# 1. Introduction

The Linux Credits file has been included with all Linux operating system distributions since March 1994. It contains information on contributors to the Linux operating system kernel development. By analyzing the evolution of the Credits file, it is possible to study the evolution of the Linux developer community. For example, by using the address information contained in the Credits file, it is possible to describe the geographical expansion of the core Linux developer community. By combining this information with other data sources one can study the different factors that have influenced the expansion of the Linux developer network.

The open source development model operates over the Internet. Huge amounts of historical data are available on the Internet that can be used to study the open source phenomenon. In theory, the different versions of open source files record the evolution of open source projects in great detail. It is therefore an intriguing possibility to automatically generate data from files included in open source program distributions.

Several research projects have tried to analyze the open source phenomenon using automated analysis of source code and other data that are generated during the development process. Some authors have focused on a single open source project, using the source code snapshots, change logs, data from version control systems, problem tracking databases, and e–mail lists *(e.g.,* Koch and Schneider, 2000; Tuomi, 2002; Mockus, Fielding, and Herbsleb, 2002; Robles–Martínez, Gonzáles–Barahona, *et al.,* 2003; Robles, Koch, and Gonzáles–Barahona, 2004). Other research projects have adopted a horizontal approach, studying multiple open source projects, for example, by analyzing data available at open source Web portals, such as sourceforge.net and freshmeat.net (Krishnamurthy, 2002; Healy and Schussman, 2003; Capiluppi, Lago, and Morisio, 2003), or by retrieving e–mail addresses and copyright notices from large source code collections spanning thousands of open source projects (Ghosh, Robles, and Glott, 2002).

In practice, automated analysis is complicated by a number of factors. This paper discusses the challenges of analyzing the Linux Credits file, shows how these challenges can be addressed, and provides data that can be used to validate studies on Linux kernel development. The Linux Credits file contains information about project contributors who have made the Linux operating system what it is today. The analysis of Linux Credits, therefore, provides important information on the open source development model and its dynamics. A focused study of this strictly limited case example highlights the problems that need to be addressed by research projects that extract research data from open source files. The presented results and methodological issues will therefore be of general interest also to other studies that attempt to use automated tools to discover data from source code files.

This paper is organized as follows. The next section describes the basic characteristics of the Linux Credits file. Section 3 discusses the problems that automated source code scanning programs have to address to extract useful data from the Linux Credits files. Section 4 describes a methodology that can be used to associate persons in Linux Credits to countries. Section 5 describes the characteristics of time–series data that has been created using the methodology. Section 6 comments on a report by Alexis de Tocqueville Institution (Brown and Orndorff, 2004) that used an earlier draft version of this paper to derive policy recommendations for open source software. Section 7 discusses the results and compares them with some previously presented claims concerning the open source development model. Section 8 points out some future research opportunities that could be addressed using the described data.

## 2. The structure of Linux credits file

The first version of the Linux operating system was released on the Internet in September 1991. It consisted of 88 files, totaling 231 kilobytes of code. At the time, it would have been difficult to predict that Linux would ten years later be considered as a major threat to Microsoft's dominance in operating systems. The amount of code in the first Linux release was rather modest. The smallest file consisted of a single line and the longest file was 678 lines, or 612 lines without comments. The median size file included in the first Linux package was 37 lines without comments. The program was written in the C programming language, which the creator of Linux, Linus Torvalds, had started to study in 1990.

The Linux kernel distribution consists of files that are needed to compile the Linux operating system. As the system has become increasingly complex, a number of documentation files have been added to the distribution package. The full operational GNU/Linux distribution includes a large number of programs in addition to the operating system kernel, and firms such as SuSE and RedHat have built businesses that package, document and support complete Linux based environments. The Linux kernel is the core of these distributions. The Linux kernel packages are also freely available through Internet sites that store all the existing historical versions of the Linux operating system code.

The kernel packages have included a Credits file in their Linux subdirectory since 13 March 1994. This file acknowledges important contributions to the kernel development. During the evolution of Linux, new contributors have been continuously added to the file. In July 2002, the Credits file contained information on 418 developers. With two exceptions, all were male [1].

The Credits file is a simple text file where the included persons can document information about their main areas of contribution, their e–mail and physical addresses, and other relevant data. A typical example of an entry in the Credits file is shown below in Table 1.

**Table 1: A typical entry in the Linux Credits file.**

| |
|---|
| N: Linus Torvalds |
| E: Linus.Torvalds@Helsinki.FI |
| D: General kernel hacker |

| |
|---|
| S: Kalevankatu 55 B 37 |
| S: 00180 Helsinki |
| S: Finland |

In the first Credits files the possible entry fields had line descriptors for name (N), e–mail (E), description (D) and snail–mail address (S). In the more recent files, fields for Web address (W) and PGP key ID and fingerprint (P) have been added. The people included in the Credits file provide this information themselves. Although in principle it is possible for anyone to add herself to the Credits file, the persons who manage the kernel releases only accept entries from persons who are perceived as real contributors for the project [2].

## 3. Problems in automatic analysis

Given the semi–structured nature of the Linux Credits file, it is possible to extract data on kernel developers using automated software tools. A typical way of doing this is to use the Perl text processing language to write a program that scans through the Credits file and creates a database of Linux kernel developers. The snail–mail address fields, for example, can then be used to associate individual developers with geographical locations.

In practice, automated analysis is difficult. To see why, it is useful to present some example entries from the Credits file. These are discussed below.

**Table 2: An entry without country data.**

| |
|---|
| N: Lars Wirzenius |
| E: liw@iki.fi |
| D: Linux System Administrator’s Guide, author, former maintainer |
| D: comp.os.linux.announce, former moderator |
| D: Linux Documentation Project, co–founder |
| D: Original sprintf in kernel |
| D: Original kernel README (for version 0.97) |
| D: Linux News (electronic magazine, now dead), founder and former editor |
| D: Meta–FAQ, originator, former maintainer |

| |
|---|
| D: INFO–SHEET, former maintainer |
| D: Author of the longest–living linux bug |

Many entries in the Credits file do not include the snail–mail fields that contain the physical address. A trivial text scanning program could be written if all the entries in the Credits file would have the same structure and if all entries would contain full address information. Such a program would locate an entry by looking a line that starts with the N: –descriptor and keep scanning the lines until it finds a line with the S: –descriptor. As some entries in the Credits file do not have a field with the S: –descriptor, the program would, however, stop only at the next line where the descriptor can be found. This would associate a name with an address from a different entry, skipping some entries in the process. A casual browsing of some recent versions of the Credits files could easily lead a programmer to make such a mistake in programming the scanning algorithm.

