

No. 08-964

IN THE
Supreme Court of the United States

BERNARD L. BILSKI AND RAND A. WARSAW,
Petitioners,

v.

DAVID J. KAPPOS, UNDER SECRETARY OF COMMERCE
FOR INTELLECTUAL PROPERTY AND DIRECTOR,
PATENT AND TRADEMARK OFFICE,
Respondent.

**On Writ of Certiorari to the
United States Court of Appeals
for the Federal Circuit**

**BRIEF *AMICUS CURIAE* OF RED HAT, INC.
IN SUPPORT OF AFFIRMANCE**

ROBERT H. TILLER
Counsel of Record
RED HAT, INC.
1801 Varsity Drive
Raleigh, NC 27606
(919) 754-4232
Counsel for Amicus Curiae

TABLE OF CONTENTS

	Page
TABLE OF AUTHORITIES.....	ii
STATEMENT OF INTEREST OF <i>AMICUS CURIAE</i> RED HAT, INC.....	1
SUMMARY OF ARGUMENT.....	4
ARGUMENT.....	6
I. THE COURT BELOW CORRECTLY APPLIED THIS COURT'S PRIOR DECISIONS ESTABLISHING THAT ABSTRACT IDEAS ARE NOT PATENTABLE, AND ITS MACHINE-OR-TRANSFORMATION TEST IS CONSISTENT WITH THOSE DECISIONS....	6
II. THE FEDERAL CIRCUIT CORRECTLY ABANDONED A MISINTERPRETATION OF STATUTORY SCOPE THAT HAS CAUSED DRAMATIC HARM TO THE INNOVATION PROCESS IN SOFTWARE.....	9
A. Software Innovation Long Predated Software Patents.....	9
B. The Proliferation of Software Patents Has Resulted in New Risks that Discourage Innovation.....	12
III. AN ABSTRACT IDEA DOES NOT BECOME PATENTABLE MERELY BY IMPLEMENTING IT IN COMPUTER SOFTWARE.....	19
CONCLUSION.....	22

TABLE OF AUTHORITIES

CASES	Page
<i>In re Allapat</i> , 33 F.3d 1526 (Fed. Cir. 1994).....	7, 8
<i>In re Bilski</i> , 545 F.3d 943 (Fed. Cir. 2008)..	8
<i>Dealertrack, Inc. v. Huber</i> , No. 06-2335, 2009 WL 2020761 (C.D. Cal. Jul. 7, 2009).....	20
<i>Diamond v. Diehr</i> , 450 U.S. 175 (1981).....	<i>passim</i>
<i>Gottschalk v. Benson</i> , 409 U.S. 63 (1972)...	<i>passim</i>
<i>Oreilly v. Morse</i> , 56 U.S. 62 (1853)	6
<i>Parker v. Flook</i> , 437 U.S. 584 (1978)	7, 8, 20-21
<i>State Street Bank & Trust Co. v. Signature Fin. Group Inc.</i> , 149 F.3d 1368 (Fed. Cir. 1998), <i>cert. denied</i> , 525 U.S. 1093 (1999).	7, 8
<i>WMS Gaming Inc. v. Int'l Game Tech.</i> , 184 F.3d 1339 (Fed. Cir. 1999).....	20
 BPAI CASES	
<i>Ex parte Cornea-Hasegan</i> , No. 2008-4742 (BPAI Jan. 13, 2009).....	20
<i>Ex parte Daughtrey</i> , No. 2008-0202 (April 8, 2009).....	20
<i>Ex parte Enenkel</i> , No. 2008-2239 (April 6, 2009).....	20
<i>Ex parte Forman</i> , No. 2008-005348 (BPAI Aug. 17, 2009)	20
<i>Ex parte Goud</i> , No. 2008-003121 (BPAI July 20, 2009).....	20
<i>Ex parte Gutta</i> , No. 2008-3000 (BPAI Jan. 15, 2009).....	20
<i>Ex parte Halligan</i> , No. 2008-2823 (BPAI April 8, 2008).....	20
<i>Ex parte Myr</i> , No. 2009-005949 (BPAI Sept. 16, 2009).....	20

