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 OF AMICUS CURIÆ FOUNDATION FOR A FREE INFORMATION INFRASTRUCTURE IN SUPPORT OF DEFENDANT

FFII e. V.
Blutenburgstr. 17
80636 München
Germany (Europe)

COUNSEL OF RECORD
Allonn E. Levy
Hopkins & Carley
408.286.9800 (main)
70 South First Street
San Jose, CA 95113

Questions Presented

1. Whether the Federal Circuit erred by holding that a "process" must be tied to a particular machine or apparatus, or transform a particular article into a different state or thing (Machine-or-Transformation test), to be eligible for patenting under 35 U. S. C. § 101, despite this Court's precedent declining to limit the broad statutory grant of patent eligibility for "any" new and useful process beyond excluding patents for "laws of nature, physical phenomena, and abstract ideas".

2. Whether the Federal Circuit's Machine-or-Transformation test for patent eligibility, which effectively forecloses meaningful patent protection to many business methods, contradicts the clear Congressional intent that patents protect "method[s] of doing or conducting business." 35 U. S. C. § 273.

Contents

Table of Authorities

1 Statement of Interest of Amicus Curiæ[1]

1.1 About the FFII

The Foundation for a Free Information Infrastructure (FFII) e. V. is a charitable association registered in Munich which is dedicated to the spread of data processing literacy. It funds the development of public information works based on copyright, free competition and open standards.

The FFII attained broad international recognition for its phrontistery role in the European debate on a software patent directive (2002–2005) and software-related patent reform. It is a registered observer at the World Intellectual Property Organisation.

The association and its members aim to reduce friction costs and risks for software authors and to prevent dilution of property rights under the Berne Convention by territorial patent grants. To this end FFII is guided by Hayek's dictum that "it is necessary [. . .] not to apply a ready-made formula but to go back to the rationale of the market system and to decide for each class what the precise rights are to be which the government ought to protect. This is a task at least as much for economists as for lawyers."[2]

For digital markets, cheap, fast and narrow rights are sought. The interests of the association are improvements of the substantive patent rules and the examination process. A global challenge is to keep patent office bureaucracies manageable, and make them adapt to an acceleration of markets.

[1]Pursuant to Sup. Ct. R. 37.6, amicus notes that no counsel for a party authored this brief in whole or in part, and no counsel or party 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 consent letters filed with the Clerk's Office.

[2]Hayek, F. A. v., Individualism and Economic Order, Chicago, University of Chicago Press, 1949, 114

1.2 About the Authors

LAURA CREIGHTON is an entrepreneur and software developer from Canada, now living and working in Göteborg, Sweden.

DR. RER. NAT. PETER GERWINSKI is an entrepreneur, software developer and physicist from Essen, Germany.

GEORG JAKOB is a lawyer, business consultant and former software developer from Salzburg, Austria, now living in Munich, Germany.

DIPL. KFM. & M. A. ANDRÉ REBENTISCH is a business intelligence and technology research specialist, currently residing in Wilhelmshaven, Germany.

1.3 The European View

How does the Bilski case affect European economy? In a globalized world, business activities crossing the Atlantic are standard, even for small enterprises. This is even more so in the software market where software can be transferred between continents at no noticeable cost via the Internet. Consequently, even a one-man software company serves customers around the globe and is affected by territorial patents.

But there is a more direct interlink. The European Patent Office (EPO) is currently revising their strategy with respect to patents on software and business methods. It can be expected that the outcome of the Bilski case will affect their decision about the application of European patent law.

The Bilski case touches the patentability of business methods. This affects virtually everyone doing business in the United States. Even the work of the Supreme Court itself is affected. The Supreme Court judges are tasked to decide on constitutional grounds on the technocratic process of patent examination in the United States, a business process, and thus the judges will become "business engineers" themselves. To what extent would a process such as patent examination be patentable? The debate cannot be left to bilateral legal proceedings in a specialized court but ought to be reconciled with the broader legal perspective of purpose. Insofar specialised case law provides a limited legal perspective on patent law.

In Europe it is equally difficult to define the scope of terms like "invention", and apply examination tests for the separation of wheat and chaff. It is not within our special field of expertise to explain U. S. case law. Instead we would like to share with you the experience with examination tests we gained during the last years in Europe. The amici hope that it will help the court to tweak the incentive machinery of patent law for the economic benefit of society.

2 Summary of Argument

The Machine-or-Transformation test instituted by the CAFC *In Re Bilski* is aimed to establish a clear limit to patentable subject-matter in accordance with case law, which consistently rejects the patentability of *abstract ideas*.

The properties of software and business methods are fundamentally different from those of machines and transformations. Economic evidence fails to provide a strong justification for a market intervention here, but indicates that certain patent claims on software algorithms and business methods and other abstract matter stifle the progress of the sciences and useful arts.

Experience with the European technicity test demonstrates that it still allows to obtain patent claims on abstract ideas by combining, in one claim, a newly discovered fundamental principle with a well-known *machine or transformation*.

To address this, it is necessary to apply the test to *the claimed object* rather than to the patent claim *as a whole*:

1. An object that consists only of laws of nature, physical phenomena, and abstract ideas cannot be claimed in a patent, independent of the form in which it is claimed.

2. An object can only be claimed in a patent if it constitutes a new concrete realization of a machine or transformation.