**Table 3: An entry with non–standard address.**

| |
|---|
| N: John S. Marvin |
| E: jsm@fc.hp.com |
| D: PA–RISC port |
| S: Hewlett Packard |
| S: MS 42 |
| S: 3404 E. Harmony Road |
| S: Fort Collins, CO 80528 |

The Credits file also has a number of entries with non–standard addresses. For example, Table 3 shows an entry which gives good information about the mailing address in a way that is difficult for an automated program to detect. Several such non–standard addresses are listed in the table below.

**Table 4: Examples of non–standard address fields.**

| |
|---|
| S: Canada K2M–2G7 |
| S: No fixed address |
| S: Concord, CA |

| |
|---|
| S: Ann Arbor, MI |
| S: Oxfordshire, UK. |
| S: England |
| S: United Kingdom |
| S: UK |

A program that makes the assumption that the last S: –line contains the country name would find one Linux developer in a country called "Oxfordshire, UK," and another in "No fixed address." Without checking the validity of the country names, one could easily make wrong conclusions about the number of developers in different countries.

When attempts are made to create time–series data using several Credits files, the problems become more challenging. This is due to several facts. First, the entries in the older Credits files tend to be more incomplete and less structured than in the recent files. Although the structure of the Credits file was intended to facilitate automatic retrieval of information, the users were assumed to be able to make sense of the provided data. As a result, both the syntax and the semantics of the entries vary. For Linux programmers who use the Credits file to find information about other developers, the free format of the Credits file is not a major problem. They simply look up developer information and make sense of the provided information as well as they can.

It is relatively easy to interpret results produced by automatic extraction tools as long as the processed data can be manually checked. This becomes more difficult when several files from different time periods are used to process thousands of entries. Table 5 and Table 6 show entries with easy to interpret but difficult to automatically process country information.

**Table 5: An entry from kernel release 1.0, 13 March 1994.**

| |
|---|
| N: Alan Cox |
| E: iiitac@pyr.swan.ac.uk |
| E: gw4pts@gw4pts.ampr.org |
| E: GW4PTS@GB7SWN (packet radio) |
| D: NET2Debugged author |
| D: Network layer debugging |
| D: AX.25 & IPX alpha releases |
| S: <No> |

**Table 6: An entry from kernel release 1.3.6, 26 June 1995.**

| |
|---|
| N: Ken Pizzini |
| E: ken@halcyon.com |
| D: CDROM driver "sonycd535" (Sony CDU–535/531) |
| S: |

Table 7, in turn, shows an entry with an exceptional order of fields, where the S: –descriptors are placed in the middle of the entry. Such entries are rare in the Credits files and appear only in some kernel release versions. Scanning programs that are tested using only a few examples of Credits files easily become programmed with the assumption that the country information can be found by looking the last line with the S: –descriptor and that this is the final line of the entry, followed by an empty line. In most cases, this, indeed, is true. Sometimes the empty lines, however, contain space and tab characters that cannot be found by simple visual inspection of the files.

**Table 7: An entry from kernel release 2.0.1, 11 June 1996.**

| |
|---|
| N: Eric S. Raymond |
| E: esr@thyrsus.com |
| W: http://www.ccil.org/~esr/home.html |
| S: 22 South Warren Avenue |
| S: Malvern, PA 19355 USA |
| D: ncurses library co–maintainer |
| D: terminfo master file maintainer |
| D: Distributions HOWTO editor |
| D: Instigator, FHS standard |
| D: Keeper of the Jargon File and curator of the Retrocomputing Museum |
| D: Author, Emacs VC and GUD modes |

Longitudinal analyses of the Credits files also have to deal with the problem of address changes. For example, in the early Credits files Linus Torvalds has an address in Finland. In 1997 he moved to Santa Clara, California, and the address information changed accordingly. When

Credits files are used to analyze the country of origin of developers, the results therefore show a decrease in the Finnish population of kernel developers in 1998. Similar changes have occurred in many countries. In some cases the countries themselves have changed. Furthermore, some entries contain multiple addresses in different countries — for example, when the person in question is temporarily studying or working abroad.

## 4. Analysis method

As the discussion above shows, automatic generation of time–series data from the Credits files requires solving some non–trivial issues. The particular way these issues are solved leads to different quantitative results and interpretation of the produced data. This is also generally true for other studies that try to extract data from open source files [3]. It is therefore important that researchers both describe their methodology accurately and validate it using independent data.

Below I present data on developer countries. The starting assumption is that the snail–mail address reflects the location of the developer. Where country data is not available, other available information in the entry will be used to associate persons with countries. If there is not enough information in the entry, we revert to other sources. The data is generated using the following procedure.

1. If the last S: –descriptor line contains text, start by assuming that it is a country name.
2. If this text is badly formatted, translate it into an appropriately formatted country string.
3. If no country information is available, use a support database.

The second step above consists of program code that matches any of the known irregularly formatted S: –fields. For example, if the last S: –line consists of a string "idaho 83686," the entry is associated with "USA." Similarly, if the line contains a string "South Australia," the entry is associated with the country value "Australia."

When country information is not available, for example, because there are no S: –lines in the entry, a support database is consulted in an attempt to resolve the country. The support database associates names in the Credits files with countries. It is built in the following way:

1. If the entry has an e–mail address or other information that clearly indicates a location, use that as the country.
2. If there is an entry in a later Credits file with similar information but with country information available, use that country.
3. If common knowledge about developer country is available, use that.
4. If there is a non–obvious e–mail address or Web address in the entry, use an IP trace router program [4] to check the physical location of the network host; if the host seems to be a local host, check the registered physical address for the host using the RIPE, APNIC, and ARIN whois databases [5] and use that country.

5. If the country is still unresolved, use Internet search engines and interview experts to check whether there is good information available about the developer's location; if such information exists, use that.
6. If country information is still unresolved after the preceding steps, use "Unknown" as the country value.

Typically, many entries without explicit country information can be resolved using descriptions or e–mail addresses in the entry. For example, Lars Wirzenius at liw@iki.fi can easily be located in Finland, although his entry does not give country information. This assignment can be verified by common knowledge about the history of Linux. Similarly, although Alan Cox does not have explicit country information in the early Credits files, his e–mail address, amateur radio call sign and later entries assign him to the UK. Also in this particular case, the developer's country is common knowledge within the Linux community. In some cases, however, there is not enough information to associate a person with a country. In those cases, the support database defines the country as "Unknown." For example, the entry for Gareth Hughes only gives an e–mail address at acm.org, which is a global professional organization that typically only forwards e–mail. The support database will therefore include the following associations, among others:

Lars Wirzenius, Finland
Alan Cox, UK
Gareth Hughes, Unknown

As was noted above, the support database is used only if explicit country data is not available. For example, when Alan Cox has an entry with S: –line "<No>," this non–standard address string, which only Alan Cox has used, is translated into "UK." When he has no S: –line, the support database is consulted and Alan Cox is associated with the country found, in this case "UK." When Alan Cox has specified a S: –line with country information "United Kingdom" or "Wales, UK" they are simply translated into "UK." All these cases, indeed, occur in the different Credits file versions.