TABLE OF AUTHORITIES—Continued

	Page
<i>Ex parte Nawathe</i> , No. 2007-3360 (BPAI Feb. 9, 2009).....	20
CONSTITUTION AND STATUTES	
35 U.S.C. § 101 (2006).....	6, 7, 8, 21
U.S. Const. art. I, § 8.....	11
OTHER AUTHORITIES	
AMERICAN INTELLECTUAL PROPERTY LAW ASSOCIATION, REPORT OF THE ECONOMIC SURVEY (2009).....	16
JAMES BESSEN & MICHAEL J. MEURER, PATENT FAILURE: HOW JUDGES, BUREAU- CRATS, AND LAWYERS PUT INNOVATORS AT RISK (2008).....	12, 13, 14, 15, 16
James Bessen & Robert Hunt, <i>An Empirical Look at Software Patents</i> , 16 J. ECON. & MGMT. STRATEGY 157 (2007)...	12
DAN L. BURK & MARK A. LEMLEY, THE PATENT CRISIS AND HOW THE COURTS CAN SOLVE IT (2009).....	11, 13, 14, 15, 16, 17
COMM. ON INTELLECTUAL PROP. RIGHTS IN THE KNOWLEDGE-BASED ECON., NAT'L RESEARCH COUNCIL, PATENTS IN THE KNOWLEDGE-BASED ECONOMY (Wesley M. Cohen & Stephen A. Merrill eds., 2003).....	12
CLAYTON M. CHRISTENSEN, THE INNOVA- TOR'S DILEMMA (2006)	18
Amit Deshpande & Dirk Riehle, <i>The Total Growth of Open Source</i> , in PROCEEDINGS OF THE FOURTH CONFERENCE ON OPEN SOURCE SYSTEMS, 197-209 (Springer Verlag, 2008).....	2

TABLE OF AUTHORITIES—Continued

	Page
ERIC VON HIPPEL, <i>DEMOCRATIZING INNOVATION</i> (2005)	4
BEN KLEMENS, <i>MATH YOU CAN'T USE – PATENTS, COPYRIGHT, AND SOFTWARE</i> (2006).....	15, 16, 17, 19
Ben Klemens, <i>The Rise of the Information Processing Patent</i> , 14 B.U. J. SCI. & TECH. L. 1 (2008).....	15, 18
Donald E. Knuth, Letter to Commissioner of Patents and Trademarks (Feb. 23, 1994)	13, 15
Mark Lemley, <i>Ignoring Patents</i> , 2008 MICH. ST. L. REV. 19 (Spring 2008)	16
Michael J. Meurer, <i>Controlling Opportunistic and Anti-competitive Intellectual Property Litigation</i> , 44 B.C. L. REV. 509 (2003).....	14, 17, 18
<i>To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy</i> , Report of the U.S. Federal Trade Commission, ch. 3 § V (2003)	<i>passim</i>
Kirk Rowe, <i>Why Pay for What's Free?: Minimizing the Patent Threat to Free and Open Source Software</i> , 7 J. MARSHALL REV. INTELL. PROP. L. 595 (2008)	13
RICHARD M. STALLMAN, <i>FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN</i> (Joshua Gay, ed., 2002)	17
Andrew W. Torrance & Bill Tomlinson, <i>Patents and the Regress of Useful Arts</i> , 10 COLUM. SCI. & TECH. L. REV. 130 (2009).....	12

TABLE OF AUTHORITIES—Continued

	Page
STEVEN WEBER, THE SUCCESS OF OPEN SOURCE (2004).....	10

IN THE
Supreme Court of the United States

No. 08-964

BERNARD L. BILSKI AND RAND A. WARSAW,
Petitioners,

v.

DAVID J. KAPPOS, UNDER SECRETARY OF COMMERCE
FOR INTELLECTUAL PROPERTY AND DIRECTOR,
PATENT AND TRADEMARK OFFICE,
Respondent.

**On Writ of Certiorari to the
United States Court of Appeals
for the Federal Circuit**

**BRIEF *AMICUS CURIAE* OF RED HAT, INC.
IN SUPPORT OF AFFIRMANCE**

**STATEMENT OF INTEREST OF
AMICUS CURIAE RED HAT, INC.**

Red Hat, Inc. is the world's leading provider of open source software and related services to enterprise customers.¹ Its software products are used by

¹ No counsel for a party authored this brief in whole or in part, and no such counsel made a monetary contribution intended to fund the preparation or submission of this brief. No person other than *amicus curiae*, its members, or its counsel made a monetary contribution to its preparation or submission. Petitioners and Respondents have consented to the filing of this brief through blanket letters of consent filed with the Clerk's Office.

Wall Street investment firms, hundreds of Fortune 500 companies, and the United States government. Headquartered in Raleigh, North Carolina, Red Hat has offices in 28 countries.

Red Hat's interest in this proceeding is based on its experience in the software industry and its commitment to the free and open source software community. Open source software is growing at an exponential rate,² and is already of strategic economic importance. It provides the technological backbone of many large corporations and supports essential functions of many national and regional governments. It is used daily by millions of individuals for such activities as web searching, email, on-line shopping, and banking. It is found in devices as varied as mainframe computers, desktop computers, cellular phones, camcorders, medical devices, automobiles, and warships.

