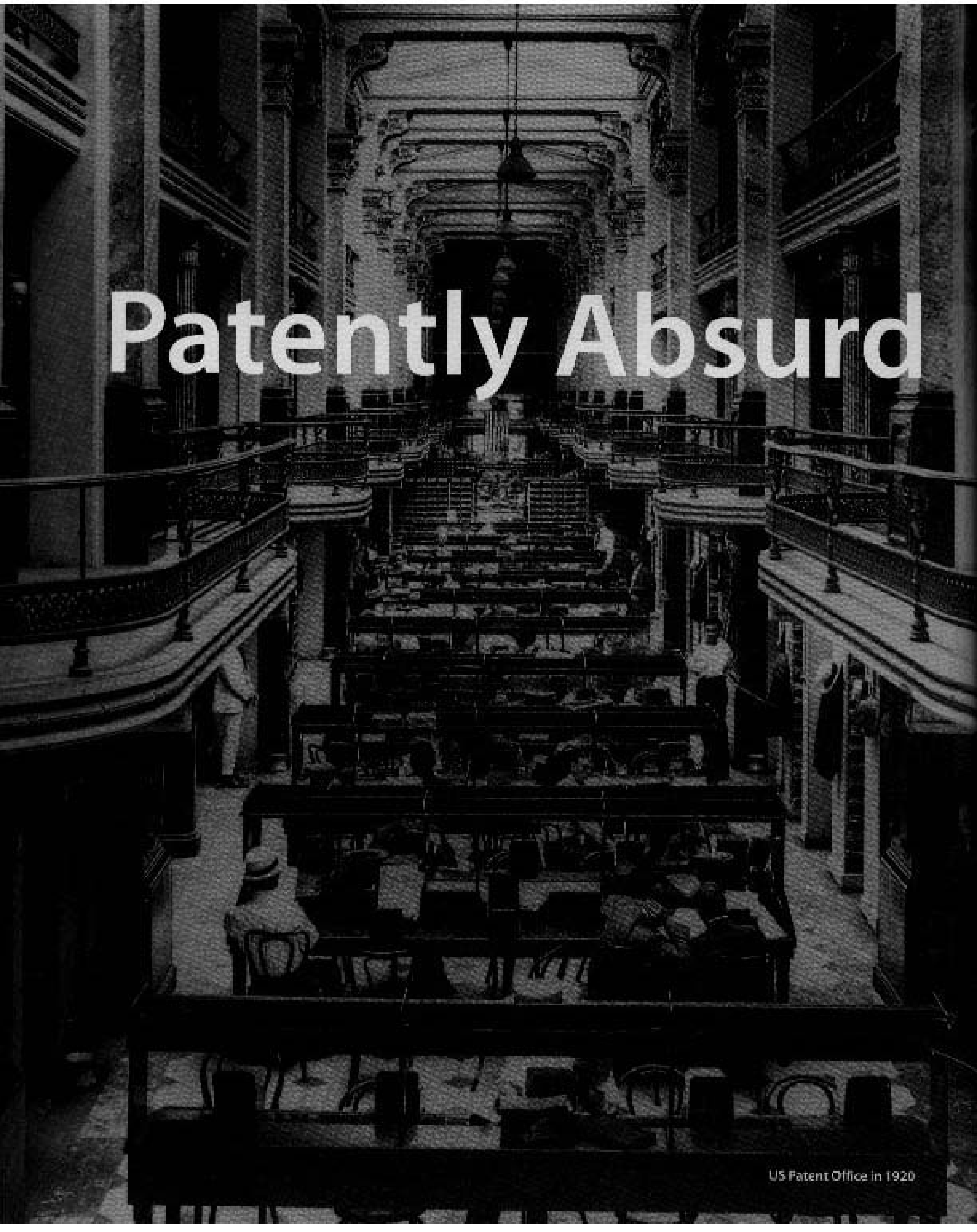
Patently Absurd


US Patent Office in 1920

How could the Patent Office ever grant a patent to Compton's on its claim to have invented multimedia?

This is how.

Simson L. Garfinkel

takes us inside the Patent Office.

Something is terribly wrong in Crystal City, and everybody who's thinking about intellectual property knows it.

The problem is rooted in a federal office complex across the Potomac River from Washington, DC, on the fourth floor of Crystal Park 2. That is the home of the US Patent and Trademark Office's Group 2300, the group called "Computer Systems and Computer Applications." Group 2300 is the one that's been carving up computer science and handing out little monopolies to nearly half the people who take the time to file a patent application.