The procedure described above means that whenever there is an entry that cannot be associated with a country, and which has not explicitly been defined as "Unknown" in the support database, the scanning program generates an exception report. This means that entries that are labeled as "Unknown" have always been manually checked.

The support database has been built iteratively using a set of Credits files that range from March 1994 to July 2002. The procedure converges when no exceptions are generated. This also signals that the support database contains all the information that is required to process the entries in the Credits files.

After the support database is completely defined, the automatic procedure described above can be used to extract developer country information. By running this procedure over a set of Credits files, a time series can then be created. Possible irregular country names are checked at this point and the translation table is modified if necessary. Currently the translation table consists of 31 search strings that match all irregular addresses until release 2.5.25.

## 5. Results

Data from several versions of the Credits file are presented below. The processed Linux kernel releases are show in Table 8. The table shows the last modification dates of the processed Credits files, the total number of names found in the file, the number of times when the unavailability of explicit country information has required a lookup from the support database, and the number of names with an unknown country.

**Table 8: Processed Credits files.**

| Release | File date | Total names | Db lookups | Unknown countries |
|---|---|---|---|---|
| 1.0 | 13–Mar–94 | 80 | 13 | 1 |
| 1.1.0 | 19–Mar–94 | 81 | 14 | 1 |
| 1.1.23 | 6–May–94 | 86 | 13 | 1 |
| 1.2.0 | 6–Mar–95 | 128 | 23 | 1 |
| 1.3.0 | 11–Jun–95 | 129 | 24 | 1 |
| 1.3.6 | 26–Jun–95 | 130 | 24 | 1 |
| 2.1.0 | 21–Apr–96 | 196 | 27 | 1 |
| 2.0.0 | 8–Jun–96 | 190 | 27 | 1 |
| 2.0.1 | 11–Jun–96 | 190 | 27 | 1 |
| 2.1.44 | 7–Jul–97 | 209 | 40 | 2 |
| 2.1.108 | 13–Jun–98 | 245 | 34 | 2 |
| 2.2.0 | 19–Jan–99 | 269 | 41 | 2 |
| 2.3.0 | 11–May–99 | 275 | 46 | 2 |
| 2.3.9 | 29–Jun–99 | 287 | 42 | 4 |
| 2.2.14 | 4–Jan–00 | 284 | 58 | 2 |
| 2.3.51 | 10–Mar–00 | 341 | 48 | 4 |
| 2.4.0 | 31–Dec–00 | 375 | 67 | 5 |
| 2.4.6 | 28–Jun–01 | 395 | 73 | 5 |
| 2.4.9 | 13–Aug–01 | 403 | 71 | 6 |
| 2.4.17 | 21–Dec–01 | 408 | 72 | 6 |
| 2.4.25 | 6–Jul–02 | 418 | 77 | 6 |

A reasonable first assumption is that the mailing addresses provided by the developers in the Credits file reflect their physical locations. In theory, people may receive mail in an address that is in a foreign country, for example, if they are temporarily working or studying abroad. By manually checking different versions of the Credits files it is possible to see some movement across countries. In general, it seems, however, that mailing addresses reflect the main geographical location of the developers well. We therefore assume that when an explicit address is found in the entry it is the physical address of the developer.

In some cases, the developers have provided multiple physical addresses or e–mail addresses that point to different countries. In these cases, the country mentioned last is used.

Manual checking also reveals that that there are some entries that have not been updated during the years and which therefore may give misleading information about the current location of the developer. This is most probable for developers who have not been actively involved in Linux kernel development during the last years. As the Credits file is a cumulative record of the developers, a person who has developed Linux kernel ten years ago in a specific country may still be counted as a developer in that country simply because of outdated entry information.

It is not possible to detect a difference between active and passive developers using information from the Credits file. Although it can be safely assumed that new entries in the Credits file reflect active and recent developments, the earlier contributors may or may not be active.

When we associate names with countries it is important to note that this association has three possible interpretations. It is possible to assume that an explicitly given country reflects the current country of the developer. If the entry is updated and contains a physical mailing address, this probably is the case. In that case, we associate a person with his or her current location. When the regional expansion of the Linux developer network is studied, we usually, however, are interested in the country where the developers have done the work that has led to their inclusion in the Credits file. This may or may not be associated with the country where the developer currently lives. Furthermore, as the Credits file is a cumulative record of contributions, it does not differentiate between contributors that are active or passive.

A contributor who has developed Linux a decade ago appears in the Credits file exactly in the same way as a contributor who has recently made contributions. The aggregate country data therefore does not necessarily reflect current activity of Linux development.

When people have updated their address information after moving to a new country, as Linus Torvalds for example has done, this appears as a decrease in the number of developers in the originating country. If country information is extracted from a specific kernel distribution and used to associate the developer with a country at some other time period, the association may be wrong. It seems, for example, that some Linux developers have moved from Eastern Europe to the U.S. while they have been included in the Credits files. For example, Petko Manolov appears first as a Bulgarian developer in the year 2000, and then as a U.S. developer in 2002, leaving Bulgaria with zero developers. In other words, some of the growth in the U.S. developer numbers result from the movement of existing Linux developers and not necessarily from the emergence of new developers in the country.

In the table above, the column for Db Lookups shows the number of times when the support database has been consulted. This number gives a very conservative upper limit for possible errors in country allocations. In the worst case, the database could give wrong country information for all developers in the database. In many cases the data entered in the support database is, however, based on rather obvious cues. For example, it is relatively safe to assume that Ross Biro, with an e–mail address at stanford.edu, can be associated with the U.S.A., or that Kai Mäkisara at vtt.fi can be located in Finland. The association of Ross Biro with the U.S. can be confirmed by a Web search that finds him at the end of 2001 at San Francisco, working for The Learning Network. Kai Mäkisara, in turn, is clearly a Finnish name which fits well with the vtt.fi address that belongs to VTT Technical Research Centre of Finland.