This would limit patent claims on software and business method to concrete realizations instead of all uses of the fundamental principles.

3 Argument

3.1 Patenting at the Boundaries: Software and Business Methods

3.1.1 Economic Aspects

Patents—like any other "exclusive right to [. . .] writings and discoveries"—are supposed to "promote the progress of science and useful arts".[3] They are the cheese in a mouse trap. The economic rationale of the patent system has lead to the false conclusion that *all* patents would further innovation. Empirical analysts measure the innovative power of entire economies by the amount of patents issued per year. "The more patents, the better" is believed by some.

But the total value of patents equals their social costs on a competitive market. There is no "free lunch" in patenting.

Until 1623, before the Statute of Monopolies was enacted in the U. K., patents were granted for almost any business activity, from the right to sell textiles to the import of spices or the export of goods. Patents had become a way for the government to over-regulate and thereby interfere with the market in ways the patent system simply wasn't designed for. About four hundred years ago, the negative effects became so unbearable that patents were limited from business activity to engineering inventions.

Businessmen are protected by antitrust regulations and law against unfair competition. Writers—also those of software—are protected by copyright. Only inventors need patent law, and leading economist have often taken a critical stance regarding the patent system.[4]

From an economic standpoint it is essential that the subject matter that is awarded protection is scarce under laissez-faire conditions, and that more of it gets produced in a protection scenario. When patents are awarded for abundant objects, observers would complain about "trivial" grants. For an economist who engineers an incentive system its application to abundant matter does not make any sense, but limits the freedom of commercial action and stifles the progress of science and

[3]Article 1, Section 8 U. S. Constitution, § 8, cl. 8

[4]Hayek, F. A. v., Individualism and Economic Order, Chicago, University of Chicago Press, 1949, 114

useful arts.

The Machine-or-Transformation test is one approach for a criterion to separate the “good” from the “bad” patents. In section 3.2 we will compare it to the European approach of “technicity”, and suggest an improved patentability test based on the exclusion of laws of nature, physical phenomena, and abstract ideas.

3.1.2 Demands for Reform

In October 2003, The Federal Trade Commission issued a report entitled: To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy. The Executive summary states: “A failure to strike the appropriate balance between competition and patent law and policy can harm innovation. For example, if patent law were to allow patents on ‘obvious’ inventions, that might have developed based on the obvious technology. [. . .] Conversely, competition policy can undermine the innovation that the patent system promotes if overzealous antitrust enforcement restricts the procompetitive use of a valid patent.”[5]

With that in mind, the Department of Justice and the Federal Trade Commission decided to examine the current balance between competition and patent law. They held a series of hearings from February through November 2002 with representatives of all kind of high-tech industries.

While the participants “found much to praise in the current patent system”,[6] their findings indicated that “Some Modifications Are Needed to Maintain a Proper Balance of Competition and Patent Law and Policy”[7] Indeed, the summary transcripts contain a depressing, unrelenting tale of woe. In testimony after testimony, concerns are raised that the USPTO has been granting patents of questionable value. Some granted patents are considered too broad, while some are on trivialities. Still others are mistakenly granted on subject matter which is nonpatentable in the first place. For many, disqualifying prior art exists, but the USPTO was clearly unaware of it at the time they issued a patent.

[5]FTC Report, executive summary, page 3

[6]FTC Report, executive summary, page 4

[7]FTC Report, executive summary, page 4, first conclusion

Some companies are accused of rushing to the patent office attempting to get a patent on whole industries, by virtue of being the first to claim an industry standard practice. Others are claiming mathematical algorithms which have always been the common heritage of all humanity. Indeed, it is not difficult to find patents that manage to suffer from all of these defects simultaneously. Other actors, sometimes characterized as "patent trolls", never innovate at all. Instead, they wait until others independently develop whatever technology they have patented, and then collect fees from the true innovators, who are trying to bring new products onto the market. It is no wonder that industry participants referred to them as "parasites". While such accusations have been in existence for as long as the patent system itself,[8] there is widespread belief that the impact of non-practicing entities ("patent trolls") is much greater today than it has ever been before.

3.1.3 The Fundamental Difference Between Hardware and Software

Over the last decades, computer software has become pervasive for global economy. Although the development of mechanical parts has been optimized over thousands of years, more and more mechanical parts of modern devices are replaced by software. Why is this so?

Lossless Copying. The manufacturing of software is fundamentally different from that of mechanical devices. One important difference is that software can be copied *without loss* at no significant expense. In contrast, mechanical devices must be built one-by-one using expensive equipment.

Ideal Parts. Another reason why the industry is replacing mechanical solutions by software is that they are much easier to create. Typically, the software which replaces even the most

[8]In 1895, George Baldwin Selden obtained a patent with a claim so broad that it literally encompassed most automobiles ever made—putting a gasoline engine on a carriage—despite the fact that he had never gone into production with a working model of an automobile. See Robert Patrick Merges and John Fitzgerald Duffy, Patent Law and Policy, Cases and Materials, Charlottesville (2002), pp 644–646 for more information about the Selden Patent.

sophisticated mechanical device can be written in a few minutes. Why is this so?

When a mechanical device is built, one has to arrange with the laws of physics. No component is 100 % reliable. All lengths, diameters, etc. come with tolerances.