The open source model produces software innovation through a mechanism of collaborative development that relies on free communication of ideas among large numbers of independent individuals and companies. To understand open source, it is helpful to understand generally 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. Source code is typically translated by a

² The amount of open source code doubles every fourteen months. Amit Deshpande & Dirk Riehle, *The Total Growth of Open Source*, in PROCEEDINGS OF THE FOURTH CONFERENCE ON OPEN SOURCE SYSTEMS, 197-209 (Springer Verlag, 2008), <http://dirkriehle.com/wp-content/uploads/2008/03/oss-2008-total-growth-final-web.pdf>.

program called a compiler into “object code” form, which basically consists of a series of instructions to be executed on a computer. Since object code consists of unintelligible strings of 1s and 0s, software is effectively unmodifiable without access to its source code. Open source software permits such modification by making the source code available to the user.

Open source software is the product of collaborative development that uses a combination of technological and legal means. Typically, an open source program originates as a community-based project whose members work together using Internet tools such as email, mailing lists, Internet relay chat, bug reporting systems, wikis, and source code version control systems. These tools enable rapid communication among geographically dispersed software developers, and make it possible for large numbers of developers from many different backgrounds and organizations to work collaboratively. A community project 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.

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 subject to 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.³ 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 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 scope of patentable subject matter is an issue of critical importance to the future development of all software, including open source. Because open source innovation depends on sharing source code and free collaboration, open source community members do not generally seek to prohibit or control use of open source software through patents, and most open source software developers view software patents as hindering innovation. Red Hat respectfully submits that this Court should evaluate the issues at bar with a view to the importance of open source software and the bright promise of future open source innovation.

SUMMARY OF ARGUMENT

In the decision below, the Federal Circuit issued a course correction. Beginning in the mid-1990's, that court disregarded the guideposts established by this

³ There are numerous widely used open source software programs, including 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.*, ERIC VON HIPPEL, DEMOCRATIZING INNOVATION 93-106 (2005), available at <http://web.mit.edu/evhippel/www/books.htm>.

Court on the limits of patentable subject matter and issued a series of decisions that opened the floodgates for patents on certain kinds of abstract ideas. As a result, there are now hundreds of thousands of patents on abstract subject matter, and tens of thousands of new patents are now granted each year for software and business methods that were previously excluded from patentable subject matter.

Far from encouraging innovation, this proliferation of patents has seriously encumbered innovation in the software industry. Software is an abstract technology, and translating software functions into patent language generally results in patents with vague and uncertain boundaries. Software products are often highly complex, created by combining hundreds or thousands of discrete (and potentially novel) elements in a cumulative process. Because the boundaries of software patents are exceedingly vague and the numbers of issued software patents is now enormous, it is virtually impossible to rule out the possibility that a new software product may arguably infringe some patent.

Thus, under the Federal Circuit's previous erroneous approach, the risk of going forward with a new software product now always entails an unavoidable risk of a lawsuit that may cost many millions of dollars in legal fees, as well as actual damages, treble damages, and an injunction that terminates a business. Only those with an unusually high tolerance for risk will participate in such a market. The more risk averse, no matter how great their business or technical gifts and innovative potential, are likely to avoid such a market and seek their fortunes elsewhere.

This case offers an opportunity to restore the historical and well-founded boundaries for patentable subject matter that exclude abstract ideas from patent eligibility. It also offers an opportunity to reaffirm the rule, supported both by case law and by sound policy, that computer software is among the types of abstract subject matter that are not patentable under 35 U.S.C. § 101. The machine-or-transformation test set forth in the decision below is fully consistent with this Court's prior case law regarding the patenting of abstract ideas. The Court should adopt this test and make clear that it excludes software from patenting.

ARGUMENT

I. THE COURT BELOW CORRECTLY APPLIED THIS COURT'S PRIOR DECISIONS ESTABLISHING THAT ABSTRACT IDEAS ARE NOT PATENTABLE, AND ITS MACHINE-OR-TRANSFORMATION TEST IS CONSISTENT WITH THOSE DECISIONS

This Court has long recognized that patents have costs as well as benefits. *See Gottschalk v. Benson*, 409 U.S. 63, 68 (1972) (explaining *Oreilly v. Morse*, 56 U.S. 62 (1853)). Patents do not always promote innovation, and they may substantially hinder it. *See id.* A patent on a process excludes others from using that process. If the patent is too broad or vague, it may block or discourage technological progress. Therefore defining the proper subject matter limits of process patents under 35 U.S.C. § 101 is of critical importance. This requires distinguishing between a patentable “process” within the meaning of Section 101, and abstract intellectual concepts.