In some cases, the association is more difficult to do. For example, Cyrus Durgin has only provided e–mail and Web page addresses at speakeasy.org, which resolves into an IP address 216.254.0.2. This belongs to a network that is registered in the ARIN database for Speakeasy Network, Inc., with an address in Seattle, Washington, U.S.A. Checking the Web site, one can see that the host is an independent U.S. Internet Service Provider, with strong roots in the Seattle area, but which operates in 11 points of presence in the U.S. and provides DSL access through a partner in 29 states. It probably therefore is correct to assume that Durgin is located in the U.S. A detailed study of information about Durgin finds him doing network and Unix administrator tasks in firms located in Seattle. Similarly, Daniel J. Maas, with a Web address http://www.maasdigital.com, has been located in Canada as his Web server seems to be located there. In some cases errors may have been introduced, as it is possible that people use their e–mail servers and Web hosts through international connections. For example, Daniel Maas also has Web pages at dcine.dyndns.org, located in the U.S., which inform that he has been studying at the Cornell University. A rough estimate is that there may be about a dozen country allocations in the most recent support database where the allocations have been based on indirect and difficult to verify IP addresses. Almost all of these are in the U.S. As it is probable that people on other continents do not commonly use U.S.–based Web hosts and e–mail addresses for their Linux–related activities, one may relatively safely assume that there are only few wrong country allocations in the support database.

The potential for wrong country allocations varies across countries and kernel releases. Recent kernel distributions typically have more detailed and more up–to–date information that can be verified through the Internet. Most of the missing country information is in the entries associated with U.S. developers. This is probably partly because physical address in the U.S. is often viewed as private information. Of the 22 Canadian entries only one entry did not explicitly specify the country in the most recent Credits file studied. Similarly, all developers from the Czech Republic explicitly gave country information. Of the twenty Australian developers five did not explicitly define their country of origin.

An important factor to note in using time–series data from the Credits files is that for a number of years there have been two different kernel distribution lines. One is intended for operational use, where reliability is more important than rapid incremental improvements. The kernel releases in this release path are commonly called "stable" versions. The other release path is used for ongoing development and incorporates more experimental additions to the kernel code. The Credits file versions have not been completely in synchrony between these two kernel release paths. As a result, the number of names in the Credits files does not always increase with time. For example, version 2.3.9, from 29 June 1999, has more names in its Credits file than version 2.2.14 that was released in January 2000. This can be seen from Figure 1 as an outlier point that breaks the relatively linear growth trend of the total developer count. The 2.3.x kernel release path is a developmental path, whereas 2.2.x is a "stable" path.

The summary data is shown in graphical form in Figure 1.

<img border="0" img src="https://firstmonday.org/ojs/index.php/fm/article/download/1151/1071/10258" alt="Figure 1: Number of people in the Credits file.">

**Figure 1: Number of people in the Credits file.**

The generated data consists of time–series of the number of people in the Credits files per country. This data can therefore be used to study regional distribution of Linux kernel developers across time. For example, it is possible to see how the number of contributors has changed in the different countries across different time periods. An example is shown in Figure 2, which shows the number of entries in a few selected countries.

<img border="0"

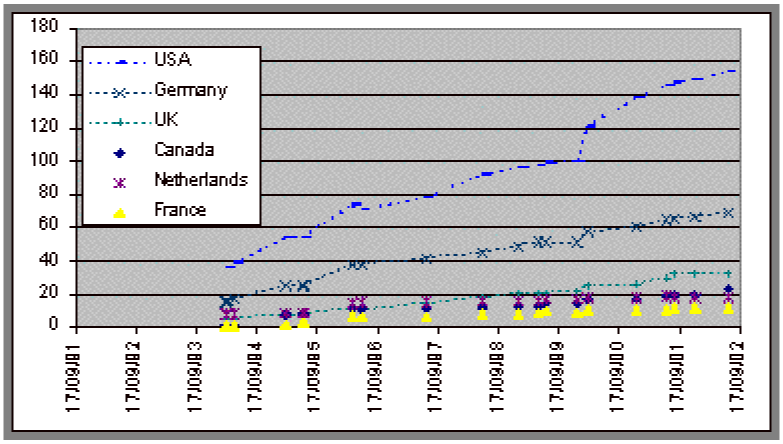

**Figure 2: Developers in different countries.**

The growth rates of developer populations have varied between different countries. This can be seen, for example, by fitting least squares estimates to the population weighted data and ranking the countries according to the estimated slope of growth. Results are shown in Table 9 below. To create the ranking, the country specific developer counts have been divided by the appropriate mid–year population of the country in question and a least squares linear curve has been fitted to the weighted country time series. The slopes of the fitted lines have then been used for the ranking. The U.S. Bureau of Census International Data Base has been used as the source of population data. For the estimates, an entry value of zero has been added for all countries, except Finland, at 17 September 1991, when the first Linux kernel was released. The starting value for Finland has been set to one.

The ranking shows that there have been countries both with slow and fast increases in the number of developers. Luxembourg appears on top of the list as its total population is only about 400,000 people and it has only one developer who appears in the Credits files since 1997. Leaving Luxembourg out, the quartile with the fastest growth includes Czech Republic, Sweden, Norway, Australia, Ireland, Denmark, the Netherlands, Germany, and Finland.

**Table 9: Country ranking according to population weighted growth rates.**

| Rank | Country | Slope | R2 |
|---|---|---|---|
| 1 | Luxembourg | 9.14E–10 | 0.76 |
| 2 | Czech Republic | 3.61E–10 | 0.90 |
| 3 | Australia | 3.05E–10 | 0.94 |
| 4 | Sweden | 2.85E–10 | 0.82 |
| 5 | Norway | 2.82E–10 | 0.78 |
| 6 | Finland | 2.57E–10 | 0.66 |
| 7 | Netherlands | 2.53E–10 | 0.84 |
| 8 | Denmark | 2.15E–10 | 0.87 |
| 9 | Germany | 2.11E–10 | 0.98 |
| 10 | Ireland | 2.09E–10 | 0.75 |
| 11 | Ukraine | 1.86E–10 | 0.98 |
| 12 | Austria | 1.77E–10 | 0.89 |
| 13 | Canada | 1.63E–10 | 0.93 |
| 14 | U.S.A. | 1.31E–10 | 0.97 |
| 15 | Belgium | 1.20E–10 | 0.91 |
| 16 | Croatia | 9.44E–11 | 0.78 |
| 17 | Hungary | 8.40E–11 | 0.85 |
| 18 | Switzerland | 7.93E–11 | 0.64 |
| 19 | New Zealand | 7.06E–11 | 0.50 |
| 20 | France | 6.03E–11 | 0.94 |
| 21 | Hong Kong | 4.67E–11 | 0.56 |
| 22 | Portugal | 3.04E–11 | 0.56 |
| 23 | Italy | 2.76E–11 | 0.92 |
| 24 | Bulgaria | 2.68E–11 | 0.34 |
| 25 | Romania | 2.03E–11 | 0.57 |
| 26 | Poland | 1.65E–11 | 0.61 |

| | | | |
|---|---|---|---|
| 27 | Taiwan | 1.53E–11 | 0.61 |
| 28 | Brazil | 1.51E–11 | 0.66 |
| 29 | Argentina | 8.97E–12 | 0.60 |
| 30 | Spain | 8.90E–12 | 0.63 |
| 31 | South Africa | 6.42E–12 | 0.50 |
| 32 | U.K. | 5.57E–12 | 0.56 |
| 33 | Mexico | 3.33E–12 | 0.59 |
| 34 | Russia | 2.62E–12 | 0.72 |
| 35 | Japan | 1.86E–12 | 0.44 |

An example of population weighted entry counts is shown graphically in Figure 3.