Software also consists of components, the so-called *algorithms*, but they are idealized mathematical objects. They have no tolerances or abrasion. A mechanical device with 100 gears needs a redesign when a 101st gear needs to be added. Its software replacement can be extended to 10,000 “virtual gears” with no noticeable effort.

These ideal parts are very flexible and can be used in very different ways. Many of them can be used in fundamentally different types of software. For example, the same compression algorithm can be used in a drawing program, a cryptography system, a web server, a device driver, and much more.

In software development, it is commonplace to combine large parts—which in turn consist of smaller and smaller parts—to form a big system. For instance, you can create a minimalistic, but useable content management system by just putting together a file server and a web server. With hardware, it is much more difficult to combine the features of two complete systems. For example, you cannot simply weld together a ship and an aircraft and expect the result to swim and to fly.

Accordingly, elaborate software systems contain much more components than mechanical devices constructed within the same number of man-hours.[9]

What happens if one grants exclusive rights on software components? Each of them would cover all uses of that component in all fields of programming. Now, since a software system consists of many thousand components, there is a high risk that it violates many of those exclusive rights.

In view of this, it would make sense not to grant exclusive rights on software components, but only on individual programs, i. e. realizations of software systems. This is what copyright does. However, as will be explained in section 3.1.4, the existing software patents have been granted not on individual programs, but on software components or combinations thereof. This has drastic economic implications for software businesses which will be discussed in section 3.1.5.

[9] See also http://www.gnu.org/philosophy/software-patents.html.

Reverse Engineering Does Not Pay Off. Once an innovative automobile has been built, a competitor can save R&D expenses by purchasing one automobile and disassembling it to see how it works. As will be shown next, the same is *not* possible with software.

As an example, consider a software which calculates the surface A of a sphere with known radius r using the well-known formula $A = 4\pi r^2$.

In order to create such a piece of software, the author writes the so-called *source code* in a programming language. In the programming language "C" (ISO/IEC 9899:1999) this software can be written as follows:

```
/* sphere.c - Calculate the Surface of a Sphere */

#include <stdio.h>
#define pi 3.14159265

int main (void)
{
  float radius, surface;
  scanf ("%f", &radius);  /* read the radius */
  surface = 4.0 * pi * radius * radius;
  printf ("%f\n", surface);  /* show result */
  return 0;  /* report success */
}
```

The source code cannot directly be executed by a computer. To make it executable, it is necessary to transform—to *compile*—it into binary code.

The binary code is a sequence of numbers which can be executed by a computer but which is hardly readable for humans. The following is an excerpt from the binary code of the "sphere" example software above.

```
        ... 8d 4c 24 04  83 e4 f0 ff 71 fc 55 89
e5 51 83 ec 24 8d 45 f8  89 44 24 04 c7 04 24 00
00 00 00 e8 fc ff ff ff  d9 45 f8 d9 c0 dc 0d 00
00 00 00 de c9 d9 5d e8  d9 45 e8 dd 5c 24 04 c7
04 24 03 00 00 00 e8 fc  ff ff ff b8 00 00 00 00
83 c4 24 59 5d 8d 61 fc  c3 ...
```

A software developer who does not want to disclose how the software works and/or wants to retain the privilege to do any modifications on it, keeps the source code as a trade secret. In that case, the binary code above is all what customers get. It is all competitors can get their hands on.

There is in fact a process called “disassembling the software” which means to extract information out of the binary code.

```
...
8D4C2404          lea ecx,[esp+0x4]
83E4F0            and esp,byte -0x10
FF71FC            push dword [ecx-0x4]
55                push ebp
89E5              mov ebp,esp
51                push ecx
83EC24            sub esp,byte +0x24
8D45F8            lea eax,[ebp-0x8]
89442404          mov [esp+0x4],eax
C7042400000000    mov dword [esp],0x0
E8FCFFFFFF        call dword 0x54
D945F8            fld dword [ebp-0x8]
D9C0              fld st0
DC0D00000000      fmul qword [dword 0x0]
DEC9              fmulp st1
D95DE8            fstp dword [ebp-0x18]
D945E8            fld dword [ebp-0x18]
DD5C2404          fstp qword [esp+0x4]
C7042403000000    mov dword [esp],0x3
E8FCFFFFFF        call dword 0x77
B800000000        mov eax,0x0
83C424            add esp,byte +0x24
59                pop ecx
5D                pop ebp
8D61FC            lea esp,[ecx-0x4]
C3                ret
...
```

The above is what can be directly obtained by *disassembling* the binary code of our example program. To what extent can it be used to reverse-engineer the software? To illustrate this, we will translate the above back to the “C” language.

```
int main (void)
{
  float x1, x2;
  f1 ("%f", &x1);
  x2 = 12.5663706 * x1 * x1;
  f2 ("%f\n", x2);
  return 0;
}
```

In this “reconstructed source code” some important pieces of information are irretrievably lost,

- all comments (`/* ... */` in “C”),

- the names “radius”, “surface” of the variables,
- the names “scanf”, “printf” of the functions, and
- the origin “4π” of the number “12.5663706”.

Even in this most simple example, only an expert can do an educated guess and recover the meaning of “x1”, “x2”, “f1”, “f2” and of the number “12.5663706”.

