there exists another source of supply. Today, software is not only provided via market/price mechanisms but also through the so called open source software development process.[10] Interestingly, this process is driven primarily not by economic rationality (i.e. seeking individual advantage) but rather by motives we have yet to understand.[11]

However, there is something strange with this open source process at least when one takes into account what has been said above about the importance of property rights and what the theory of economic rationality suggests. Within the open source process, developers expressively forego to claim exclusive property rights on the fruits of their labor.[12]

Surprisingly, instead of running into a *"tragedy of the commons"* (HARDIN 1968), it is flourishing. This kind of software development process shows an amazing degree of sensitivity to security and reliability issues.[13] One could suspect that there is a connection between the motivation of software developers (mainly no monetary interests), the feedback from users (demand for use value) to producers[14] and the outcome with regard to reliability and security. But the validity of that suspect has yet to be scrutinized.

2 The task for research

Changing institutions have lead to changes in the incentive structure (Cf. NORTH 1999) with patently suboptimal results. Intellectual property is ***property without liability***, it seems. (IT–)security without liability won't work since the costs of security incidents are imposed on the public instead of their creators.

[9]It is to mention, that the shareware distribution method fits the customers needs much better. However, only small firms and freelance programmers do support shareware. The large software companies prefer up–front charges and shrink-wrap licenses with liability limitations.

[10]The Open Source model for software development evolved from the free software movement initiated by RICHARD STALLMAN. STALLMAN decided to go a different way of software development when increasingly confronted with intellectual property claims that hindered him to change some piece of software. Cf. STALLMAN (1999).

[11]There is little knowledge based on empirical data about the motives of open source developers. Last year, a study conducted at the TU Berlin gathered a large data set which gives some clue to the issue. Until now, it is not fully analyzed. Cf. ROBLES et. al (2001). Though, there is an amount of literature dealing with the more general issue of motivation in the context of private supply of public goods. Cf. LUNNEY (2001: 860 ff.) for a discussion.

[12]Scrutinizing this issue, one may come to the conclusion GUPTA (1997) draws: *"It is, therefore, well accepted after two decades of intense analyses that the solution to this logical quagmire lies, not with the rationality of the actor taking part in collective action, but with the definition of rationality."* quoted with LUNNEY (2001: 860), supra note 158.

[13]See for example KOERNER (2000).

[14]The informal organization of the developers and the use of e–mail lower the transaction costs for communication and thus, presumably, further the establishment of an value–based community. See, for example, LUNNEY (2001: 858) following HEGEL and discussing an *honor system* as a viable framework for private supply of public goods.

An examination of the existing literature has shown that this relation of intellectual property based economic motivation to security and liability issues has not been systematically investigated yet, although some relevant contributions[15] already exist.

Following the investigations outlined above of the historical background of the software development process, the legal developments accompanying it and the identified different approaches to software development it is of interest to identify the economic, technical, legal and sociological forces that drive the different approaches of software development. The "Law and Economics" theory point of view as developed by RICHARD POSNER thirty years ago (POSNER 1973) appears to be a promising starting point for the analysis.

According to Posner, *"legal rules, including the rules made by judges, are to be judged by their consequences"* (FRIEDMAN 1998: 55). That would call for some kind of *assessment of the law* with regards to security aspects.

Among others, the following issues concerning intellectual property rights[16] need to be discussed:[17]

- Intellectual property rights (copyright, trade secret, trademark and patent protection) essentially have not been judged by their consequences for centuries. As the ruling incentives for economic behavior they seem to backfire in the case of software protection. The incentives are set in a way to promote the race for market share at the expense of quality and security considerations. At the same time, no effective liability laws exist to compensate.

- Intellectual property rights give exclusive access to information while public security (IT–security included) requires open access to at least some information.[18] Existing approaches to overcome this dilemma have proven inappropriate, but legislators widely ignore the issue.

- Intellectual property rights tend to become more and more fragmented. That is due to the easy available opportunities for appropriation of knowledge copyright and patent laws offer. And as we have learned from the investigation of anticommons phenomena (cf. HELLER 1998; SCHULTZ et al. 2002), fragmented property rights may well lead to underutilization of resources. The conclusion goes that fragmented property rights in security–relevant knowledge will probably lead to underutilization of that knowledge.[19]

[15]See, for example, ANDERSON (2001) and KANER (1996, 1997a, 1997b, 1998).

[16]"The Digital Dilemma", an expertise prepared by the U.S. NATIONAL RESEARCH COUNCIL, stated: *"Because the expansion of patent law to cover information inventions has occurred without any oversight from the legislative branch and takes patent law into uncharted territory, this phenomenon needs to be studied on a systematic basis, empirically and theoretically."* (NRC 2000, 228) That is true and there is nothing to add. Unfortunately, such requests have not been taken seriously enough.

[17]I have done some preliminary examinations. First results became part of a contribution to an expertise for the German Federal Ministry of Economics and Technology (LUTTERBECK et al. 2000). The expertise deals with economical consequences of enhanced patent protection for software and possible effects on the security and reliability of software products. Further research was presented as a paper contributed to the international conference "Innovations for an e–Society. Challenges for Technology Assessment" (GEHRING 2001). Much research work remains to be done.

[18]"Repair deemed unlawful" through the ban on reverse engineering contained in copyright law is just one example for problems induced by intellectual property. Cf. ROSENOER (1997: 22 ff.)

[19]The case of the RSA–Patent can serve as an example. Cf. MARTI (2000).

References

[1] ANDERSON, Ross (2001): **Why Information Security is Hard - An Economic Perspective**, online: `http://www.cl.cam.ac.uk/ftp/users/rja14/econ.pdf` [28 Aug 2001].

[2] BENDER, David (1988): **Computer Law. Software Protection**, New York: Matthew Bender & Co., 1988.