<img border="0"

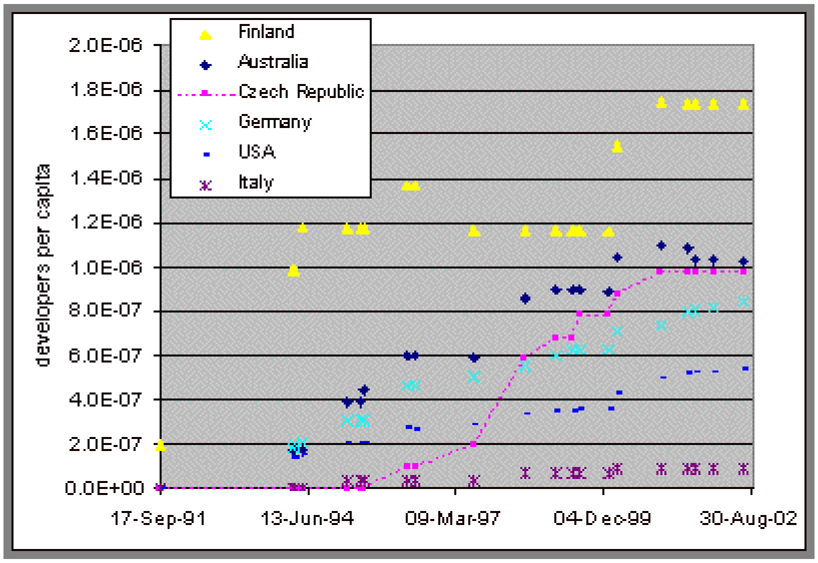

**Figure 3: Number of entries in some example countries, per capita.**

Many of the country rankings are statistically meaningless as there is a large number of countries with only few developers. In the last kernel release studied, there were 28 countries with ten or fewer developers and only seven countries with more than ten developers. The median number of developers was 2 and the average 11.7. The number of developers in different countries in the most recent kernel version 2.5.25, from July 6, 2002 is shown in Table 10. The full data set is in Appendix 1.

**Table 10: Number of developers in different countries in Linux Credits, version 2.5.25.**

| Country | Number of developers |
|---|---|

| | |
|---|---|
| Bulgaria | 0 |
| Argentina | 1 |
| Croatia | 1 |
| Hong Kong | 1 |
| Japan | 1 |
| Luxembourg | 1 |
| Mexico | 1 |
| New Zealand | 1 |
| Portugal | 1 |
| Russia | 1 |
| South Africa | 1 |
| Taiwan | 1 |
| Ukraine | 1 |
| Ireland | 2 |
| Poland | 2 |
| Romania | 2 |
| Spain | 2 |
| Switzerland | 2 |
| Hungary | 3 |
| Austria | 4 |
| Belgium | 4 |
| Denmark | 4 |
| Norway | 4 |
| Italy | 5 |
| Finland | 9 |
| Sweden | 9 |
| Brazil | 10 |
| Czech Republic | 10 |
| France | 12 |
| Netherlands | 18 |
| Australia | 20 |
| Canada | 23 |
| U.K. | 32 |
| Germany | 69 |
| U.S.A. | 154 |

| | |
|---|---|
| *EU countries* | *187* |

## 6. Credits and authorship

A recent think–tank report by Alexis de Tocqueville Institution (Brown and Orndorff, 2004) used a draft version of this paper to argue that Linux could be based on intellectual property infringements and inaccurate allocation of developer credits. The report claimed that the history of Linux is too amazing to be true, and that it is improbable that a single individual without much experience in software development could have created a full operating system in just a few months. The report implied that Linux, therefore, could be derived from earlier software code and, in particular, from the Minix operating system. The report also claimed that the analysis of the Credits file shows that Linux developers may have acted against their employers' intellectual property policies, and that missing entries in the Credits file may indicate that Linux may include copyrighted code that has not been acknowledged. Hence, the authors of the report claimed that the future of open source software and Linux is therefore threatened by the problem of assigning authorship to specific pieces of code, and potential legal costs resulting from this. As the argument to an important extent has been based on the data presented in this paper, a few observations may be useful.

First, as Section 2 above pointed out, the amount of code in the first release of what later became known as the Linux operating system was rather modest. It consisted of 88 files, with median size of 37 lines of code. Based on common knowledge about software development, it therefore appears that a single computer enthusiast could well have created the first Linux version in a couple of months. In fact, by reading the original source code, it is quite clear that a single author, still in the early phases of learning to program operating systems, has produced it. From the point of view of copyright law, the first version of Linux kernel therefore cannot be defined as a derivative work [6].

Second, the unavailability of Credits file during the early phases of Linux development does not signal unclear authorship. The fact that the first Credits file appears only in 1994, with Linux version 1.0, has a very simple explanation. During the early phases of the development, the amount of source code was small and the developers were aware of each other's contributions. Explicit recording of "credits" would have made little sense, as all developers knew where each piece of software came from. In fact, the development process was based on iterative development and improvement of source code. The developers, therefore, did not only know who had produced specific components of the system, but they also knew the complete history of these developments. Explicit Credits files only started to make sense when the Linux project grew to a more complex development project where newcomers did not necessarily know the history of the project [7].

Third, the Credits files do not record authorship in any legal sense. Formal copyrights and informal moral rights for the different parts of the Linux source code are embedded in the source code. The Credits file only acts as a "hall of fame" without any direct links between copyrighted work and authorship. The Credits file, therefore, is irrelevant from the point of view of intellectual property rights. The addition of Linux contributors to the Credits file is controlled purely by behavioral rules decided by the developers themselves.