The size of the full source code of this program is 13 lines, including comments and empty lines. The typical size of a real-world program is between 10,000 and 10,000,000 lines of source code; some very large projects even reach 1,000,000,000 lines and above. Thus in realistic cases, there is no chance to reconstruct a useful source code from the binary code.

As a consequence, the task of software reverse engineering is extremely hard. It is never done to save work, but only as a last resort in some special situations, for example when there are no other means to achieve interoperability.

No Disclosure by Patents. Trade secrets can generate economic problems. When a company goes out of business, valuable knowledge might get lost. The patent system has been specifically designed to compensate for this kind of problem. This suggests that patents would be particularly useful in the software field as an incentive to disclose the source code. Unfortunately this does not work.

As will be shown in section 3.1.4 below, a typical software patent does not disclose source code. It does not even cover a complete realization of a software system, but only an elementary component of software—an *algorithm*. The source code discloses more than a patent, and is of more significance for a professional skilled in the arts. It remains a trade secret.

Free and Open Source Software. On the other hand there is a working system which encourages programmers to disclose their source code, Free and Open Source Software (FOSS).[10]

FOSS is based on copyright. When a software system is released under a FOSS license, the author deliberately discloses his source code under certain conditions. In exchange he gets

[10]For more information about free software see http://www.gnu.org.

access to a large code base of other FOSS. That way, even direct competitors can—and in fact do—pool their efforts to create better software. Competition takes place in terms of service quality, not of the code base.

To render this possible, the "certain conditions" above needed to be crafted very carefully. One essential condition for FOSS is that it can be copied freely by everyone who has obtained it in a legal way. In particular, there must not be any control over the number of copies. As soon as there are conditions which require control over the number of copies, FOSS cannot be used.

Patent licenses typically require a fee per copy. For this reason, it is very difficult to obtain a patent license for FOSS. Accordingly, there is a long list of FOSS projects which had to be terminated to evade a patent lawsuit.[11]

These development risks aggravate when the patent covers a standard file format or protocol. In such a case, even if there is a better algorithm than the patented one, there is no way around the patent without giving up interoperability,[12] and the patent effectively locks out FOSS from the field of programming where the standard is defined.

3.1.4 Examples

Example: US 5,960,411. *Method and system for placing a purchase order via a communications network.*

This is the "one-click shopping" patent by Amazon. Among all software patents, this might be the one most widely known in Europe. Its main claim reads:

> 1. A method of placing an order for an item comprising:
>
> - under control of a client system,
>   - displaying information identifying the item; and
>   - in response to only a single action being performed, sending a request to order the item along with an identifier of a purchaser of the item to a server system;

[11]See http://eupat.ffii.org/patents/effects/index.en.html.

[12]See the cases STAC, JPEG, MPEG, Dolby, VOIP, ASF, LZW, TTF, RSA, WWW, and RDF in footnote 11 above plus the examples in section 3.1.4 below.

- under control of a single-action ordering component of the server system,
  - receiving the request;
  - retrieving additional information previously stored for the purchaser identified by the identifier in the received request; and
- generating an order to purchase the requested item for the purchaser identified by the identifier in the received request using the retrieved additional information; and
- fulfilling the generated order to complete purchase of the item
- whereby the item is ordered without using a shopping cart ordering model.

From the claim we see that this patent is not about a *particular* method how to order an item in an online shop by using just a single mouse click. It is about the *abstract idea* to implement this type of online shop. The claimed object is not a concrete realization of one-click shopping but the principle of one-click shopping itself.

In this strict sense, this patent is *not* on "software", but on an abstract idea of a software component—an *algorithm*.

What are the economic effects of this patents? 23 days after it was granted, there was a lawsuit[13] which led to a preliminary injunction against a competitor which lasted from December 1999 to February 2001.

In addition to these direct effects, this patent raised a lot of uncertainty among programmers and potential founders of online shops.

In March 2003 there was a settlement between the opponents; the details have not been published. From February 2006 to October 2007 the patent was re-examined; finally 21 of 26 claims were rejected due to prior art.

In an open letter,[14] the CEO of the patent holder said that the company spent thousands of hours and millions of dollars

[13]Amazon.com Inc. v. BarnesAndNoble.com Inc., 239 F. 3d 1343 (Fed. Cir. 2001)

[14]See http://web.archive.org/web/20011204205032/www.amazon.com/exec/obidos/subst/misc/patents.html. A summary is available at http://www-cse.stanford.edu/ classes/cs201/projects-99-00/software-patents/amazon.html.

to develop the system. They subsequently patented the process because they did not want to see an innovation they spent time and money developing be adopted by their competitors.

Here we must carefully distinguish between expensive R&D and the actual *claimed object*. Most certainly, the development of an innovative online shop was expensive in terms of time and money. Anyway they do not need a patent to protect this investment against adoption by competitors.

They can allow everyone access to their online shop and can even sell binary copies of their software and keep the *source code* as a trade secret. Then, as we have seen in section 3.1.3, a competitor cannot simply copy the innovation, but needs to re-implement it from scratch and invest the same amount of time and money as the original innovator did. The primary innovator can stay ahead of his competitor by moving on to the next innovation. In other words, the rules of a free competitive market apply.

