redhat

1801 Varsity Drive Raleigh NC 27606 | 919.754.3700 Fax: 919.754.3701 | www.redhat.com

30 April 2009

Enlarged Board of Appeal
European Patent Office
Erhardstrasse 27
80331 München
Germany

Re: Case G3/08: Referral under Art. 112(1)(b) EPC by the President of the EPO (Patentability of programs for computers) to the Enlarged Board of Appeal.

Dear Sirs,

Red Hat, Inc. ("Red Hat") appreciates the opportunity to present this written statement to the Enlarged Board of Appeal concerning the patentability of programs for computers. This issue is of great concern to Red Hat, which is the leading provider of open source computer programs and related services to enterprise customers. Red Hat is based in the United States, but it has offices in twenty-eight countries, including such European Patent Convention ("EPC") signatories as the Czech Republic, Italy, France, Germany, Spain, Switzerland, and the United Kingdom.

Red Hat respectfully submits that its experience and knowledge of the software industry and of the effects of software patents are relevant to the questions under consideration. This submission therefore addresses the following topics: (1) the nature and significance of computer programs and of open source software, (2) the effects of software patents on innovation, (3) recent changes in the law regarding software patents in the United States, (4) interpretation of the exclusion for programs for computers in Article 52 EPC and finally (5) brief responses to the questions posed in the referral.

I. The Nature and Significance of Computer Programs and of Open Source Software

From an economic and social perspective, software is an unusual good.[1] Unlike most goods, it can be replicated and distributed at close to zero cost, and such distribution does not result in any depletion of the software supply. *See, e.g.*, Steven Weber, *The Success of Open Source* 9 (2004). Producing it requires no substantial capital investment; only a computer is needed. In terms of functionality, software now contributes to an untold number of human activities. Its economic and practical significance is enormous.

It is, therefore, of special importance to consider carefully the legal system governing software. Inconsistent prior decisions in this area and the changing law in various countries, including the United States, make this referral particularly timely. Such consideration should take into account the

1 The referral, and this brief, treat the term computer program as synonymous with the term software. The referral provides the following definition of "computer program": "a series of steps (instructions) which will be carried out by the computer when the program is executed." This definition is consistent with common industry understanding and published definitions.

experience and perspectives of software developers and companies of various types. Red Hat's experience in the software business is, in some respects, similar to most software companies, but in one respect it can offer a unique perspective: that of the leading provider of open source software[2] to enterprises.

Open source software is ubiquitous in developed countries and used daily by millions for such activities as web searching, email, online shopping, and banking. It is found in devices as varied as desktop computers, cellular phones, camcorders, MRI medical devices, automobiles, and jumbo jets. It provides the technological backbone of many large corporations and supports essential functions of many national and regional governments. There are numerous widely used open source software products.[3]

The open source model produces software innovation through a mechanism of collaborative development that fundamentally relies on communication of ideas by large numbers of independent individuals and companies. To understand the meaning of open source, it is helpful to understand how software is made. Software begins as plain text "source code." Programmers write and edit source code in human-readable programming languages that allow specification of software features and behavior at a high level of abstraction. Software is commonly distributed in machine-executable "object code" form, produced by "compiling" the source code of the software. Since object code consists of nearly unintelligible strings of 1s and 0s, software is effectively unmodifiable unless one has access to its source code.

Open source uses a combination of technological and legal means to facilitate collaborative development and commercial exploitation. Typically, an open source package originates as a community-based project that makes its software publicly available in source code form, under licensing terms that grant very broad, royalty-free copyright permissions allowing further use, copying, modification and distribution. The Linux kernel, for example, is licensed as a whole under the GNU General Public License, version 2, the most widely-used open source license.

In making source code available and conferring broad copyright permissions, open source differs significantly from traditional proprietary software. A vendor of proprietary software generally develops the software in-house and provides only object code to the user under severely restrictive licenses that allow no rights to copy, modify, or redistribute that code. Such vendors retain the source code as a trade secret.