Thus this Court has determined that “[p]henomena of nature, though just discovered, mental processes, and abstract intellectual concepts are not patentable, as they are the basic tools of scientific and technological work.” *Benson*, 409 U.S. at 67. It runs directly counter to the objective of fostering innovation to allow patents that impede scientific and technological progress. A patent on an algorithm or other abstract idea, as opposed to a specific tangible process, blocks innovation. *Id.* at 68.

To be sure, “[t]he line between a patentable ‘process’ and an unpatentable ‘principle’ is not always clear.” *Parker v. Flook*, 437 U.S. 584, 589 (1978). This Court has repeatedly faced this line-drawing problem in the context of computer-related patents, and has consistently articulated the guideposts to be used. It has affirmed and reaffirmed that “[t]ransformation and reduction of an article ‘to a different state or thing’ is the clue to patentability of a process claim that does not include particular machines.” *Benson*, 409 U.S. at 70. *Accord Diamond v. Diehr*, 450 U.S. 175, 192 (1981); *Flook*, 437 U.S. at 589.

In the mid-1990s, however, the Federal Circuit took an approach at odds with this Court’s teachings in *Flook*, *Benson*, and *Diehr*. 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 Inc.*, 149 F.3d 1368 (Fed. Cir. 1998), *cert. denied*, 525 U.S. 1093 (1999), the Federal Circuit significantly broadened the standards for patentable subject matter. These and subsequent Federal Circuit cases ignored the risks of granting patents on abstract ideas, and instead held that usefulness (“a useful, concrete and tangible result”) was sufficient to satisfy Section 101.

State Street, 149 F.3d at 1373; *Allapat*, 33 F.3d at 1544.

As explained in the next section, this departure from this Court's teachings caused enormous damage to the patent system in general and the software industry in particular. In the decision below, the en banc court of appeals acknowledged that its "useful, concrete and tangible result" test was problematic. *In re Bilski*, 545 F.3d 943, 959-60 (Fed. Cir. 2008). It carefully reexamined this Court's decisions in *Flook*, *Benson*, and *Diehr*, acknowledged the importance of not extending patentable subject matter so far as to impede technological innovation, and articulated a test that is entirely consistent with those decisions.

Using language from *Flook*, *Benson*, and *Diehr*, the Federal Circuit's test distinguishes a patentable process from an abstract idea by considering whether "(1) it is tied to a particular machine or apparatus, or (2) it transforms a particular article into a different state or thing." *Id.* at 954. The appeals court explained that "the use of a specific machine or transformation of an article must impose meaningful limits on the claim's scope to impart patent-eligibility." *Id.* at 961. In addition, the court explained that "the involvement of the machine or transformation in the claimed process must not merely be insignificant extra-solution activity." *Id.* at 962 (citing *Flook*, 437 U.S. at 590).

The decision below thereby corrected the erroneous approach that the Federal Circuit took to Section 101 in the mid-1990s. The Federal Circuit's machine-or-transformation test is in full accord with this Court's prior decisions in *Flook*, *Benson*, and *Diehr*.

II. THE FEDERAL CIRCUIT CORRECTLY ABANDONED A MISINTERPRETATION OF STATUTORY SCOPE THAT HAS CAUSED DRAMATIC HARM TO THE INNOVATION PROCESS IN SOFTWARE

The decision below did not purport to address in categorical terms the patenting of either business methods or software. It is obvious, however, that the machine-or-transformation test (or any other replacement test considered by this Court) will govern attempts to patent these and other abstract ideas. Patenting of software has been particularly controversial, and presents in a clear form the challenge of separating abstract ideas from patentable processes. The creation and expansion of the field of software patents therefore is worth considering both as an example of the larger problem posed by abstract patents and a problem in its own right.

