Ivan F. Villanueva B.
Pflugstr. 10/4
10115 Berlin
Germany Tel. +49 160 23 160 13
Email: ivan@ogai.name

April 30, 2009

European Patent Office
The Registry of the Enlarged Board of Appeal
Fax: +49 89 2399 3014

Case number G 3/08

The Stop Software Patents coalition is a broad Initiative in the tradition of the Eurolinux-Alliance, which, in the course of the Software Patent directive proposed in 2002 and rejected in 2005, collected more than 500.000 signatures of concerned EU citizens against the European Patent Office's practise to grant patents on software.

Our coalition is also a revival of the Economic Majority Project [1] initiated by FFII e.V., which was entrusted by 1,948 companies, with a minimum of 31,503 employees and annual turnover of 3,258,244,082 EUR, to defend their interests, which they define as follows:

> Our enterprise is worried about plans to legalise patents on software solutions ("computer-implemented inventions").
> We rely on software copyright. We need to be sure that we own what we write. We need to be sure that we can publish and distribute our own programs. We need to be sure that, as long as we respect the rules of copyright, we can run any software on any office or network computer.
> We urge legislators to confine the patent system strictly to the limits of applied natural science. In principle, only knowledge which had to be obtained through costly experiments with forces of nature should be eligible for the broad, slow and expensive

[1] `http://economic-majority.com/`

monopoly protection which the patent system offers.

We have recently started a new petition webpage [2] in which we expect a much bigger number of companies. So far 14.441 European individuals (mainly programmers), 1.272 European companies and 188 European associations with 69.976 members have signed it. We expect much bigger figures during 2009. The text says:

> **Introduction**
>
> Our petition aims to unify the voices of concerned Europeans, associations and companies, and calls on our politicians in Europe to stop patents on software with legislative clarifications.
>
> The patent system is misused to restrain competition for the economical benefit of a few but fails to promote innovation. A software market environment is better off with no patents on software at all. Healthy competition forces market players to innovate.
>
> European court decisions still accept in many cases the validity of the software patents granted by national patent offices and the European Patent Office (EPO) that is beyond democratic control. They not only continue to grant them, but also to lobby in favor of them. Despite the current deep crisis of the patent system, they are unable to reform and put at risk too many European businesses with their soft granting policy.
>
> On 2005 the Commission appeared to be more supportive to the interests of major international conglomerates than of small and medium sized enterprises from Europe - who are a major driving force behind European innovation. The European Parliament rejected at the end the software patent directive, but has no rights for legislative initiatives.
>
> **Studies**
>
> A large number of serious scientific and economic studies justify ruling out patents on software. Copyright for software, but no patents
>
> Software authors are already protected by copyright law, allowing others to innovate in the same space generating healthy

[2] `http://http://stopsoftwarepatents.eu/`

competition, but this protection is undermined by patents on software. It is far too easy to violate patents on software whilst being completely unaware of any transgression. Software companies do not use and do not need the patent system to innovate. They must be protected from owners of dubious granted patents.

Litigation instead of innovation

Software patents miss their legitimate purpose. Patents on software favour litigation in detriment of innovation, defeating their democratic justification. They force software producers to spend on bureaucracy, lawsuits, and circumventing dubious granted claims on software what would otherwise be spent on Research and Development. Owners of patents on software, who sometimes doesn't produce software themselves, obtain a means to exert unfair control over the market.

American mistakes

In the USA there are billions of dollars in litigation over software patents each year, and not only between software companies, but also other companies just because they have a web site (this starts to happen in Europe also). This mistake needs to be avoided in Europe.

We urge our legislators

- to pass national legal clarifications to substantive patent law to rule out any software patent;
- to invalidate all granted claims on patents that can be infringed by software run on programmable apparatus;
- to also strive to propagate these rules to the European level, including the European Patent Convention.

We are grateful to the EPO for accepting written statements from any third party on the referral G3-08. We hope with our statement to be useful to the honorable board members which have been given the task no less than to answers questions with an enormous complexity and implications that will considerable influence our fast becoming digitally interconnected societies, as Internet and standard computers are deeply interconnected with software patents.

However, we think that this referral represents the last hope of the current patent system to heal itself. Neglecting the democratic decision of the only EU institution elected directly by Europe's citizens, the European patent office has continued its practise of granting patents on software until this day.

Therefore, if the pending decision should fail to reconquer the significant meaning of the provision in Art. 52 (2) (3) EPC, we see no alternative to the legislator stepping in and reforming substantial patent law as stated in our petition.