[3] FRIEDMAN, David D. (1998): **Posner, Richard Allen**, pp. 55–62 in: Peter NEWMAN (ed.): The New Palgrave Dictionary of Economics and The Law, Vol. 3, London: MacMillan Reference, 1998.

[4] GEHRING, Robert A. (2001): **“Software Patents” — IT–Security at Stake?** Paper prepared for the international congress “Innovations for an e–Society. Challenges for Technology Assessment”, Oct. 17–19, 2001, Berlin, Germany, online: `http://ig.cs.tu-berlin.de/ap/rg/2001-10/index.html` [02 May 2002].

[5] GUPTA, Dipak K. (1997): **Group Utility in the Micro Motivation of Collective Action: The Case of Membership in the AARP**, J. Econ. Behav. & Org. 1997, Vol. 32, pp. 302.

[6] HANSEN, Klaus P. (1992): **Die Mentalität des Erwerbs. Erfolgsphilosophien amerikanischer Unternehmer** [The Mentality of Making Profit. Philosophies of Success of American Entrepreneurs], Frankfurt/Main, New York: Campus, 1992.

[7] HARDIN, Garrett (1968): **The Tragedy of the Commons**, Science Magazine, Vol. 162, 13 Dec 1968, pp. 1243-1248.

[8] HELLER, Michael A. (1998): **The Tragedy of the Anticommons: Property in the Transition from Marx to Markets**, Harvard L. R. 1998, Vol. 111, pp. 623–688.

[9] KANER, Cem (1996): **Liability for Defective Content**, in: Software QA, 1996, Vol. 3, No. 3, p. 56, online: `http://www.badsoftware.com/badcont.htm` [28 Aug 2001].

[10] KANER, Cem (1997a): **The Impossibility of Complete Testing**, in Software QA, Volume 4, No. 4, p. 28, 1997, online: `http://www.kaner.com/articles.html` [28 Aug 2001].

[11] KANER, Cem (1997b): **Software Liability**, 1997, online: `http://www.kaner.com/articles.html` [28 Aug 2001].

[12] KANER, Cem (1998): **The Problem of Reverse Engineering**, in Software QA, Vol. 5, #5, 1998, online: `http://www.kaner.com/articles.html` [28 Aug 2001].

[13] KOERNER, Brendan I. (2000): **The World’s Most Secure Operating System.** In: The Standard, August 14, 2000, online: `http://www.thestandard.com/article/display/0,1151,17541,00.html` [18 Oct 2000].

[14] LUNNEY, Glynn S. Jr.: **The Death of Copyright: Digital Technology, Private Copying, and the Digital Millennium Copyright Act**, Virginia L. R. 2001, Vol. 87, No. 5, pp. 813–920.

[15] LUTTERBECK, Bernd; HORNS, Axel H.; GEHRING, Robert A. (2000): **Sicherheit in der Informationstechnologie und Patentschutz für Softwareprodukte – ein Widerspruch?** [Security in Information Technology and Patent Protection for Software Products: A Contradiction?, Short Expertise Commissioned by the Federal Ministry of Economics and Technology], Berlin, December 2000, online: `http://www.sicherheit-im-internet.de/download/Kurzgutachten-Software-patente.pdf` [28 Aug 2001].

[16] MARTI, Don (2000): **Focus: Security.** In: Linux Journal, 2000, No. 10, p. 91.

[17] NATIONAL RESEARCH COUNCIL (NRC) (2000): **The Digital Dilemma. Intellectual Property in the Information Age**, Washington, D.C.: National Academy Press, 2000.

[18] NORTH, Douglass C. (1999): **Where have we been and where are we going?** pp. 491–508 in BEN–NER, Avner (ed.); PUTTERMAN, Louis (ed.) (1999): Economics, Values, and Organization, Paperback Edition, Cambridge, U.K.: Cambridge University Press, 1999.

[19] OSTROM, Elinor; HESS, Charlotte (2001): **Artifacts, Facilities, And Content: Information as a Common–pool Resource.** In: Focus paper, discussion drafts for the Conference on the Public Domain, Duke Law School, Durham, NC, November 9–11, 2001, online: `http://www.law.duke.edu/pd` [14 Nov 2001]

[20] POSNER, Richard (1973): **Economic Analysis of Law**, Boston: Little, Brown, 1973.

[21] ROBLES, Gregorio; SCHEIDER, Hendrik; TRETKOWSKI, Ingo; WEBER, Niels (2001): **Who Is Doing It? A research on Libre Software developers**, Research Paper, TU Berlin, August 2001, online: `http://ig.cs.tu-berlin.de/s2001/ir2/ergebnisse/OSE-study.pdf` [24 Mar 2002].

[22] ROSENOER, Jonathan (1997): **CyberLaw. The Law of the Internet**, New York: Springer, 1997.

[23] SHAPIRO, Carl; VARIAN, Hal R. (1999): **Information Rules. A Strategic Guide to the Network Economy**, Boston, MA: Harvard Business School Press, 1999.

[24] SCHNEIER, Bruce: **Secrets and Lies. Digital security in a networked world**, New York: Wiley, 2000.

[25] SCHULTZ, Norbert; PARISI, Francesco; DEPOORTER, Ben (2002): **Fragmentation in Property: Towards a General Model**, George Mason University, Law and Economics Working Paper Series, online via SSRN: `http://papers.ssrn.com/abstract=300681` [13 Mar 2002].

[26] STALLMAN, Richard (1999): **The GNU Operating System and the Free Software Movement**, pp. 53–70, in DiBona, Chris; Ockman, Sam; Stone, Mark (eds.): Open Sources. Voices from the Open Source Revolution, Sebastopol: O'Reilly, 1999.