Fourth, the affiliations given in the Credits files do not necessarily reflect the historical affiliations of the people mentioned in the file. The Tocqueville report, for example, argued that Linux developers may have harmed their employers' intellectual property rights by releasing software to the Linux community. To make this argument, the report shows that an employer of a person mentioned in the Credits file has adopted an intellectual property policy, according to which only software that cannot be appropriated commercially may be released as open source. This would apparently imply that the Linux developer mentioned in the Credits file might have broken the intellectual property policies of his employer by providing code to the Linux kernel, and that his managers might have acted against the interests of the firm and its owners by allowing this to happen. As the Credits file is an accumulated record of contributors and as the current addresses and affiliations of the developers have changed during the years, the policies of the current employers, however, are quite irrelevant for any discussion on historically created intellectual property. For example, Linus Torvalds remains one of the main contributors to the Linux kernel project and his name will appear in the future versions of the Credits file, independent of the intellectual property policies of his current or future employers. Microsoft, for instance, could not retrospectively create copyright policy infringements simply by hiring all the persons mentioned in the Credits file, even if this would imply that all the developers would change their affiliations to Microsoft in the future versions of the Credits file, and subsequently be required to stop producing open source software [8].

The difficulty to accurately allocate credit in software development projects should not, however, be automatically interpreted as evidence of misallocated credit or intellectual property rights infringements, as the Tocqueville report, for example, has done. Software products are often based on incremental innovation where existing technologies and knowledge are recombined to create new functionality. The history of the Internet, for example, shows that authorship, indeed, is often misallocated [9]. This fact, however, could easily be used to argue that the current intellectual property regime — and specifically software related patents — may require reconsideration. In networked and combinatorial systems, intellectual property rights probably are often granted to inventors who only partially deserve the credit. Or, to put it in another way, developers may deserve much more credit than there is intellectual property available today. One way to deal with this issue is to create explicit representations of moral authorship that are only loosely connected with current concepts of intellectual property. The Linux Credits file is an example of such an approach.

## 7. Discussion

Automatic extraction of data from open source code is a possible, but not a trivial, task. Computer programs typically operate on relatively well–defined tasks where users have enough domain specific knowledge to detect programming errors and where statistical and case–based testing can, at least in theory, be applied. Text scanning programs, in contrast, often operate on open and ill–defined domains. Text can be used in many different ways for communication. Interpretation of semantically ambiguous text, however, is usually impossible without context information and human sense–making capabilities. The programming of text analysis tools, therefore, typically requires detailed domain specific knowledge.

Errors in traditional computer programs can often be detected by the fact that the program crashes or that it produces obviously incorrect results. For example, operating systems function in a relatively closed world that is highly constrained by the underlying hardware. If program code has errors or if the programmer has made mistakes in interpreting the world, reality usually hits without much delay. In such a world, the hardware provides a foundation for objective evaluation of software errors.

Text processing programs that extract data from unknown domains operate in a very different world. This world is full of surprises and unexpected phenomena. When the size of the studied domain grows, it becomes increasingly difficult for the researchers to understand where and how their tools work. In practice, when automated text analysis tools are used in areas where their correct operation has not been validated, the results become unpredictable. It is particularly difficult to understand and interpret data that has been automatically generated if no analyses of potential sources of error and estimates of their impact are available.

One approach to alleviate this problem is to benchmark and test automated data extraction tools against a known set of data. Such benchmarking data can be created in a bootstrapping process that starts by analyzing a limited and well–defined domain where potential errors and their impact can be studied. In effect, we can isolate a small micro–world from a potentially open and ambiguous world and improve the research methodology within the micro–world until the results converge.

Methodologically, this is a variation of the grounded theory approach, often used in qualitative social studies (Glaser and Strauss, 1967). When the automated tools produce conceptually coherent results — for example categorize the data in known countries — and the tools do not produce any exception reports, the domain has been adequately understood. The results can then be verified using independent sources of knowledge to check that the converged results are valid. For example, we can study the list of generated countries and check that they are countries also in the real world. Indeed, we can see that some "countries" such as Hong Kong and Czech Republic have changed their status during the history of Linux [10]. After one micro–world has become understood and known, broader studies can then verify their results and estimate their errors using data from previous studies.

Open source research that is conducted using automated text processing tools is particularly challenging. As many open source projects have detailed historical archives of the different versions of the software, it is possible to generate time–series data that describes the evolution of the studied systems. Combined with other statistical data, such data can provide interesting theoretical and empirical insights. Typically, the massive amounts of text in open source archives, however, means that automated processing is necessary. If the resulting data is used to test theoretical hypotheses, the results depend on the quality of data. As the discussion above shows, there is a considerable risk that theorists get the basic facts wrong.

By reading the various open source research papers available on the MIT open source archive [11], one may note that conceptual and empirical confusions have been relatively common in this new and multidisciplinary research area. Many authors, for example, have repeated Raymond's (1998) description of open source development as a "bazaar" that provides an alternative to the traditional "cathedral building." Historically cathedrals, however, were often built in ways that closely resemble the open source model [12]. Inspired by Raymond, some authors have built economic models based on the assumption that the open source model is "better" or "more reliable" than the proprietary software development model, without much evidence to support and qualify these beliefs. Early open source papers often took for granted that there is a unified "open source community" with shared values, motives, and development approaches. Some authors have struggled with technical details, for example, by describing the Unix operating system as a computer programming language, or by explaining the modularization of the Linux operating system as a result of modern object–oriented programming practices which, in fact, have not been used in Linux development. The Tocqueville report, discussed above, makes similar mistakes, for example, by claiming that Linux has been the only commercially successful open source operating system, and that a single individual created it in just a few months.

In new research domains, errors are easy to make and also the academic peer–review system has difficulties in reviewing the quality of research. Without sufficient domain–specific knowledge and existing benchmarks it is difficult to understand the empirical validity and relevance of new research. The rapid publication cycle–times in open source related research and the broad availability of working papers on the Internet exacerbate the problem of scientific quality control as erroneous information is easily available and spreads quickly.

An example may illustrate these problems. It appears that the first study on the Linux Credits files was reported by the present author in a working paper distributed at UC Berkeley and Stanford in April 2000. Some of the results were presented in a conference organized at Berkeley, and slides from this presentation subsequently became incorporated in a BRIE (Berkeley Roundtable on the International Economy) working paper (Weber, 2000), with permission, but without reference to their original source [13]. The working paper has been on the BRIE Web site since the summer 2000 and linked to the MIT open source working paper repository since December 2000, and it has apparently inspired later studies that have tried to extract data from the Linux Credits files. Both automated and manual bibliometric studies would always miss the fact that some of the data came from the present author. This connection can only be made by people who already know about it.

It is clear that both in open source code and scientific literature credits are not always accurately recorded. Whereas the scientific quality control system tries to improve the accuracy of the attribution of authorship, the Linux credits files explicitly note that the list most certainly is only a partial list of contributors. Linux and other open source developers, for example, rely on the

GNU C compiler and its libraries but the contributions of the compiler developers are not explicitly noted in the source code files.