One may think a patent would imply disclosure of their innovation, but that is a misconception. The patent is on the *idea* to write a one-click shopping system and only discloses the *idea*. It does *not* disclose any detail *how* to implement such a system.

The Machine-or-Transformation test would have been helpful to prevent a patent grant and economic damage. However, as shown below, the Machine-or-Transformation test can be circumvented unless applied very carefully.

Example: US 5,736,943, also granted in Europe as EP 0,719,483. *Method for determining the type of coding to be selected for coding at least two signals.* Also granted in Europe as

This is one of many patents covering the audio compression standard *MPEG-1 Audio Layer 3*, publicly known as MP3.

The patent addresses the problem to decide, when coding a stereo signal, whether both channels should be coded together or separately. Its main claim covers the idea to base this decision on a (well-established) psychoacoustic calculation. (Thus the patent covers much more than "only" MP3.)

MP3 is not the best audio compression system, but it is the de-facto standard. A device or software which can play more advanced audio formats such as *Ogg-Vorbis* but cannot play MP3 is unlikely to have success on the market. As a result, the owner of the MP3 patents has control over the complete market segment of digital audio.

According to the licensing conditions[15] it is not possible to obtain a license for writing a FOSS system providing MP3.[16] Consistently, almost all existing FOSS MP3 *encoder* systems had to be abandoned due to legal pressure by the patent owner.[17] At the time of this writing, the patent owner seems to tolerate FOSS MP3 *decoder* systems, but there is no official statement about this which could provide legal certainty.

In another prominent case,[18] a compensation of $ 1.53 billion was demanded for the infringement of two MP3 patents.[19] This illustrates that even obtaining a license does not provide legal certainty, because the same algorithm might be covered by a multitude of patents by different owners.

This example illustrates a drawback of the Machine-or-Transformation test. The application of the MP3 algorithm involves a machine, a standard computer or a dedicated MP3 player, thus the MP3 patents would have passed the test. Nevertheless, the Machine-or-Transformation test can be a useful tool for deciding whether a patent should be granted or not—see section 3.2.2 below.

Example: US 5,546,528, also granted in Europe as EP 0,689,133. *Method of displaying multiple sets of information in the same area of a computer screen.*

This patent covers the idea to attach tabs to dialog windows in programs. Claim 1 is about establishing a small region on the screen ("tabs") such that an action in the tabs (typically a mouse click) changes the contents of a larger region on the screen ("palette"). Besides tabbed dialogs, this also covers menu structures of web pages and much more.

Programmers agree that this is a trivial feature of dialogs which was well-known before the patent was filed in 1994. European patent attorneys agree that this patent meets high standards and that it only looks trivial in retrospect. In any

[15]See http://www.mp3licensing.com.

[16]Please note that FOSS is explicitly free *for commercial use*, thus the exception "for non-commercial use" does not apply here.

[17]See http://swpat.ffii.org/patente/wirkungen/mpeg/index.en.html.

[18]Lucent Technologies Inc. v. Gateway Inc., 470 F. Supp. 2d 1180 (S. D. Cal., 2007)

[19]US 5,341,457 and US 5,627,938

case, the patent is an exclusive right on the abstract idea to create this type of dialogs—in contrast to the copyright on an individual program or library implementing that idea.

In 2002, the court of Delaware held up this patent and rejected all prior art presented by the defendent, who had to pay a compensation of $ 2.8 millions.[20]

Almost any end-user program infringes this patent. If the patent holder wants to, he can sue about everyone who produces software. This kind of legal uncertainty is the economic impact of patents granted on abstract ideas rather than concrete machines or transformations.

How would the Machine-or-Transformation test apply to this patent? Again it can be circumvented by tying the patent to a machine—a computer with a display and an input device (mouse).

As a conclusion, the Machine-or-Transformation test needs to be improved in order to fullfill its purpose to sort out patents with a negative economic impact.

Example: US 5,787,449. *Method and system for manipulating the architecture and the content of a document separately from each other.*

In its claim 1, this patent covers the idea to separate, in a text document, the actual text from its structure (sections, emphasis, etc.) and to resolve both back into the document. This is one basic algorithm for text processing and an essential component of most text processors.

In an ongoing case,[21] a verdict which would have prevented Microsoft Inc. selling its software *Word* was reverted. In their appeal, Microsoft stated that this prevention would irreparably harm their company. The computer vendors HP and Dell supported Microsoft's appeal by stating that the software Word is widely used and it would be against the public interest to take it away from the market.

While the market share of Microsoft Word is certainly above the average, these arguments hold in fact for *all* software prod-

[20]**http://www.heise.de/newsticker/Adobe-gewinnt-Patentklage-gegen-Macromedia–/meldung/27109**

[21]**http://www.heise.de/ct/I4i-erhebt-neue-Vorwuerfe-im-Patentrechtsstreit-mit-Microsoft–/news/meldung/145146, http://www.techflash.com/microsoft/Microsoft_rails_against_Word_injunction_This_is_not_justice_55035182.html**

ucts. Every company is irreparably harmed when selling their major product is prevented, and it is always against the public interest to take away products from the market. It is a question of balance whether this harm exceeds the harm done to the patent holder by selling the product.

3.1.5 Implications

As we have seen in the previous section, patents on software and business methods have a high potential to create economic problems. Why is this so?