Heather Herndon is a patent examiner in Group 2300. Herndon sits at her desk in an office that's crowded but meticulously organized. At her left is a pile of "file wrappers"– bulging blue folders held together with rubber bands and steel clips – that hold patent applications in progress. At her side is a Dunn 386 PC – a cheap IBM clone – with a low-speed 2400-baud modem. All around her are cabinets and drawers filled with paper files: mostly old patent applications, as well as some particularly important articles that summarize landmark advances in computer science. And on the table right before her is a prime example of what all the fuss is about: a software patent.

The really interesting software patents are the ones that examiners like Herndon and attorneys around the country are fighting over right now – the ones that haven't yet been granted. And under US law, the Patent Office is forbidden from discussing or revealing any application in progress.

Because of the confidentiality rules, Herndon has prepared for this meeting (one of the first granted by the Patent Office between a journalist and an examiner on the subject of software patents) by going into her files and finding a patent that's already been issued. It's Patent Number 5,060,171, "a system and method for superimposing images," issued to Stephen C. Steir, et al.

Steir's patent describes "an image enhancement system and method" that shows people what they might look like with a different head of hair. It uses a simple technique to combine an image of a person's head with a second image of a disembodied hair piece: first you find and smooth the boundary between the two images, then you figure out a new color for each pixel in the boundary by averaging the pixels in the nearby vicinity. The patent describes how you can change the size of the hairstyle image, so that a single hairdo can be placed over tall, thin heads or short, fat ones.

Simson L. Garfinkel (simsong@mit.edu) *is a computer consultant and science writer.*

Read through Patent Number 5,060,171 and you'll learn everything about how Steir's invention works. If you are a programmer, you might be tempted to go home and throw together a few hundred lines of C++ to try out the algorithm yourself. You might even demonstrate that program to your bald Uncle Arthur, who happens to be considering a hair transplant.

But if you do, you'll be breaking the law.

That's because Patent Number 5,060,171 gives Clearpoint Research (the company to which Steir et al assigned their patents) an absolute monopoly on the techniques it describes – until October 22, 2008. And that monopoly is backed up by the federal courts, the federal marshals, and ultimately the entire United States Army.

Ignorance Is Not a Defense

There are three main ways to prevent people from ripping off your intellectual property in the United States: trade secrets, copyrights, and patents. But until recently, few programmers or lawyers bothered to patent the techniques described by computer programs. Many people thought that patenting computer programs was neither ethical nor legal; most of the rest thought that it wasn't necessary.

Programs, after all, were protected by copyright laws. What's fundamentally vexing about software patents is that they can be unknowingly violated by any programmer, even one who has never heard of the patent and who independently implements the invention. What may seem like a trivial hack to a gifted programmer may already be a patented routine.

Just about the only way programmers can protect themselves from patent infringement is by paying for a patent search – both a time-consuming and an expensive process – which still doesn't conclusively determine whether or not a program is in the clear.

A patent search can cost anywhere from a few hundred dollars to a few thousand. Simply finding the software patents can be a challenge, since the Patent Office doesn't identify them as such. Then there's the sheer number of software patents that the Patent Office has issued: more than 12,000, according to Gregory Aharonian, who runs the Internet Patent News Service, a free Internet mailing list that tracks newly issued patents and developments in intellectual property law. Aharonian has been tracking government-funded software for the last ten years and patents for the last three.

Beyond the Patent Office, complications are born of the nature of software itself. Develop a new drug or a new electronic circuit: seeing whether the invention has ever been patented before is a fairly straightforward task. But after nearly a decade of granting software patents, the Patent Office still hasn't come up with a system for organizing them that's in a language that most programmers can understand. Many software patents – such as Patent Number 5,060,171 – don't even have the words "computer," "software," "program," or "algorithm" in their abstracts, making them harder to locate for those unfamiliar with the terrain. And no easy way exists to take a few lines or a page of computer code and get a list of the patents that it might violate. Identifying all of the patentable inventions inside an entire program is nearly impossible. A typical piece of software might violate three. Then again, it might violate a hundred.

So, patent searches for computer programs become expensive propositions. But violating a patent can be more expensive still. According to Stanford University Professor John Barton, patent infringement suits are among the most expensive kind of litigation in the US today, with the average cost of a patent suit being US$500,000 per side per claim. Not surprisingly, the cost of insurance to protect companies against patent infringement is equally steep: $50,000 per product with a $50,000 deductible in the case of multimedia software, says Rob Lippincott, president of the Interactive Multimedia Association, a trade organization for large and small multimedia publishers. "These kinds of numbers are basically intolerable," says Lippincott, adding that the cost of merely defending an infringement will wipe out most small software houses, whether they win or lose.

But merely defending an infringement suit is peanuts next to losing one. Just ask Microsoft, which was found guilty by a jury of violating STAC Electronics's patent on data compression in February 1994. The cost: $120 million in compensatory damages.