As an alliance not limited to, but especially consisting of programmers and business investors, we consider of vital importance that the EPO clarifies, in terms understandable not only by law professionals, whether software run on standard computers can infringe patents. **Could a computer using 20 years old hardware technology, which is connected to the Internet, infringe a valid patent in the opinion of the board?**. This is a vital question in our opinion, as also the questions by Lord Justice Jackob in *[2006] EWCA Civ 1371*, that, due to their importance, we copy here in the hope that the board members will keep in mind when answering the other questions in the referral:

> (1) What is the correct approach to adopt in determining whether an invention relates to subject matter that is excluded under Article 52?
> (2) How should those elements of a claim that relate to excluded subject matter be treated when assessing whether an invention is novel and inventive under Articles 54 and 56?
> (3) And specifically: (a) Is an operative computer program loaded onto a medium such as a chip or hard drive of a computer excluded by Art.52(2) unless it produces a technical effect, if so what is meant by ?technical effect?? (b) What are the key characteristics of the method of doing business exclusion?

The current European software business market is developing much more faster now that it was in the past thanks to open source software [3] , open

[3]Open Source Software is used by many governmental institutions like the German Foreign Affairs Ministry. Most prominent open source software are: *Linux/GNU* made by the (Kernel developers including big companies like IBM) and the Free Software Founda-

standards [4] , open platforms [5] used in mobile phones technology, or extensible web applications [6]. . Only if a clearly line is drawn between patentable and not patentable subject matter, companies (and also open source developers) can assert their investment costs for new businesses.

It is our intention to be helpful to the members of the board explaining from a programmer and businessman perspective how software is created and used, and how technology related to software has changed. As the many prominent philosophers of the 20th century [7] have made clear, it is not possible to understand concepts without looking at their history and at their usefulness both in a conceptual system (when classifying entities) and pragmatically in the real word. An in-depth analysis of the concept of *software* related to others like *hardware*, and the analysis of its uses in the discourse and in the real world, is considered by us a necessity. As it is an analysis and definition of the concepts like *technical effect*. The first time frame begins in the 17th century with the invention - or maybe should it be called discovery? [8] - of the binary (or digital) system, and it ends when the EPC was signed. How *software* was understood at the time the EPC was signed is crucial to interpret the meaning of *software as such*. The second time frame analyzed in this Amicus Curiae Brief ends when the EPC was revised in 2000. The third time frame longs until now.

At the end we will try to answer the questions proposed by the head of the EPO. in a way that could *make happy* the proponents of software patents

tion, the Internet browser *Firefox* made by the Mozilla Foundation, and the office suite *OpenOffice* whose development was coordinated among others by the big company Sun Microsystems.

[4]The Internet protocols for instance are basically a conglomerate of non-patented open standards that anyone can use for interoperating with other computers connected with the network, making easy for computer experts to create web pages, send emails, and other Internet services.

[5]Most prominent examples are the *Java Micro Edition*, *Android* and the free SDK of the Iphone

[6]like the social networks Facebook and MySpace, which their users can extend with their own software

[7]We are referring to the main work of Theodor W. Adorno, J?rgen Habermas, Ludwig Wittgenstein, Hans-Georg Gadamer and Martin Heidegger.

[8]Many philosophers following the tradition of Platonic idealism state that certain kind of knowledge, like mathematics, is just remembered, found or perceived, but not invented

as well as its critics. The two groups were clearly identifiable during the debate about the European CII directive proposal. On the one hand we have big software companies with a frightening software patent portfolio, the so maliciously called *EPO customers* and *monopolists* by the other group; on the other hand we have an enormous amount of individuals and small and middle European enterprises responsible for the majority of jobs in Europe in the sector. [9]

What we think has always been the intention of the legislator is to promote innovation, which sometimes requires a protectionist patent system that guaranties the return of investment, and at the same time to avoid that this system is misused to monopolise knowledge that needs to be independently available for anyone, like mathematics or scientific theories. But what about software?