A. Software Innovation Long Predated Software Patents

The importance of the software industry to the United States economy is well recognized. What is less well recognized is that major innovations and economic successes in the software industry occurred prior to the Federal Circuit's decisions in the mid-1990s encouraging software patents. Such enormously successful software products as Microsoft Word, Oracle Database, Lotus 1-2-3, the Unix operating system, and the GNU C compiler all date from the 1980s or earlier—well before the proliferation of software patents. Market forces, rather than patents, spurred development of these products. *See To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy*, Report of the U.S. Federal Trade Commission, ch. 3 § V, at 46 (2003),

available at <http://www2.ftc.gov/os/2003/10/innovationrpt.pdf> (hereinafter “FTC Innovation Report”).⁵

Indeed, in the 1972 *Benson* decision, this Court took note of the exclusion of software from patenting, of problems caused by attempts to patent software, and of the industry’s impressive growth without patents. “Direct attempts to patent [software] programs have been rejected on the ground of non-statutory subject matter.” 409 U.S. at 72 (quoting ‘*To Promote the Progress of . . . Useful Arts,*’ *The President’s Commission on the Patent System* 13 (1966)). “Indirect attempts to obtain patents and avoid the rejection, by drafting claims as a process, or a machine or components thereof programmed in a given manner, rather than as a program itself, have confused the issue further and should not be permitted.” *Id.*

The *Benson* Court, quoting the President’s Commission, also noted the inability of the Patent Office to examine adequately software patent applications. 409 U.S. at 72. At the same time, the Court noted “that the creation of programs has undergone substantial and satisfactory growth in the absence of patent protection and that copyright protection for programs is presently available.” *Id.*

⁵ The profit motive is, of course, an important incentive for software development, but it is not the only one. Open source software developers generally work for open source projects on a voluntary basis. See STEVEN WEBER, *THE SUCCESS OF OPEN SOURCE* 129 (2004). Some of the motivations for their contributions include improving the functioning of a product for business or personal use, enhancing programming skills, reputation, philosophical commitment to free software, and personal enjoyment. *Id.* at 134-36.

Thus the software industry began and reached maturity without the benefit of extensive patent monopolies. This is not to say there was no legal protection for software products. As the *Benson* Court noted, copyright law provided (and it still provides) substantial protection for software products.⁶

This recent history, by itself, calls into serious question whether software patents serve the primary purpose of the patent system of encouraging innovation. See U.S. Const. art. I, § 8. Many of the world's most successful software companies and software products originated and grew strong without incentives from patents. Instead, these successes arose from the dynamics of the competitive market place. *FTC Innovation Report*, ch. 3, § V at 46. That is, prior to the expansion of patentability for software in the mid-1990s, survival in the market place for software depended primarily on the ability to innovate better and more quickly than competitors. Competition, without patent monopolies, resulted in a remarkably dynamic software industry with an impressive record of innovation.⁷

⁶ Copyright protects authors against the copying of their software. Patents, of course, block independent invention of patented technology. Although some patent advocates use the rhetoric of “theft” of ideas to support their arguments, there is evidence that the great majority of patent lawsuits are not against defendants who copied inventions but rather against independent inventors. DAN L. BURK & MARK A. LEMLEY, *THE PATENT CRISIS AND HOW THE COURTS CAN SOLVE IT* 28 (2009) (hereinafter “THE PATENT CRISIS”).

⁷ Although it is frequently assumed that patents encourage technological innovation, there is little empirical evidence supporting this view. In a 2003 report “Patents in the Knowledge-Based Economy,” the National Academies undertook a comprehensive review of the United States patent system, and

B. The Proliferation of Software Patents Has Resulted in New Risks that Discourage Innovation

Since the mid-1990s, there is one respect in which software patents have been successful as a species: they have proliferated. At present in the United States there are at least 200,000 issued software patents. See JAMES BESSEN & MICHAEL J. MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS, AND LAWYERS PUT INNOVATORS AT RISK 22 (2008) (hereinafter “PATENT FAILURE”), available in part at <http://researchoninnovation.org/dopatentswork/>. They continue to increase at the rate of approximately 20,000 per year. See James Bessen & Robert Hunt, *An Empirical Look at Software Patents*, 16 J. ECON. & MGMT. STRATEGY 157, 158 (2007). This proliferation has raised significant risks for software developers.

For years, far-sighted industry leaders, scholars, and software developers have warned of these risks and opposed software patents. See PATENT FAILURE at 189. These include Bill Gates, co-founder of Microsoft. In 1991, Mr. Gates stated, “If people had understood how patents would be granted when most

concluded that “[t]here are theoretical as well as empirical reasons to question whether patent rights advance innovation in a substantial way in most industries.” COMM. ON INTELLECTUAL PROP. RIGHTS IN THE KNOWLEDGE-BASED ECON., NAT’L RESEARCH COUNCIL, PATENTS IN THE KNOWLEDGE-BASED ECONOMY 2 (Wesley M. Cohen & Stephen A. Merrill eds., 2003), available at http://www.nap.edu/catalog.php?record_id=10770. Scholarly studies have called into question the basic assumption that patent protection in general spurs innovation. Andrew W. Torrance & Bill Tomlinson, *Patents and the Regress of Useful Arts*, 10 COLUM. SCI. & TECH. L. REV. 130, 133-34 (2009).

of today's ideas were invented and had taken out patents, the industry would be at a complete standstill today." Kirk Rowe, *Why Pay for What's Free?: Minimizing the Patent Threat to Free and Open Source Software*, 7 J. MARSHALL REV. INTELL. PROP. L. 595, 595 (2008).