Explicit credit claims and copyright notes in open source code do not therefore necessarily accurately reflect authorship or developers' contributions. Furthermore, as sociologists and historians of science have shown, institutional memory tends to be very selective [14]. Earlier contributions are often forgotten when complex social forces produce reinterpretations of history [15].

Although open source research has a unique opportunity in penetrating the barriers of institutional memory and avoiding social distortions in the allocation of credit, memory is also overwritten in the Linux kernel contributions. The different parts of the Linux kernel code have undergone considerable creative destruction during the years [16]. Old code has continuously been replaced and made redundant by new code. Linux development is cumulative system development where early contributions may be major contributions even when they are later discarded and made invisible. As the Linux kernel developers have used many different ways to tag their contributions in the source code files during the years, a detailed study of kernel contributions is a very challenging task indeed. This is the reason why we have focused above on the relatively simple task of analyzing the Linux Credits files.

The data presented in this paper can be used to cross–verify the methodology and results of several papers that discuss the structure and development of the Linux developer community. For example, David Lancashire (2001) has argued that the analysis of the Linux Credits files shows that the high concentration of Linux developers in the Nordic countries and the low per capita number of developers in the U.S. reflects the different structure of economic opportunities in the different regions of the world. This is an interesting argument. Based on the discussion above, it is, however, easy to see that it is not supported by the data provided by Lancashire or the data described in this paper.

The core of Lancashire's argument is that the high demand for computer programmers in the U.S. during the 1990s has led to an abundance of economically beneficial opportunities, implying high opportunity costs of joining open source projects in the U.S. Lancashire therefore concludes that traditional economic arguments can explain the geographical distribution of Linux kernel developers. This conclusion, however, is based on data from a single kernel release (2.3.9). As should be clear from the above discussion, Linux Credits files are cumulative records of persons who have been involved in Linux development. The snapshot picture provided by a cumulative record of Linux developers should therefore be complemented with a more dynamical analysis.

A quick glance at Figure 2 reveals that there does not seem to be any obvious slowdown in the growth rate of Linux developers in the U.S. during the Internet boom years. When the developer data is aggregated in the European level, the European Union (EU–15) has about 20 percent lower developer density than the U.S. In other words, contrary to Lancashire's claim, there does not seem to be any "apparent erosion of American support for free software development." Similarly, the population–weighted developer counts in Figure 3 do not seem to support the

Lancashire argument about opportunity costs. As can be seen from Table 9, the story is more complex.

In particular, to support the claim that economic opportunity costs explain the geographical distribution of Linux developers, one should analyze the relationships between the growth rate in the developer population in a specific country and the opportunities in that country. It is obvious that as only 35 countries of the world have Linux developers with entries in the Credits files, the concept of opportunity and its related costs can only be understood within a larger socio–cultural context. For example, there are no developers from India, mainland China or Islamic countries. The fastest growth in developer populations has in recent years come from the Czech Republic and Australia. Most spectacularly, 99.6 percent of Linux developers mentioned in the Credits file are male. A simple economic theory of opportunity costs would therefore imply that women and men exist in two different economic spheres, or that women are paid much better for commercial software development than men.

## 8. Future research

Already a quick glance at Figure 3 produces several alternative hypotheses to the thesis that opportunity costs or wages in the local software industry can explain country differences in the Linux developer community. Finland seems to rank high partly because of the historic origin on Linux. The most important distribution host in the early years of the Linux history was funet.fi, located in Finland. As many Finnish hackers had good access to this Internet node, they were well aware of the evolution of the Linux system. The relatively high Finnish participation in the Linux development activity, therefore, seems to be more a story about innovation diffusion than a story about opportunity costs. The facts that Finland ranks highest among the countries in the proportion of science and engineering students [17] and that it was among the first countries to be connected to the U.S. Internet backbone [18] most probably also play a role here.

It is also worth noting that technological change has been an important factor in the expansion of Linux. The first Unix was developed at the Bell Labs in 1971 for PDP minicomputers. Towards the end of the 1970s, several alternative versions of the Unix operating system were created, most important of these being the UC Berkeley BSD Unix, which had strong support for computer networking. With its support for TCP/IP networks, Unix became the most popular operating system in the Internet. The Minix operating system, released in January 1987, was the first Unix–like operating system developed for cheap Intel–based PCs. Source code was available for all these systems. In this sense, Linux is a fourth generation Unix. It was made possible because PCs and the rapidly expanding Internet brought collaboration within the reach of computer enthusiasts, combining personal computing with global networking. The fact that these factors came together in many different countries almost simultaneously is reflected in the fact that the Linux developer network rapidly expanded to many countries. In this sense, Linux emerged exactly when it became possible.

The rapid recent expansion of Linux development activity in the Czech Republic is probably at least partly related to increased access to computer networks [19]. One might also speculate that the relatively low participation of Italy, France, and Spain may be partly explained by language barriers. These countries, of course, also had some of the lowest Internet penetration rates in Europe during the 1990s.

An interesting phenomenon emerges when the data presented above is extended to recent versions of the Credits file and complemented with information in the Maintainers file that describes the division of labor within the Linux developer community. At the end of 2003, there were about 250 persons maintaining specific parts of the Linux system. The Credits file, in turn, had grown to about 450 names and 35 countries. The developer affiliations increasingly appeared to be large businesses and government–related agencies. For example, the modules relating to system security apparently are now professionally developed and managed by people in firms that are known for their close contacts with the U.S. government. As Linux has increasingly been accepted as a core element in corporate and government organizations, developers are increasingly difficult to describe in terms that could have been relevant in the early phases of the Linux history. Future research could find a rich domain of study by analyzing the evolution of organizational and institutional ecologies in the history of Linux.

The data extracted from the Linux Credits files can be used to test a number of hypotheses about the factors that influence the growth and expansion of open source networks. For example, it is possible to study the correlations between national educational levels, information technology investments, growth of software related industries, cultural communication patterns, and expansion of open source networks. Furthermore, it is possible to ask why some countries are not involved in the Linux development.

Conclusions from such studies will be limited by the fact that Linux is a quite exceptional open source project. It is a highly successful project, which has attracted much interest, economic support, and millions of users. Linux is also special, as it has been an operating system project, where software architecture has been strongly constrained by hardware. Results from the analysis of the Linux Credits files, therefore, cannot necessarily be generalized into other successful open source projects. In particular, the boundary conditions that have made the Linux development model effective may have changed during the years. Indeed, the Linux developer community has not only created the Linux system itself but also many tools and procedures that make the current development process possible. These tools and procedures were not available for the early developers of Linux. For example, many recent open source research projects have tried to analyze the evolution of open source code and its developer communities using information recorded in software version control systems. Such information, however, cannot be used to analyze the history of Linux, as version control systems have not been used in Linux development until recently (Shaikh and Cornford, 2003).