The existing software patents are in fact not patents on specific software *realizations*, but on abstract fundamental principles of programming, the *algorithms*. This brings an extremely high risk of patent infringement to software developers.

A patent claim typically takes 10–100 lines to describe the algorithm in human language. Computer languages have been specifically designed to describe algorithms, so we can expect an algorithm to fit in typically less than 100 lines of source code.

As demonstrated in section 3.1.3, software consists of a large number of "ideal parts"—the algorithms. A typical software system comprises 10,000 to 10,000,000 or even more lines of source code, which accounts for a minimum of 100 to 10,000 algorithms, each of which might be covered by a patent.

So instead of one patent covering one software *realization*, we have one software realization covered by hundreds or thousands of patents on software *algorithms*.

As a consequence, it is unfeasible to write software without infringing a large number of patents. Each patent infringement can result in a lawsuit. In this sense, software patents generate a "minefield" for software authors.

While this is a problem for all types of software business models, it specifically endangers Free and Open Source Software (FOSS). As substantiated in section 3.1.4, most patent holders refuse to grant licenses to FOSS, and many FOSS projects had to be abandoned due to patents.

So instead of fulfilling their purpose to promote disclosure of information, software patents inhibit free sharing of source code.

3.1.6 Case Law

In Europe, U. S. patent law is percieved as without significant limits. For example in 2004, Brigitte Zypries, the German Federal Minister of Justice at the time, stated:[22]

> In sharp contrast to the Americans, we are working on a distinction by requiring technology, inventive step, novelty and industrial application for patentability. These are severe limitations. From this it follows that in Germany, much less is patentable than in America.

The European Patent Convention, in its Art. 52(2)(c) explicitly states that “schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers [. . .] as such [. . .] shall not be regarded as [patentable] inventions”. However, the difference between the patent systems of the EU and of the United States are often overestimated. Even the often cited principle—and likely source of such a misconception—of granting patents for “anything under the sun” has its limits: Patentability shall extend *only* to anything under the sun that is *made by man*,[23] Furthermore, patents shall be granted on processes only to an extent where such a thing *made by man* produces a *useful, concrete and tangible result*.[24] Last but not least, this principle and its limitations can be understood only as embedded within the general exclusions of abstract ideas, natural laws and natural phenomena from patentability.

But this is only the most basic set of rules that arises if we look at the U. S. patent system “at its most generous state” and the broadest possible framework for the granting of patents in the U. S., i. e. the minimum limits having emerged throughout the last decades since Diamond v. Diehr.[25] It is apparent that the intention of such rules is far from patentability without *any*

[22]Interview with German Federal Minister of Justice Mrs. Brigitte Zypries, c't 16/04, http://www.heise.de/ct/Ein-c-t-Gespraech-mit-Justizministerin-Zypries-und-Ministerialdirektor-Hucko-ueber-geistiges-Eigen--/artikel/125352

[23]Rep. No. 1979, 82d Cong., 2d Sess., 5 (1952), H. R. Rep. No. 1923, 82d Cong., Sec. 2d Sess., 6 (1952)

[24]State Street Bank & Trust Co. v. Signature Financial Group Inc., 149 F. 3d 1368 (Fed. Cir. 1998), Ex parte Lundgren, Board of Patent Appeals and Interferences, United States Patent and Trademark Office (2004)

[25]Diamond v. Diehr, 450 U. S. 175 (1981)

limitations: To give only two examples, a machine that does not produce any useful result shall not be patentable and even though an idea might be considered "made by man", no patent shall be granted on it.

It is for good reasons that, even in cases arguing for the broadest patentability, the need for limitations is endorsed.

This Court has already elaborated in a detailed manner on how to translate the rules and limitations to the daily practice of granting patents. In Gottschalk v. Benson,[26] it was held that a procedure for solving a given type of mathematical problem is known as an "algorithm". Allowing a patent on such a procedure would "wholly pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself". Nevertheless, the mere presence of an algorithm in a solution shall not per se exclude the possibility of a patent being granted, as long as all other requirements of patentability are met.[27]

But taking a closer look at this set of limitations and how they relate to algorithms, a few questions remain:

1. Is an algorithm an abstract idea in the sense of patent law?
2. When is the result of an algorithm/calculation "tangible"?
3. Must the tangible result be part of the same claim?

Ad 1. Until after Diamond v. Diehr,[28] yes. Only In re Hiroyuki IWAHASHI[29] started to reverse this.

Ad 2. According to Diamond v. Diehr, any claimed algorithm has to be treated as prior art. The patent can be granted only if the same claim as the algorithm contains also something else that produces a useful, concrete and tangible result.

Ad 3. According to Diamond v. Diehr, *something else that produces a tangible result* has to be a part of the same claim as the algorithm.

[26]Gottschalk v. Benson, 409 U. S. 63 (1972)

[27]Parker v. Flook, 437 U.S. 584 (1978), Diamond v. Diehr, 450 U. S. 175 (1981)

[28]Diamond v. Diehr, 450 U. S. 175 (1981)

[29]In re Hiroyuki IWAHASHI, Yoshiki Nishioka and Mitsuhiro Hakaridani, 12 U. S. P. Q. 2d 1908; 888 F.2d 1370 (1989)

However, almost all these questions have been answered by specialized Courts and institutions in a way contradicting the spirit of the decisions by this Court. The complexity of these rules and the fact that they are laden by terminology that is prone to interpretation shows that more clarity is not only desired, but vital. Actually, the history of case law since Diamond v. Diehr might serve as an example how decisions of lower Courts and institutions can extend the case law of this Court, possibly acting against the reasoning underlying these decisions.