The open source development model has proven to be highly effective in producing software of superior quality. Because there are many developers working as collaborators in a distributed fashion, innovation happens rapidly.[4] Because of the many who volunteer their time, and the availability of the

---

2 "Open source software" is also known by the earlier term "free software" and "software libre." "Free" here refers to the freedom to study, modify and share software, rather than availability at no cost; all open source licenses permit and facilitate commercial distribution. The English word "free" has the benefit of emphasizing the ethical foundations of open source but can cause confusion if misunderstood to mean only gratis. Therefore we have used the label "open source software" in this submission.

3 Some well-known examples of commercially important open source programs are the Linux operating system kernel, the Apache web server, the Firefox web browser, the MySQL database management system, and the GCC compiler collection.

4 *See, e.g.*, E. von Hippel, *Democratizing Innovation*, ch. 7 (2005).

source code under royalty-free licenses granting generous modification and distribution rights, the cost of producing and improving software is low. Software bugs and security problems are quickly identified and remedied. Moreover, because users have access to the source code, those users can diagnose problems and customize the software to suit their particular needs.

The growth of open source software has been remarkable and continues to accelerate; the amount of open source code is now doubling every fourteen months. A. Deshpande and Dirk Riehle, The Total Growth of Open Source, in *Proceedings of the Fourth Conference on Open Source Systems,* 197-209 (Springer Verlag, 2008). As explained in the next section, the remarkable development of open source software, and also advances in proprietary software, suggest that certain traditional assumptions regarding the patent system must be reconsidered.

II. The Effects of Software Patents on Innovation

A primary objective of the patent system is to promote innovation. *See* Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS), Part 1, n 1, Art. 7; U.S. Const. Art. I, Sec. 8. The nature of patentable subject matter should be surveyed with this objective in view. It is commonly assumed, without supporting evidence, that patents always promote innovation, but this is not the case. In fact, changes in United States law in the 1990s allowed for proliferation of software patents, but there is evidence that this change not only was unnecessary for software innovation, but also has actually hindered innovation.

The historical record of innovation in software shows that, both for open source and proprietary software, remarkable progress in the U.S. and Europe occurred prior to the time that software patents became generally available. Innovative open source software projects began to appear by the early 1980s. At that time, software patents were relatively few in number and case law was interpreted to limit their availability. *See Diamond v. Diehr*, 450 U.S. 175, 185–86 (1981). By contrast, it was already settled that copyright law covered software. Thus the early innovators of open source software had no reason even to consider obtaining patents on their work, and in fact opposed software patents.

Such opposition is not surprising, because the open, collaborative activity at the heart of the open source model is fundamentally at odds with the patent system. Patents exclude the public from making, using, or selling patented inventions. An open source developer seeks to contribute code to the community -- not to exclude others from using the code. The exclusionary objectives of the patent system are inherently in conflict with the collaborative objectives of open source.

Of course, proprietary software companies do not necessarily share such collaborative objectives, but their experience with respect to software patents is much the same. In the United States, case law allowing software patents dates from the mid-1990s. *See In re Alappat,* 33 F.3d 1526 (Fed. Cir. 1994) (en banc). But major innovations in proprietary software long pre-date the availability of such patents. Such enormously successful proprietary software products as Microsoft Word, Oracle Database, and Lotus 1-2-3 all date from the early 1980s – well before the proliferation of software patents. Competitive forces, rather than patents, spurred development of these products. *See To Promote Innovation,* Report of the U.S. Federal Trade Commission, Chap. 3, Sec. 4, 46 (2003) (hereinafter "*FTC Innovation Report*"). Copyright law provided, and continues to provide, broad legal protection for such products. It is not surprising that many proprietary software developers have opposed software patents. J. Bessen and M. Meurer, *Patent Failure* 189 (2008) (hereinafter "Bessen

and Meurer").

As a result, however, of case law in the mid-1990s, software patent applications and software patents dramatically increased, and at present in the United States there are at least 200,000 issued software patents. *See* Bessen and Meurer at 22. The proliferation of software patents has raised significant new risks for both open source and proprietary software developers.