Similar, Donald E. Knuth, Professor Emeritus at Stanford University and one of the world's most respected computer scientists, wrote in 1994, "When I think of the computer programs I require daily to get my own work done, I cannot help but realize that none of them would exist today if software patents had been prevalent in the 1960s and 1970s." Donald E. Knuth, Letter to Commissioner of Patents and Trademarks 2 (Feb. 23, 1994), available at [http://documents.epo.org/projects/babylon/eponet.nsf/0/5294C4422611FE7BC12575B6006414D2/\\$File/G3-08_amicus_curiae_brief_Knuth_en.pdf](http://documents.epo.org/projects/babylon/eponet.nsf/0/5294C4422611FE7BC12575B6006414D2/$File/G3-08_amicus_curiae_brief_Knuth_en.pdf). Dr. Knuth also stated, "I strongly believe that the recent trend to patenting algorithms is of benefit only to a very small number of attorneys and inventors, while it is seriously harmful to the vast majority of people who want to do useful things with computers." *Id.*

The views of Mr. Gates and Dr. Knuth were shared by many firms and developers in the 1990s. PATENT FAILURE at 189. The risk that they articulated—that patents tend to hinder software innovation—relates to at least two different aspects of software: the incremental nature of software development and the near impossibility of establishing clear boundaries for software patents.

In general, software innovation is cumulative in nature—that is, new products typically build on products built previously. *See FTC Innovation Report*, ch. 3, § V at 44-45; THE PATENT CRISIS at 47. Innovation

is rapid and product cycles are short.⁸ Major software products are complex, involving many thousands or even millions of lines of code and many different components. Components are normally developed using many earlier-developed sub-components. Some software products contain thousands of distinguishable components, any number of which could (in view of erroneous patenting practices) already be patented. *See FTC Innovation Report*, ch. 3 § V at 52; *THE PATENT CRISIS* at 53-54.

It is, however, practically impossible to know with reasonable certainty whether a new software product could be said to infringe some prior software patent. Patents are conventionally referred to as intellectual property. However, as James Bessen and Michael Meurer have explained in detail, patents differ substantially from tangible property in that their boundaries are often fuzzy and unpredictable. *PATENT FAILURE* at 46-72. If patents do not give clear notice of their limits, they create a risk of inadvertent infringement. Vague patents also enable opportunistic behavior. For example, a patentee may, based on vague language, claim ownership of a technology unknown to the inventor, but instead first conceived by someone else. *Id.* at 199.⁹

⁸ Because of rapid product cycles, it is difficult for a software inventor to use a patent productively to enforce legitimate rights. A patent lawsuit is likely to take longer to resolve than a product cycle, and may even take several product cycles. *See THE PATENT CRISIS* at 57.

⁹ For example, a plaintiff may argue that a pre-internet patent covers some use of internet technology. Michael J. Meurer, *Controlling Opportunistic and Anti-competitive Intellectual Property Litigation*, 44 B.C. L. REV. 509, 542 (2003).

This problem of uncertain patent boundaries is particularly acute with software patents. Software is an abstract technology.¹⁰ Software algorithms can be represented in numerous different ways, and even computer scientists sometimes disagree over whether two software technologies are equivalent. See PATENT FAILURE at 22, 203. Thus it is not surprising that software patents are typically framed in abstract language with uncertain boundaries. See PATENT FAILURE at 23, 203; THE PATENT CRISIS at 27, 58. As a result, a software developer, when shown a software patent, often cannot be sure whether the patent reads on newly developed code.

This difficulty is multiplied hundreds or thousands of times with regard to a complex software product combining hundreds or thousands of discrete components. A separate but related problem faces all software developers—that of the impossibility of patent clearance, or determining whether there are existing patents that may be said to read on a new product. There is no reliable, economical method for

¹⁰ Computer software is abstract because it is, in essence, nothing more than a set of mathematical algorithms, expressed in a particular programming or machine language. An algorithm is a mathematical construct, consisting of a series of steps for solving a problem. See BEN KLEMENS, MATH YOU CAN'T USE – PATENTS, COPYRIGHT, AND SOFTWARE 48-51 (2006) (hereinafter “MATH YOU CAN'T USE”). Computer scientists view software as consisting entirely of algorithms. See Ben Klemens, *The Rise of the Information Processing Patent*, 14 B.U. J. SCI. & TECH. L. 1, 9-11 (2008). As Donald Knuth has explained, “[It is not] possible to distinguish between ‘numerical’ and ‘nonnumerical’ algorithms, as if numbers were somehow different from other kinds of precise information.” See Letter to Commissioner of Patents and Trademarks at 1. This Court has held that algorithms are not patentable. *Benson*, 409 U.S. at 72.