In Diamond v. Diehr, the definition of "algorithm"

> "1. A fixed step-by-step procedure for accomplishing a given result; usually a simplified procedure for solving a complex problem, also a full statement of a finite number of steps. 2. A defined process or set of rules that leads [sic] and assures development of a desired output from a given input. A sequence of formulas and/or algebraic/logical steps to calculate or determine a given task; processing rules."[30]

was significantly broader than the definition this Court employed in Benson and Flook and therefore did not adopt it. The court also indicated, that drawing a distinction between 1. and 2. would be artificial by stressing that even processes falling within the definition offered by the petitioner, would constitute patentable subject matter.

Creating an artificial dichotomy from what actually is explained as synonymous in the source cited by the petitioner himself does indeed look more like an unnecessarily clumsy and pretentious way of saying that abstractions/logic/software can be claimed in a patent, when it was already clarified by this Court that it cannot.

Nevertheless, the United States Court of Appeals for the Federal Circuit introduced this artificial dichotomy In re Hiroyuki IWAHASHI,[31] making it the basis of this decision:

> Over-concentration on the word "algorithm" alone, for example, may mislead. The Supreme Court

[30]Brief for Petitioner in Diamond v. Bradley, O. T. 1980, No. 79–855, p. 6, n. 12, quoting C. Sippl & R. Sippl, Computer Dictionary and Handbook 23 (2d ed. 1972).

[31]In re Hiroyuki IWAHASHI, Yoshiki Nishioka and Mitsuhiro Hakaridani, 12 U. S. P. Q. 2d 1908; 888 F.2d 1370 (1989)

> carefully supplied a definition of the particular algorithm before it [in Benson], i. e., “[a] procedure for solving a given type of mathematical problem.” The broader definition of algorithm is “a step-by-step procedure for solving a problem or accomplishing some end.” Webster’s New Collegiate Dictionary (1976).

The Court of Appeals thereby implied that there would exist mathematical, abstract algorithms as opposed to (non-mathematical, non-abstract) “applied” algorithms. This then was reiterated and wholeheartedly adopted in State Street[32]

> Today, we hold that the transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces “a useful, concrete and tangible result”—a final share price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades.

This effectively reversed the holding of Gottschalk and Parker, explained in more detail in Diehr that

> [. . .] the algorithm is treated for 101 purposes as though it were a familiar part of the prior art; the claim is then examined to determine whether it discloses “some other inventive concept.”

to any calculation can be *claimed on itself*, as long as the patented solution as a whole constitutes a useful, concrete and tangible result.

In Ex parte Lundgren,[33] this was even further extended, as it was stated that ”useful arts” in U. S. Const., Art. I, § 8, cl. 8 cannot be understood as a limitation to “technological” arts.

Summarizing, the case law of the Supreme Court defined a clear line, in which algorithms could not be claimed on their

[32]State Street Bank & Trust Co. v. Signature Financial Group Inc., 149 F. 3d 1368 (Fed. Cir. 1998)

[33]Ex parte Lundgren, Board of Patent Appeals and Interferences, United States Patent and Trademark Office (2004)

own and had to be accompanied—in the same claim—by something else meeting the requirements of patentability. This is not unlike the decisions of the Board of Appeals of the European Patent Office, which interpreted the so-called "technology requirement" in an analogous way until it sought to harmonize its practice with the United States after State Street.[34]

3.2 Determining the Boundary

3.2.1 The European Approach: Art 52 EPC and Technicity

European law provides for an explicit exclusion of patents on software and business methods. Art 52 of the European Patent Convention (EPC) reads:

"(2) The following in particular shall not be regarded as [patentable] inventions [. . .]:

(c) schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers; [. . .]

(3) The provisions of paragraph 2 shall exclude patentability of the subject-matter or activities referred to in that provision only to the extent to which a European patent application or European patent relates to such subject-matter or activities as such."

In the light of Art 52(2)(c) EPC it is counterintuitive that patents like EP 0,719,483 and EP 0,689,133 were granted. What did happen?

On 20th February 2002, the Commission of the European Communites submitted a Directive proposal[35] to the European Parliament. On page 7 it states that computer programs which have "a technical character [. . .] are not considered to fall under the exclusion in Article 52(2) as they are considered not to relate to programs for computers 'as such'."

So the Directive distinguishes between "technical" and "non-technical" software. What is the difference?

[34]State Street Bank & Trust Co. v. Signature Financial Group Inc., 149 F. 3d 1368 (Fed. Cir. 1998)

[35]Commission of the European Communities, Directive Proposal COM (2002) 92 final 2002/0047 (COD), 2002

On the same page we find that “all programs when run in a computer are by definition technical (because a computer is a machine) [. . .].”

In other words: all software would be technical, thus the exclusion by Art. 52(2) EPC would be void.

This Directive proposal was rejected by the European Parliament on 6th July 2005 and never came into effect. Nevertheless it illustrates how Art. 52(2) EPC can be circumvented. It explains how it was possible that the EPO granted tens of thousands of patents on software algorithms and business methods in spite of Art. 52(2) EPC.