From an economic point of view, many studies [10] clearly criticizes software patents as a mean to promote innovation and growth in the software market. But why then have the patent and law experts, working at the EPO, its Board of Appeals and at courts, being slowly moving from a restrictive interpretation of EPC-52 to another one that is now causing an economical damage in innovation terms? Many companies are investing now more in lawsuits regarding software patents than in research and development. [11] It is impossible, and not only in economical terms, by small companies to research for possible patents that could be seen as violated by their software products. Additionally they don't use the patent system. [12] Such companies are by no means plagiarists. Defensive patent pools [13] are an option for some middle sized enterprises but definitely not for small ones. Additionally, a

[9]See research of the *Institut f?r Mittelstandsforschung (IfM) Bonn.*

[10]The list is very long, just a selection: Study of the Deutsche Bank Research (2004-06-22), An Empirical Look at Software Patents by James Bessen and Robert M. Hunt (2003-5), Sequential Innovation, Patents, and Imitation - MIT Department of Economics Working Paper by James Bessen and Eric Maskin (2000), Opening Doors and Smashing Windows: Alternative Measures for Funding Software Development by the Center for Economic and Policy Research (CEPR) (2005-10)

[11]

[12]See statements of entrepreneurs for instance at `http://www.economic-majority.com/testimony/index.en.php` and `http://www.patentfrei.de/`

[13]Two examples of such defensive patent pools are `http://www.openinventionnetwork.com/` and `http://www.patentverein.de/verein.php`

system that obliges its players to invest more in protecting themselves from attacks [14] by others instead of research and development cannot be called something else than *perverted*, and remembers the weapons proliferation logic (including atom bombs) at the time of the Cold War.

Patenting in the software and business method field has become a serious determent to investment in ICT markets, largely because of the uncontrollable risks of patent infringement. Patent infringement happens to market players of all sizes, but in particular the more successful ones are pursued. That is due to the fact that patents are not very well defined rights and their actual scope can only be determined in a full-fledged litigation case. All software patent searches are unreliable and you cannot find all of them. The costs of patent litigation are prohibitive high for SMEs which usually results in settlements with trolls under less favorable terms.

In the USA but also in Europe there is the feeling of an overshooting of patenting practice which causes great headaches in examination institutions but also on the larger market. What is "protection" to the applicant (without any positive right to carry out his invention) translates into a market entry barrier for his competitor. From a theoretical standpoint a problem of imitation is solved by the protection but that does not seem to be the real concern of innovative applicants. They just want a tangible rights that make it easier to raise venture capital, as a means of signaling to the Venture Capitalists. In Europe we find a more debt based financing system for companies with venture capital being the exception. If an innovator risks his own capital or indebts himself to put his ideas into practice any uncontrollable legal risks in the frontier market are commercial nightmares. Patent protection resembles protection by land mines. If a small player steps on a mine he is finished, he also cannot defend himself with a single land mine. An army, the larger player, may be put in the position to lose some soldiers, mine area (portfolios) and exchange maps (cross-licensing). A small innovator is finished with a single patent infringement. Patents provide an unique opportunity to 'shoot' emerging competitors 'in the cradle'. So basically the critical question is if the innovative company is able to grow fast enough to

[14] Companies increase deliberately their patent pool with the intention of being able to sign bilateral agreements in case of a thread by other company of patents infringements. The most famous recently publicly known examples are the agreements between Microsoft and Novell, and Microsoft and TomTom

survive a patent ambush.

Companies want their investment in software shielded from risks and friction costs. At present the patent system causes itself more risks for the commercial environment than it eliminates. Small and Medium Enterprises, as the backbone of the European economy, takes the largest burden. From the perspective of accountancy we find still no sound economic method to rate the value of "unused" ICT patent assets. Some scholars argue that there is a risk of a patent bubble because too many questionable patents have been granted by patent offices around the world and ICT companies which stockpiled patents have to reevaluate these assets. Only few patents generate a licensing revenue stream. How is the software patent bubble going to burst? The likely scenario is enforcement of formerly passive patents, proliferation of patents to patent trolls and destruction of ICT markets by patent ambushes.

All commercial considerations are of course irrelevant for the granting process. The granting authority consider neither the commercial effects on the applicants nor market. Does the European Patent Convention provide an applicable legal base to grant? That is the question the examiners have to consider. For software and business methods the answer is with a grain of salt: no. We also find that European Patent Office went too far in the recent years. It granted software and business method patents without a sufficient legal base. Politically the commercial opposition to soft patenting is on a rise as more and more litigation cases pop up, and national IPR enforcements rules are strengthened.