searching the hundreds of thousands of existing software patents.¹¹ PATENT FAILURE at 50, 69-70. *See also* MATH YOU CAN'T USE at 79-80. The clearance problem is made even worse by the existence of tens of thousands of applications that for eighteen months after filing are unpublished.

Thus, simply by virtue of producing and marketing an innovative software product, a software developer assumes the risk of a costly patent infringement lawsuit.¹² *See FTC Innovation Report*, ch. 3. § V 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. PATENT FAILURE at 22, 192. The cost of defending a patent lawsuit frequently amounts to several million dollars. AMERICAN INTELLECTUAL PROPERTY LAW ASSOCIATION, REPORT OF THE ECONOMIC SURVEY I-128-29 (2009). Such lawsuits involve technical issues that are difficult for judges and juries to understand, and so even with a strong defense the outcome is usually far from certain. If

¹¹ The unreliability of indexing for software patents also means that software patents are of little use in advancing innovation by disclosing new technology. From a software developer's point of view, it is completely impractical to seek new ideas in patents, and few if any do so. Most avoid reading patents, for fear that a chance encounter may increase the risk that they will one day be accused of willful infringement. *See THE PATENT CRISIS* at 32; Mark Lemley, *Ignoring Patents*, 2008 MICH. ST. L. REV. 19, 21 (Spring 2008).

¹² A further indication of the ineffectiveness of the patent system for software is that there is little patent licensing prior to development and distribution of products. *See THE PATENT CRISIS* at 59. Because of vague patent boundaries and unreliable search methods, it is not possible to determine all possible rights at issue and strike bargains as to those rights.

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. Michael J. Meurer, *Controlling Opportunistic and Anti-competitive Intellectual Property Litigation*, 44 B.C. L. REV. at 542.

Some large technology companies have addressed the risk of inadvertent infringement of patents by seeking as many patents as possible, on the theory that a large patent portfolio signals the possibility of a countersuit and thus will deter other companies from bringing a patent lawsuit.¹³ See *FTC Innovation Report*, ch. 3 § V at 56; THE PATENT CRISIS at 55. 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. See RICHARD M. STALLMAN, FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN 101-103 (Joshua Gay, ed., 2002); MATH YOU CAN'T USE at 83-85.

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

¹³ 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 designated open source licenses. See https://www.redhat.com/legal/patent_policy.html.

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*, ch. 3 § V at 52. Moreover, although established companies may be able to bear the cost of this deterrence strategy, small companies and potential new competitors generally lack the resources to do so. Thus the system discourages new entry into the market, and thereby hinders innovation. *See id.* at 51-52. *See also* CLAYTON M. CHRISTENSEN, *THE INNOVATOR'S DILEMMA* 26, 52 (2006) (showing that small firms generally lead in new technologies).

Moreover, even for companies with the financial resources 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 businesses created expressly for the purpose of exploiting vague patents. *See* Ben Klemens, *The Rise of the Information Processing Patent*, 14 B.U. J. SCI. & TECH. L. at 27-31; *Meurer, supra*. These are sometimes referred to as non-practicing entities or, less politely, as patent trolls. These entities acquire vague patents at low cost with a view to threatening 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 produce no products, they are not deterred by the possibility of a countersuit.

In sum, all software companies, developers, and users face substantial risks from software patents. These risks include whether an unknown patent may cover newly written code or some other preexisting code in a complex product, whether such a patent could be the basis of a lawsuit, whether a relevant patent is in the hands of an aggressive party dedicated to bringing patent lawsuits, and whether a trial may result in an injunction or damages award.

These risks have obviously not brought the software industry to a standstill. Established companies are protected to some extent by their patent portfolios and war chests. But for them and even more for new players, software innovation, like sky diving, requires a high tolerance for risk. See *MATH YOU CAN'T USE* at 91. Developers without a high risk tolerance are likely to find the threat of ruinous lawsuits to be discouraging, and to use their talents and energy in less hazardous endeavors. Thus software patents discourage new entries into the marketplace and new software innovation.