The developmental path that led to the current Linux system may be difficult to repeat a decade later. Today, for example, it would be quite unrealistic to send an e–mail to an Internet newsgroup announcing the availability of 88 files that contain a rudimentary Unix–type operating system, and expect a tidal wave of interest. The world of computing has changed in many ways during these years. Linux itself has been an important part of this change. Theoretical generalizations that are based on the history of Linux, therefore, may require careful analysis of the changes that have occurred during the evolution of Linux. We can learn from history, of course, but technological change also continuously creates new domains for human activity. Technological development makes time inherently irreversible. In such emerging worlds, universal truths are far and few, and difficult to find before they become outdated. Better understanding of what exactly happens in open source projects, and how they became what they are, is, however, extremely important in a world where software both drives and facilitates social and economic development.

## About the author

Ilkka Tuomi is currently Visiting Scientist at the European Commission’s Joint Research Centre, Institute for Prospective Technological Studies, Seville, Spain. From 1987 to 2001 he worked at the Nokia Research Center in various positions, most recently as Principal Scientist, Information Society and Knowledge Management. From June 1999 to December 2000, he was Visiting Scholar at the University of California, Berkeley.
E–mail: Ilkka.Tuomi@cec.eu.int.

## Notes

1. There are six names in the Credits file whose gender could not be verified by searches on the Internet or by asking other persons mentioned in the Credits file. As no information indicated that the persons in question would be female, I have made the assumption that they are males. This assumption was made for Sam Mosel, Pat Mackinlay, Niibe Yutaka, Chih–Jen Chang, Ani Joshi, and Asit Mallick. In general, as the developers come from many different cultures that have different naming conventions it is not a simple task to categorize them based on gender. Also Linux kernel developers sometimes make mistakes in this regard. For example, in some kernel discussions Andrea Arcangeli is referred to as a female. It may be easy for a Finnish speaker to know that Kai is a male name in Finnish. In some other languages it would be female name.

2. One should note that inclusion in the Credits file does not always imply major contributions to the Linux project. There are also some developers included who have made relatively minor contributions. I am grateful for Alan Cox for clarifying this point.

3. The FLOSS study by Ghosh, Robles and Glott (2002), for example, used an algorithm that assumed that source code has copyright lines where a developer is associated with the copyright. As a result, firms such as Sun Microsystems or institutions such as University of California at Berkeley were counted as particularly active software developers.

4. IP trace router is a program that sends IP packets to a specified destination and notifies the sender about all the intermediate hosts that the packet traverses to reach its destination. Using the generated routing trace it is possible to see where the destination host is physically located.

5. Internet domains are registered in RIPE, APNIC and ARIN databases depending on their geographical origin and domain type. RIPE maintains information on networks registered in the European, Middle East, Central Asia and African countries north of the equator, APNIC on networks in Asia–Pacific, and ARIN on networks in the American, Caribbean and Sub–Saharan countries, as well as networks registered in Antarctica and the .com, .org, .mil, .edu, .gov, and .net domains. The whois program can be used to find out in which network a specific IP address is registered and who has registered that network domain. It is possible to manage networks across country borders and operate country specific domains outside their real physical location. This typically occurs only for domains such as Tuvalu (.tv) and Coconut Islands (.cc) which are easy to detect as virtual addresses.

6. Copyright only protects expression, and the fact that Linux development was influenced by general knowledge about Unix–related operating systems is therefore irrelevant from the legal point of view. The Tocqueville authors apparently confuse copyright issues with the fact that in commercial software development developers often try to avoid any contact with earlier competing products. Software developers often avoid contact with earlier products to pre–empt any possible future claims that their work has been based on competitor’s business secrets and to prove that any such claims would be frivolous. This is a common legal strategy in highly competitive environments where lawsuits are used to deter competition. Ideas, however, are not protected by copyright. The report’s extensive discussion on the possible influences of Minix on the development of Linux, therefore, is historically interesting but has no consequences from the copyright point of view. Similarly, the analysis of source code structures and logic using "pretty printers" or reverse engineering software, proposed in the report as a means to reveal influences between software projects, is irrelevant when the issue is about copyrights. One should also remember that intellectual property rights are always granted under constraints that balance private and public interests. For example, copyrights are combined with "fair use" rules that, for instance, give citation rights. Similarly, patent monopolies are granted in exchange of revealing the underlying knowledge so that it can lead to further inventions and improvements. Even if Minix would have been both copyrighted and patented, reading its source code, therefore, would not be a problem from the intellectual property point of view, unless the new code would have incorporated pieces of earlier code or infringed a patent.

7. I tried to clarify this point to the lead author of the Tocqueville report in an interview referenced in the report, apparently with limited success.

8. Microsoft is used as an example here, based on a discussion with the report’s lead author, available information about the funding of the Alexis de Tocqueville Institution, and the specific way the report describes the evolution of Linux and its Credits file. It appears that the report was probably written mainly to promote the strategic interests of Microsoft among U.S. government policy makers.

9. Examples include packet switching, the Internet, and the World Wide Web. I have discussed these examples and their implications to intellectual property rights and innovation theory in Tuomi, 2002.

10. For example, I have used the population of Hong Kong for population–weighted data, instead of using the population of People's Republic of China.

11. http://opensource.mit.edu.

12. Tuomi, 2002, p. 164.

13. The paper was labeled as a working draft, with a request not to cite or quote the paper. It has, however, become widely quoted.

14. Tuomi, 2002, chapter 9.

15. Although Linus Torvalds typically makes ironic statements about his role as the "creator" of Linux, popular accounts often take these statements seriously. The conventional heroic model of innovation requires heroes. This model does not fit well with historical facts, and it is particularly unsuitable for Internet–related innovations, but usually this means that the facts are adjusted so that a heroic story can be told. Thus, for example, Linus Torvalds emerges as the inventor of GNU/Linux and Tim Berners–Lee as the inventor of the Web.

16. Tuomi, 2001; 2002, chapter 10.

17. At the end of the 1990s, Finland had the fourth highest per capita count of engineering graduates in the world, after Singapore, Korea, and Japan.

18. Finland was connected to the NSFNET backbone together with the other Nordic countries and Canada at the beginning of 1989. The Finnish Unix hackers requested a country code for Finland already in December 1986.

19. It is also related to SuSE's expansion to Czech Republic. I'm grateful for Roland Dyroff for pointing this out.