With respect to software development generally, innovation is generally incremental in nature – that is, new products typically build on products built previously. *See FTC Innovation Report* at 44-45. Innovation is rapid and product cycles are short. Major software products are complex, involving many thousands or even millions of lines of code, and at times hundreds or thousands of distinguishable features that could already be patented. *See FTC Innovation Report* at 52. In these conditions, software patents create particular problems.

Software patents are generally claimed in relatively abstract language, compared with patents concerning other subject matter, and as a result their boundaries typically are vague and ill-defined. *See* Bessen and Meurer at 23, 203. Therefore it is often not possible to be confident that a particular patent does not read on a particular new product. Moreover, there is no reliable, economical method for searching the hundreds of thousands of existing software patents that would allow a developer to rule out the possibility that a new product (which may have hundreds or thousands of separate components and features) infringes one or more patents. Bessen and Meurer at 50, 69-70. Thus simply by virtue of producing and marketing an innovative software product, a software developer assumes an unmeasurable risk of a costly patent infringement lawsuit. *See FTC Innovation Report* at 53-54, 56.

In the U.S., software patents are more than twice as likely to be the subject of a lawsuit than other patents and account for one quarter of all patent lawsuits. Bessen and Meurer at 22, 192. They account for much of the enormous cost of patent litigation. *Id.* The cost of defending a patent lawsuit frequently amounts to several million dollars. Such lawsuits involve technical issues that are difficult for judges and juries to understand, and so even with a strong defense it is usually not possible to be sure of the outcome. If there is a judgment of infringement, the penalty may be an injunction ending further production and enormous monetary damages. Defense costs and litigation risks are so large that in most cases defendants agree to some payment to settle such cases. Even when claims appear to have no valid basis, targets frequently agree to pay for licenses based on the mere threat of litigation.

Some large technology companies have addressed the risk of inadvertent infringement of patents by obtaining as many patents as possible, on the theory that a large patent portfolio will deter other companies from bringing a patent lawsuit, because of the risk of a countersuit.[5] *See FTC Innovation Report at 56.* This approach is often compared to the Cold War military strategy of mutually assured destruction. Companies with such portfolios often enter into cross-licensing agreements with other large companies that have their own patent portfolios in an attempt to obtain a modicum of patent peace. *Id.* at 52.

While such defensive measures are understandable from an individual enterprise's perspective,

5 Red Hat, like some of its competitors, has built a patent portfolio. This portfolio is designed to be used only for the purpose of defending against patent aggression. Red Hat has extended a public Patent Promise under which it pledges not to enforce its patents against parties that infringe those patents through their use of software covered by designed open source licenses.

they are far from optimal. They create a vicious cycle: to defend against a multitude of vague patents, companies obtain still more vague patents. Resources expended on this strategy are, of course, unavailable for research and development or for other more productive purposes. *FTC Innovation Report* at 52. Established companies may be able to bear the cost of the deterrence strategy, but small companies and potential new competitors lack the resources to do so. Thus the system discourages new entry into the market. *See id.* at 51-52.

Moreover, even for companies with the wherewithal to build patent portfolios, the deterrence approach is not always effective. With the proliferation of software patents has come the expansion of a class of business created expressly for the purpose of exploiting the weaknesses in the patent system, which are sometimes referred to either as non-practicing entities or as patent trolls. These entities acquire vague patents at low cost with a view to extorting licensing fees or bringing lawsuits against operating businesses. They frequently conceal their identities and holdings until the companies that are their targets, which have no knowledge of the relevant patents, are locked in to a product and business strategy. Then they demand ransom. Because such entities create nothing and have no productive operations, they are not deterred by the possibility of a countersuit.