We recommend to

- respect the substantive exclusion of software and business method patents
- make the requirement of a technical character for specific by coming up with a clear definition of what is technical or not.
- keep the box of classical patenting in the technical fields it was designed for. A closed box of classical engineering does not preclude legislative action to define specific examination rules for software and business methods. We have seen that with biopatents.
- Patentability of business methods and software needs examination rules that are specifically designed with these subject matter in mind.
- In this respect let us also highlight the requirements under Article 10

> of the TRIPS agreement. TRIPS 10(2) says: Compilations of data or other material, whether in machine readable or other form, which by reason of the selection or arrangement of their contents constitute intellectual creations shall be protected as such. Such protection, which shall not extend to the data or material itself, shall be without prejudice to any copyright subsisting in the data or material itself.

It is apparent from that wording that TRIPS 10 regards copyright, not patents, as the complementary protection instrument to software compilations protected under TRIPS 10. In the WIPO copyright treaty and the EU Software Protection Directive the copyright protection of software is defined. Any definition of software "as such" needs to be put in line with the rights conferred under the EU software directive and the definition of software found there.

Any unnecessary overlap of legal protection systems causes friction and eliminates business opportunities.

QUESTION 1: CAN A COMPUTER PROGRAM ONLY BE EXCLUDED AS A COMPUTER PROGRAM AS SUCH IF IT IS EXPLICITLY CLAIMED AS A COMPUTER PROGRAM

In Question 1 it is asked whether the exclusion would only extent to "literal" claims on "computer programs". We do not support a strict interpretation and endorse the standard interpretive principle "substance over form".

According to EPC Article 52(2) programs for computers are included in a list of items that are not "inventions". So as a matter of principle computer programs are not patentable. Together with other items in the list of exclusion this applies only to the extent that a patent or patent application would relate to such a computer program as such (EPC Article 52(3)). It is subject to divergent interpretations how far the exemption under this provision goes. The actual scope of patentability for computer programs depends on the interpretation of the legal base.

The patent granting practice of the 1980ths was fundamentally different from how the legal base became interpreted these days. The European Patent Convention as the applicable legal base was only reformed once in 2000 but the Boards of Appeal expanded to scope of patentability by case law development. Case law of the Technical boards of Appeal draws a very narrow line concerning the exclusion where a "computer program as such" is postulated which is not patentable. The "computer program as such" is a kind of legal phantom. All software that is in actual use would become patentable subject matter and software "as such" a legal fiction to satisfy the exclusion.

From our perspective a dynamic inconsistency expresses flaws in the political economics of the appeal system. We found the argument that applicants came up the new claim drafting formulations: "In the 1990s, applicants stated to formulate claims for their computer implemented inventions in terms of the computer program, e. g. 'Computer program for carrying out method X, or 'Computer readable medium for storing a computer program for carrying out method X'." We find that claim drafting creativity is merely a means to negotiate the examination rules. Like a tax adviser uses loopholes in the taxation regime, a patent agents uses available loophole in substantive interpretation for his or her clients. In taxation the government attempts to close the loopholes because it pursues its fiscal objectives while in patenting the granting process bows in. In our Opinion the decision T 1173/97 with its permission of computer program product claims (CPPs) circumvents the substantive exclusion as it is found in the law.

We find it disturbing that a political question concerning the extend to which computer programs are eligible to a patent is left to case law of an institution that has an institutional bias: On the one hand a patent office is largely financed by fees for granted patents and has an institutional conflict of interest, on the other hand a patent office does not internalize the social costs of excessive grants. Opposition procedures usually suffer from a "free-rider" problem as economists call it. A party which actually opposes a patent grant is only driven by its singular individual interest which is lower than the social interest (as aggregated individual interest) in an opposition. Therefore the opposition procedures express an inherent bias to grant as a result of an unfulfilled "Samuelson condition". It comes at absolutely no surprise that the unbalance leads to an expansive decision making process of the Boards of Appeal.

If there is an interest to patent software as demonstrated by the applications of computer program products that is a political position, and proponents should seek a democratic majority to make software patentable. The "technical" debate how to claim "software inventions" would then be simplified. However, as there is reasonable doubt about the merits of software patenting and no proper permission, it would be advisable to review the decisions of the Technical boards and realign them with the EPC as the applicable legal base.

QUESTION 2 (A) CAN A CLAIM IN THE AREA OF COMPUTER PROGRAMS AVOID EXCLUSION UNDER ART. 52(2)(c) AND (3) MERELY BY EXPLICITLY MENTIONING THE USE OF A COMPUTER OR A COMPUTER-READABLE DATA STORAGE MEDIUM?

This fundamentally depends on the question if software should be patentable. A computer program

- requires a medium
- requires a computer for its performance, the very purpose of its existence
- otherwise it would be "phantom" we mentioned above.
- According to the principle of "substance over form" and the applicable legal base, the answer is "no".