III. AN ABSTRACT IDEA DOES NOT BECOME PATENTABLE MERELY BY IMPLEMENTING IT IN COMPUTER SOFTWARE

In connection with addressing the test for excluding abstract ideas from patenting, this Court it should also clarify the application of that test in the context of computer programs that run on general purpose computers. Lower court case law and commentary since the Federal Circuit's *en banc* decision shows that this issue is an important one on which this Court's guidance is needed.

The basic question is whether an otherwise unpatentable idea becomes “tied to a particular machine” when it is implemented in software for execution on a general purpose computer. Prior to the decision below, the Federal Circuit gave credence to the idea that a general purpose computer could be transformed into a particular machine by executing software. *WMS Gaming Inc. v. Int’l Game Tech.*, 184 F.3d 1339, 1348 (Fed. Cir. 1999). On the other hand, this view was recently rejected in *Dealertrack, Inc. v. Huber*, No. 06-2335, 2009 WL 2020761, at *4 (C.D. Cal. Jul. 7, 2009). Moreover, it has been recently repeatedly rejected by the Board of Patent Appeals and Interferences. See, e.g. *Ex parte Myr*, No. 2009-005949 (BPAI Sept. 16, 2009); *Ex parte Forman*, No. 2008-005348 (BPAI Aug. 17, 2009); *Ex parte Goud*, No. 2008-003121 (BPAI July 20, 2009); *Ex parte Daughtrey*, No. 2008-0202 (April 8, 2009); *Ex parte Halligan*, No. 2008-2823 (BPAI April 8, 2008); *Ex parte Enenkel*, No. 2008-2239 (April 6, 2009); *Ex parte Nawathe*, No. 2007-3360 (BPAI Feb. 9, 2009); *Ex parte Gutta*, No. 2008-3000 (BPAI Jan. 15, 2009); *Ex parte Cornea-Hasegan*, No. 2008-4742 (BPAI Jan. 13, 2009). This Court’s decisions in *Benson* and *Diehr* signal that the mere fact that otherwise unpatentable software is executable on a general purpose computer should not convert such software into a patentable invention.

This is not to say that software may not be part of a patentable process or machine. As *Diehr* recognized, an algorithm that is plainly unpatentable by itself may be a part of a process that is patentable when it involves a physical transformation of the sort that has traditionally been considered patentable, such as an industrial process for curing rubber. 450 U.S. at 184-88. See also *Parker v. Flook*, 437 U.S. at

589-94. The Court has also recognized that an otherwise abstract idea may be patentable when “tied to a particular apparatus.” *Flook*, 437 U.S. at 588, n.9.

In *Benson*, the patent application covered “a method of programming a general-purpose digital computer to convert signals from binary-coded decimal to pure binary form.” 409 U.S. at 65. The Court found that the procedure at issue was a mathematical algorithm, and amounted to an unpatentable abstract idea. *Id.* at 65-66. It is important to note, however, that the algorithm had “no substantial practical application except in connection with a digital computer.” *Id.* at 71. The claims were intended to “cover any use of the claimed method in a general-purpose digital computer of any type.” *Id.* at 64.

Thus the algorithm claimed in *Benson* could have been viewed as “tied to a machine,” inasmuch as it was functionally tied to a digital computer. Nevertheless, this Court held that the claims were abstract ideas outside the scope of Section 101. Thus *Benson* precludes an interpretation of Section 101 that views abstract ideas as patentable based on their implementation in software running on a general purpose computer.

The more recent decision in *Diehr* is consistent with this understanding. The rubber curing process in *Diehr* involved an algorithm and a computer, but this Court’s analysis of subject matter turned on the claim as a whole, which concerned the physical transformation of the rubber—not the implementation of the algorithm in a computer program. 450 U.S. at 184-85. *Diehr* explained that use of the computer did not render the process unpatentable,

but the decision makes clear, by its focus on the transformation of rubber, that use of the computer alone does not suffice to make the process patentable. *See id.* at 187.

The *Diehr* Court cautioned against allowing the prohibition on patenting of abstract formulas to “be circumvented by attempting to limit the use of the formula to a particular technological environment.” 450 U.S. at 191. “To hold otherwise would allow a competent draftsman to evade the recognized limitations on the type of subject matter eligible for patent protection.” *Id.* at 192. In view of this caution, this Court should make clear that the test for patentable subject matter cannot be satisfied by the mere drafting device of adding a general purpose computer as an element in a claim that is otherwise directed to an unpatentable algorithm.

CONCLUSION

The judgment of the Federal Circuit should be affirmed.

Respectfully submitted.

ROBERT H. TILLER
Counsel of Record
RED HAT, INC.
1801 Varsity Drive
Raleigh, NC 27606
(919) 754-4232

Counsel for Amicus Curiae