In sum, legal changes in the U.S. that have allowed software patents have not served the purpose of promoting innovation. Experience has shown that such patents are not only unnecessary, but actually detrimental. They discourage innovation, by imposing costs and risks on software development that would not otherwise exist. As discussed below, however, the U.S. Court of Appeals for the Federal Circuit recently appeared to recognize that the existing system must be substantially modified, and it has provided an approach for repairing the system.

III. Recent Case Law Has Significantly Changed the Standard Applicable in the United States to Software Patents

As noted in the previous section, beginning in 1994 case law in the U.S. opened the floodgates for software and other abstract patents. In the leading cases of *In re Allapat*, 33 F.3d 1526 (Fed. Cir. 1994), and *State Street Bank & Trust Co. v. Signature Fin. Group*, 149 F.3d 1368 (Fed. Cir. 1998), the Federal Circuit significantly broadened the standards for patentable subject matter. That court has recently, however, revisited the issue of abstract patents and established a new standard that narrows the range of patentable subject matter. *In re Bilski*, 545 F.3d 943 (Fed. Cir. 2008) (en banc) (petition for writ of certiorari pending).

Bilski involved a business method claim (not a software patent claim), but the standard it sets is applicable to software patents. Based on prior case law, the *Bilski* court recognized that "abstract intellectual concepts are not patentable, as they are the basic tools of scientific and technological work." 545 F.3d at 952 (citation omitted). The court thus recognized that allowing patents on an overly broad basis could hinder innovation, and thereby run directly counter to the central purpose of the patent system. With that concern in mind, it articulated the following test for determining when a patent process claim is properly tailored: it must be either be "tied to a particular machine or apparatus" or "transform[] a particular article into a different state or thing." *Id.* at 954. The court also determined that "the recited machine or transformation must constitute more than mere 'insignificant postsolution activity.'" *Id.* at 957.

Although the *Bilski* court noted that it was not deciding the issue of whether computer programs

were in any circumstances patentable, its *dicta* on this issue is significant. The court noted that the U.S. Supreme Court had previously found that merely "tying [a] process to a computer" did not suffice for patentability when the algorithm at issue "had no utility other than operating on a digital computer." 545 F.3d at 955 (discussing *Benson* case). The court noted that "the claim's tie to a digital computer did not reduce the preemptive footprint of the claim since all uses of the algorithm were still covered by the claim." *Id.* This implies that the "tied to a particular machine" branch of the *Bilski* test is not satisfied merely by a claim directed to software that runs on a general purpose computer. A computer program is by definition intended only for use on a computer, and so its "preemptive footprint" is never reduced by limiting the claim to such a use.

The *Bilski* court partially overruled *Alappat* and *State Street* and discarded the standards in those cases that had been applied to allow many patents directed to computer programs. 545 F.3d at 960 n. 19. Thus the system that has been in place since the mid-1990s has been replaced with a new approach. The ultimate effect of that new approach on software patents will be determined by future cases.[6] While it is not certain that future cases will make software patents unavailable, it is at least clear that this possibility exists.

IV. Interpreting Article 52 EPC According to Its Plain Language

The central issue in this referral is the meaning of the computer program exclusion of Article 52 EPC. That primary guide to the meaning of the exclusion is the language used by the drafters. *See* Vienna Convention on the Law of Treaties, Art. 31(1) (1969) ("A treaty shall be interpreted in good faith in accordance with the ordinary meaning to be given to the terms of the treaty in their context and in light of its object and purpose.").[7] Although supplementary evidence may properly be considered, the records relating to drafting of the relevant terms here are unilluminating.[8] Thus the meaning of the exclusion at issue should be determined based on the language of Article 52.