In an ongoing case,[36] the Enlarged Board of Appeal of the EPO is addressing the circumvention of Art. 52(2)(c) EPC in a similar way.

Question 2 of the case reads:

> “2.(a) Can a claim in the area of computer programs avoid exclusion under Art. 52(2)(c) and (3) merely by explicity mentioning the use of a computer or a computer-readable data storage medium?
>
> (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?”

A “yes” to question 2.(a) would be equivalent to the Directive proposal.

In a more subtle way, a “yes” to question 2.(b) does also justify patents on software and business methods. The “further technical effect” is not required to be new. It is sufficient to include a well-known technical device along with a patent claim on a (new) software algorithm to obtain the patent.

The same mechanisms can be applied to circumvent the Machine-or-Transformation test proposed in the Bilski case. To establish the exclusion of abstract ideas, further improvements are required. This will be addressed in section 3.2 below.

3.2.2 The Machine-or-Transformation Test

According to the Machine-or-Transformation patent eligibility test, a claim to a process qualifies to be considered for patenting

[36]Enlarged Board of Appeal of the European Patent Office, case G 3/08, 2008

only if it (1) is implemented with a particular machine, that is, one specifically devised and adapted to carry out the process in a way that is not concededly conventional and is not trivial; or else (2) transforms an article from one thing or state to another.

This test is a means to an end. It is there not to cross, but to define the border between patentable new and useful processes and non-patentable laws of nature, physical phenomena, and abstract ideas. This answers the first question presented by the petitioners.

Regarding the second question, the American Inventor Protection Act in 1999 introduced a limited "prior use" defense in American patent law, specifically for "methods of doing or conducting business". This regulation, inserted as § 273 into the American Patent Act, implicitly acknowledges the existence of business methods, by virtue of its limitation to those methods.

Unlike the second question of the petitioners suggests, 35 U. S. C. § 273 does *not* reflect a "clear Congressional intent that patents protect 'method[s] of doing or conducting business.'" The regulation only recognizes business method patents as a fact and tries to bring their economic impact under control by setting a limit.

This is achieved by the Machine-or-Transformation test: setting a limit. It does not contradict 35 U. S. C. § 273, but complements it.

On the other hand one must take very seriously the concerns raised by Judge Mayer in his dissent to this test.

> "Bilski, for example, could simply add a requirement that a commodity consumer install a meter to record commodity consumption. He could then argue that installation of this meter was a 'physical transformation'."

The same kind of "clever draftsmanship" has been advocated in Europe by the EPO, which for many years saw its job as one of "helping its clients, the would-be patent holders, apply for *and receive* patents" and featured on its website a document which described in detail how to circumvent Art. 52(2) EPC.

3.2.3 Improving the Patentability Tests

The purpose of the the Machine-or-Transformation test, reiterated in Bilski, is to limit patentability and to exclude undesired patent applications. To serve this purpose, it must be accompanied with other measures. Here is a list of a few features that would be most helpful.

1. The Machine-or-Transformation test must be applied to *the claimed object* rather than to the patent claim as a whole. An object can only be claimed in a patent if it constitutes a new, concrete realisation of a machine or transformation.

   Without these further requirements, patent claims will incidentally involve machines or transformations which are irrelevant to the claim, but serve to patent laws of nature, physical phenomenon or abstract ideas.

2. In applying the "suggestion test", an ability to combine or modify prior art references should be assumed that is consistent with ordinary creativity and problem-solving skills in the art.

   The suggestion test states that if the prior art would have already suggested the claimed invention, then the claimed invention is obvious. However, the participants at the FTC/DOJ Hearings expressed concern with some recent applications of the suggestion test. To show that a claimed invention is obvious, the USPTO had to point to particular items of prior art that *concretely* suggest how to *combine* all of the features of a claimed invention. It is those fields where the combining of elements is most straightforward that the chances are least that one can find an existing document that outlines that particular combination, simply because nobody having ordinary skill in the art would ever require instruction at that level. How to combine existing elements would be considered "too obvious" to need mention.

3. An expanded version of the suggestion test is relevant to all inventions that use a computer. Simply writing a computer program and then running it on a computer is not novel, under the meaning of patent law. And doing so is obvious, again under the meaning of patent law.

   These days, we experience a similar dawn of mankind as Henry Ford experienced 100 years ago.[37] Back then it was

[37]See also the FTC report, executive summary, page 12

completely obvious for an engineer to put an engine on a carriage. In these days, it is completely obvious for someone skilled in the art of computer programming to convert mathematical principles into computer programs. Thus while what Bilski has done may be termed "innovative" in an entrepreneurial sense, it is in no way innovative in the sense of patent law.

4. The "commercial success test" should be abondoned. This test states that in some circumstances, courts may consider the commercial success of a claimed invention to indicate that it was not obvious. It must have satisfied an unmet need, which was not obvious to anybody else, or else they would have already been satisfying it.

   Commercial success comes from many factors, many of which have nothing to do with the claimed invention, e. g. Marketing, advertising, a pre-existent dominant position, etc. Henry Ford, after all, claimed that his business success was based on the completely obvious idea of putting a gasoline engine on a chassis. His great success arised from other factors.