(B) IF QUESTION 2 (A) IS ANSWERED IN THE NEGATIVE, IS A FURTHER TECHNICAL EFFECT NECESSARY TO AVOID EXCLUSION, SAID EFFECT GOING BEYOND THOSE EFFECTS INHERENT IN THE USE OF A COMPUTER OR DATA STORAGE MEDIUM TO RESPECTIVELY EXECUTE OR STORE A COMPUTER PROGRAM?

- Our opinion under (A) is mirrored in the referral: "The very purpose of a computer program is to be executed by a computer, and to be executed by a computer it must be stored on a computer-readable data storage medium. Even though both the computer and the data storage medium are without doubt technical apparatus, the implicit use of a computer or data storage medium cannot be sufficient to avoid exclusion of computer programs as such. Otherwise the exclusion would be rendered void."
- If the objective is that the exclusion is not rendered void it needs to be closely examined what the effects of "effects going beyond those inherent in the use" are on the substance of the exclusion.
- "A computer program product is not excluded from patentability under

Art. 52(2) and (3) EPC if, when it is run on a computer, it produces a further technical effect which goes beyond the 'normal' physical interactions between program (software) and computer (hardware)".

- However, there is no sufficient legal base for a "further technical effect" requirement in the EPC 52 for the exclusion.

QUESTION 3 (A) MUST A CLAIMED FEATURE CAUSE A TECHNICAL EFFECT ON A PHYSICAL ENTITY IN THE REAL WORLD IN ORDER TO CONTRIBUTE TO THE TECHNICAL CHARACTER OF THE CLAIM?

Yes.

(B) IF (A) IS ANSWERED IN THE POSITIVE, IS IT SUFFICIENT THAT THE PHYSICAL ENTITY BE AN UNSPECIFIED COMPUTER?

We believe that effects of software with a computer are relevant. Generally software does not change the runtime state of the computer but not the computer as such. There are some exemptions as for instance the possibility to destroy the computer with certain software operations.

Operations by the software as for instance the permanent storage of data on a medium as a floppy disc are conventional uses of the medium.

QUESTION 4 (A) DOES THE ACTIVITY OF PROGRAMMING A COMPUTER NECESSARILY INVOLVE TECHNICAL CONSIDERATIONS?

No.

(C) IF (A) lS ANSWERED IN THE NEGATIVE, CAN FEATURES RESULTING FROM PROGRAMMING CONTRIBUTE TO THE TECHNICAL CHARACTER OF A CLAIM ONLY WHEN THEY CONTRIBUTE TO A FURTHER TECHNICAL EFFECT WHEN THE PROGRAM IS EXECUTED?

Relevant is if we find an "invention". Usually what results from programming is a program, and what results from inventing is an invention. Programs are not considered as inventions according to Article 52(2) (applies irrespective of the "patentability" considerations under Article 52(3)). For this reason programming as an activity is not relevant.

Programming can also mean the configuration of a technical apparatus but that is unrelated to software. This may bring clarity into the following question:

It seems important to consider the actual tasks performed by a programmer. Would he be responsible for the design of the technical system and the role that the computer program plays therein, and thus be solving technical problems, or would the design be the task of an engineer who would then pass on his (programming) requirements to the programmer?

In the first situation we have potentially an inventor with an invention to which a software component is an element.

In the second situation we have a special setting in software development where work is distributed and a "software engineer" defines the software solution on a high abstraction level and lets the programmer put it into an operational state. Both are software developers. Typical is this scenario for instance for the software embodiment of business processes in Enterprise Resource Planning systems.

Furthermore, does the answer depend on whether the considerations of a programmer involve any technical details of the particular computer on which the program will run?

Ironically the second scenario does not involve the consideration of details on the "software engineering" level because that is the very purpose of the abstraction. The person who considers more details would then just apply standard methods of its arts which define how you transform these abstract conceptions into another form that is closer to its machine performance. Ironically it demonstrates how software development is fundamentally different from inventing and "software engineering" has a lot in common with architectural works.

As a final note a programmer who writes programs in an assembler language is not closer to the technology than a programmer in a higher programming language. In fact assembler language come closer to the abstract machines mathematicians developed in the 1940ths.

Basically what needs to be considered in software development is the processing, the transformation of data, the solution of data processing problems but that is beyond "technical" considerations in the classical meaning of patent law.

Sincerely,

Ivan F. Villanueva B.

[illegible]