Article 52 EPC (with emphasis added) states:

> (1) **European patents shall be granted for any inventions**, in all fields of technology, provided that they are new, involve an inventive step and are susceptible of industrial application.
>
> (2) **The following in particular shall not be regarded as inventions** within the meaning of paragraph 1:
> (a) discoveries, scientific theories and mathematical methods;
> (b) aesthetic creations;
> (c) schemes, rules and methods of performing mental acts, playing games or doing

6 *Bilski* was recently applied to invalidate a software patent claim in *Cybersource Corp. v. Retail Decisions, Inc.*, No. C 04-03268 MHP, 1009 WL 81 5448 (N.D. Cal. 2009). Decisions of the Board of Patent Appeals and Interferences have also recently applied *Bilski* to reject software patent claims. *See Ex parte Gutta* (BPAI Jan. 15, 2009); *Ex parte Becker* (BPAI Jan. 26, 2009)

7 *See also* European Patent Office, Extended Board of Appeal Decision G 5/83, of 05.12.1984, OJ EPO 1985, 64 (determining that the European Patent Office should apply the Vienna Convention to interpret the European Patent Convention).

8 *See* Aerotel Ltd, 2006 EWCA Civ. 1371, at 11 ("So, one asks, what help can be had from the travaux preparatoires to the EPC? The answer is not a lot." Dr. Justine Pila "shows that the travaux provide no direct assistance to any of the categories we have to consider.")

business, and **programs for computers**;
(d) presentations of information.

(3) **Paragraph 2 shall exclude the patentability of the subject-matter** or activities referred to therein **only** to the extent to which a European patent relates to such subject-matter or activities **as such**.

Thus Article 52, insofar as it speaks to programs for computers, makes clear that (1) patents are granted for "inventions," (2) programs for computers are not inventions, and (3) the exclusion for programs for computers only applies to such programs "as such."

As previously noted, according to the referral a computer program is properly defined as "a series of steps (instructions) which will be carried out by the computer when the program is executed." Application of this practical definition according to the terms of Article 52 would effectively resolve most or all of the uncertainties noted in the referral. That is, once a patent examiner has determined that a claim in a patent application concerns in essence nothing other than "a series of steps (instructions) which will be carried out by the computer when the program is executed," the examiner should conclude that the claim does not involve an invention as the term is used in Article 52, and the claim should be denied.

Certain comments in the referral suggest, however, that this straightforward approach has not been adequately considered. The referral to the Enlarged Board of Appeal states, "The wording of Article 52(3) EPC does not provide any guidance as to when an item mentioned in paragraph 2 is to be regarded as an invention." Referral at 14. While true, this statement seems to suggest that such guidance would be appropriate. This disregards the unambiguous language in paragraph 2 that listed items "shall not be regarded as inventions." Given that items in paragraph 2 are, by definition, not inventions, there is no need for paragraph 3 to address *when* those items are inventions.

The referral also states, "It is established that if the subject-matter of a claim has technical character, then it is not excluded from patentability under Art. 52(2) and (3) EPC. Referral at 7. It is correct that decisions of the Board that have taken this view, but the matter is "established" only in this limited sense. We respectfully submit that this approach should be reconsidered, on the grounds that, depending on the meaning given to the term "technical," it may be inconsistent with the plain language of Article 52.

The Board's prior decisions have attempted to draw a functional distinction between "programs for computers" and "programs for computers as such." Yet there is no sound reasoning supporting this distinction. The leading decision T 1173/97 *IBM* is a fair example of such deficient reasoning. Its entire argument on this point is as follows:

> 5.1. Within the context of the application of the EPC the technical character of an invention is generally accepted as an essential requirement for its patentability.
>
> 5.2. The exclusion from patentability of programs for computers as such (Article 52(2) and (3) EPC) may be construed to mean that such programs are considered to be mere abstract creations, lacking in technical character. The use of the expression "shall not be regarded as inventions" seems to confirm this interpretation.

5.3 This means that programs for computers must be considered as patentable inventions when they have a technical character.

This may be summarized as: (1) inventions have a technical character, (2) computer programs as such are abstract and therefore lack a technical character, and (3) computer programs that have a technical character are patentable inventions. In other words, the Board assumed that "computer programs as such" refers to a special class of computer programs that is different from the computer programs that are patentable. This assumption has no linguistic or logical foundation. And, as explained above, to the extent this approach leads to broader availability of software patents under the EPC, it runs counter to the objective of promoting innovation.

"As such" is a commonly used phrase or idiom. It means "intrinsically considered" or "in itself." Merriam-Webster Online Dictionary. *See also* Shorter Oxford English Dictionary, vol. 1, p. 126 (5th ed. 2002) ("as being what has been named"); Random House Webster's College Dictionary p. 76 (1997) ("as being what is indicated" or "in itself or in themselves"). That is, the expression "as such" is used for emphasis, and for nothing else. The expression is never used to signify a special category within a class. A dog "as such" is just a dog. A tractor "as such" is just a tractor. Likewise, a computer program "as such" is a computer program, plain and simple.

If the drafters of Article 52 had wanted to express the idea that only a portion of the class was to be excluded from a rule, they could have done so in various ways. For example, to explain that ice cream is forbidden with certain exceptions, they might have said, "All ice cream is forbidden, except for strawberry ice cream." Or they might have said, "Ice cream is prohibited, unless it is strawberry." But one would never attempt to express the concept that some ice cream is not part of the general rule of exclusion by saying, "Ice cream is excluded from eating only to the extent it is ice cream as such." This statement could not reasonably be understood to mean that only a particular type of ice cream is impermissible.

Yet decisions of the Board have adopted an interpretation of "as such" that is just as insupportable. Far from simplifying analysis, this approach has led to further complexities. If it is assumed that programs "as such" means not "programs in themselves," but rather "programs that have particular characteristics," it is necessary to identify the distinguishing characteristics. Again, the Board's decision in T 1173/97 *IBM* illustrates this difficulty. There the Board posited that distinguishing between "programs" and "programs as such" depended on whether the program had a "technical character."

Even if there were some basis in the treaty language for creating a special category of patentable computer programs, the "technical" distinction seems ill-chosen. Computers are commonly perceived as being in some sense technical. Indeed, it is not unusual to hear the complaint that a subject is too "technical" whenever it is not already familiar and comfortable. Thus, to create any meaningful distinction between computer programs, it is necessary to make further distinctions between the various possible types of technical characters. In T 1173/97, the Board attempted to do so in discussing production of a "technical effect." But this refinement produced no clear dividing line for future decisions. Subsequent decisions have been no more successful in arriving at such a clear line. This failure argues for a different approach in applying Article 52.

Under the approach here proposed, Article 52 should be read according to its plain language. "As such," as used in Article 52(3), is properly interpreted according to its ordinary meaning of "in and of itself or themselves, separate and apart from other things or processes." Under this approach, a computer program could be part of a larger patentable invention, even though it could not in and of itself be a patentable invention. This possibility is noted in T 0204/93 *AT&T*, where the Board said that "a computer may control, under control of a program, a technical process and, in accordance with the Board's case law, such a technical process may be patentable." The Board in T 0204/93 further noted that "computer programs as such, independent of such an application, are not patentable irrespective of their content, i.e. even if that content happened to be such as to make it useful, when run for controlling a technical process."

This understanding of the significance of "as such" -- that a patentable process may *include* the use of a computer program in one or more of its steps, but may not *be* a computer program -- is similar to the reasoning of the U.S. Supreme Court in *Diamond v. Diehr*, 450 U.S. 175 (1981). In that case, the U.S. Supreme Court considered the patentability of a process for molding synthetic rubber into cured precision products. The invention involved measuring temperatures inside a mold and feeding the measurements into a computer, which used a well known mathematical formula to calculate the correct cure time and which then stopped the curing process. The Court acknowledged that its prior cases had found certain computer-related inventions to be unpatentable when they involved no more than "the programming of a general purpose computer" for "computing a number." *Id.* at 185-86 (citations omitted). In *Diehr,* the Court distinguished these prior cases, on the grounds that, even though the invention included computer operations, the patent was not for those operations, but rather "for a process of curing synthetic rubber." *Id.* at 187. The Court held that "when a claim containing a mathematical formula implements or applies that formula in a structure or process which, when considered as a whole, is performing a function which the patent laws were designed to protect (e.g. transforming or reducing an article to a different state or thing), then the claim satisfies the requirements" for patentable subject matter. *Id.* at 192. *See also In re Bilski, supra.*

In sum, prior decisions of the Board have erred in interpreting Article 52 contrary to the plain language of its drafters. The plain language interpretation proposed here is consistent with the direction of U.S. case law and also with promoting innovation in software development.

V. Answers To Questions in The Referral

Based on the foregoing analysis, we respectfully submit the following answers to the specific questions referred to the Enlarged Board of Appeals.

1. Can a computer program only be excluded as a computer program as such if it is explicitly claimed as a computer program?

   Answer: No. The substance of a claim, rather than the labels used, should be the basis for determining whether a claim involves patentable subject matter.

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?

Answer: No. If the claim properly interpreted is directed in essence to a computer program or components of a computer program, the fact that the claim also makes reference to well-known computer machinery elements (such as a computer itself, or a processor and memory) or to data storage should not affect the analysis.

(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?

Answer: The issue of exclusion should be determined based on whether the claim is, in its essence, a claim for a computer program, and it should be unnecessary to address the issue of a technical effect. To the extent, however, that the concept of "further technical effect" is deemed relevant, it should be carefully defined and limited to cases involving significant physical transformation outside the computer or data storage medium and extending beyond the kinds of post-solution activity that typically accompany the use of computers.

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?

Answer: As stated in response to question 2(b), the issue of exclusion should not be determined based on whether there is a technical effect, but rather based on whether the claim is directed to a computer program. However, if a claim includes a computer program as one of several essential elements but is not merely directed to a computer program and also involves a significant physical transformation, it may be patentable. In making a determination on this issue, it is relevant whether the claim encompasses a technical effect on a physical entity in the real world that is separate from computer hardware that stores or runs the program.

(b) If question 3(a) is answered in the positive, is it sufficient that the physical entity be an unspecified computer?

Answer: No, for the reasons previously explained.

(c) If question 3(a) is answered in the negative, can features contribute to the technical character of the claim if the only effects to which they contribute are independent of any particular hardware that may be used?

Answer: No. As previously noted, determination of whether a claim including a computer program involves patentable subject matter should be made based on whether the claim is, in essence, a claim for a computer program.

4. (a) Does the activity of programming a computer necessarily involve technical considerations?

Answer: To non-programmers, computer programming is often considered a complex activity that is in a broad sense "technical," but this does not at all signify that such activity is patentable.

(b) If question 4(a) is answered in the positive, do all features resulting from programming thus contribute to the technical character of a claim?

Answer: As previously explained, determining whether a computer program is patentable should not be done based on whether there is a technical character.

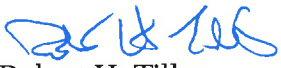
(c) If question 4(a) is answered in the negative, can features resulting from programming contribute to the technical character of a claim only when they contribute to a further technical effect when the program is executed?

Answer: This question is ambiguous in that the definitions of "technical character" and "technical effect" are unclear. This problem highlights the difficulty of basing determinations of patentable subject matter on such terminology. This approach should be modified, and patentability should be determined based on the language of Article 52. Thus, if the claims of a patent application are directed only to a computer program, the application should be denied.

Conclusion

For the reasons set forth above, Article 52 should be read according to its plain language. "As such," as used in Article 52(3), is properly interpreted according to its ordinary meaning of "in and of itself or themselves, separate and apart from other things or processes." Under this approach, a computer program could be part of a larger patentable invention, but it could not in and of itself be a patentable invention. Reading extraneous distinctions concerning the technical character of computer programs into the words "as such" has created a set of intractable interpretive problems. We therefore respectfully request that the Enlarged Board reaffirm the plain language of the computer program exclusion. This approach is consistent both with the language of the EPC and sound policy promoting innovation in software development.

Sincerely,



Robert H. Tiller
Vice President and Assistant General Counsel, IP
Red Hat, Inc.