The alternative to a lawsuit, of course, is licensing any software patents that a program might happen to infringe. But then the costs can really swell. For instance, the REFAC Technology Development Corporation of New York holds Patent Number 4,398,249 on natural order recalculation. In 1989, REFAC sued Lotus, demanding 5 percent of all revenues from Lotus 1-2-3. Five years later, the case is still pending in a New York federal court. And last year, when Compton's New Media, a division of Tribune Co., was awarded a far-ranging patent in multimedia databases, the company indicated that it wanted 1-3 percent of net revenue for any potentially infringing multimedia product. License just a few of these patents, and you've licensed away your profit margin.

Such dangers terrify small businesses, shareware authors, and people writing free software. A person writing a simple utility can find himself or herself at the wrong end of a multimillion-dollar lawsuit for lost profits, with treble damages if a court finds that the infringement was "willful." "Software patents turn every decision you make while writing a program into a legal risk," says Richard Stallman, one of the leaders of the opposition to software patents. "They make writing a large program like crossing a minefield. Each step has a small chance of stepping on a patent and blowing you up."

The League for Programming Freedom, which Stallman helped create, has called for the elimination of patents that apply to computer programs. But it may be too late for such sweeping and simplistic reforms.

Who Are These Guys?

At the heart of the software patent crisis is the Patent and Trademark Office itself. In recent years, the office has issued a steady stream of patents on techniques that nobody would have dreamt patentable just a few years before. The patents can be dauntingly broad. Recently, for example, the Patent Office awarded Patent Number 5,173,051, which describes a system for "curriculum planning and publishing" using a computer and a videodisc player. The patent was filed October 15, 1991, by Optical Data Corporation of Warren, New Jersey, and issued December 22, 1992 – record time for a patent that, in the words of Harvard researcher Brian Kahin, "makes out millions of teachers to be infringers."

Then there is Patent Number 5,105,184, the so-called "Energizer Bunny Patent," issued in 1992 to Software Advertising Corp. It covers "displaying and integrating commercial advertisements with computer software" – in other words, any advertisement integrated into a screen saver. Or Patent Number 5,263,127, granted last year to an enterprising engineer at Digital Equipment Corp., which patents a technique used in object-oriented programming that can be implemented by using exactly two machine-language instructions.

This year, the Patent Office will issue more than 100,000 patents, 4,000 of which could be classified as software patents, says Gregory Aharonian. Since it typically takes two to four years for the Patent Office to grant a patent, and since newly issued patents apply retroactively to any product that was created after the patent application was filed, companies that want to avoid an infringement suit need to keep track of the new patents – a full-time job for a team of highly trained people.

As a result, few developers realize that they are infringing a patent until "they get the letter from the law firm that they've never heard of in the city they've never been to that says they are infringing US Patent Number so-and-so," says Robert Merges, a professor of patent law at Boston University.

One person who found himself on the wrong side of such an infringement suit is Vern Blanchard, a programmer whose company was destroyed by a patent that wasn't even valid.

Blanchard is president of American Multi-Systems, a San Diego-based company that wrote a program to let professional bingo players play dozens of bingo cards at the same time. The program runs on an IBM PC. Blanchard was ready to start marketing his system, along with custom-built tables and personal computers, to the big-time bingo halls, when one of his competitors filed suit against American Multi-Systems for patent infringement.

The case should have been thrown out of court for two reasons, says Blanchard. For starters, he says, his competitor's patent "covered a hand-held calculator type device," not a general-purpose computer running a program. And the patent couldn't cover general-purpose computers, he says, because programs that play bingo are commonly written by students in introductory computer science courses. There was simply nothing novel or new about the technique that the patent described, and novelty is a basic requirement of patentability.

Nevertheless, says Blanchard, the company that held the patent was able to convince a judge to grant a preliminary injunction that took American Multi-Systems's product off the market.

Eventually Blanchard discovered a critical piece of "prior art" – concrete evidence that the invention described by the patent had been thought of before by somebody else – and was able to convince the judge

The Patent Office is quite good at recruitment and training. It has to be.

Few examiners stay on for more than a few years. An exception is Heather Herndon, shown here among piles of pending applications.

The Automated Patent System (below) is one of the largest databases in the world, according to the Patent Office.

If it can't be found online, it might be hiding somewhere in the paper files. And you wonder why it takes years for a patent to be approved?

to lift the preliminary injunction. Unfortunately, by that time American Multi-Systems had effectively put itself out of business with legal fees.

"Judges are not particularly literate in technical issues," says Blanchard. "When they see a patent they presume that it's valid, as they should. (To them), if the Patent Office says that this is a valid patent, well, of course it's a valid patent."

In practice, these simple rules have created headaches for the 150 patent examiners of Group 2300.

In order to be a patent examiner, you've got to have a college degree in science or engineering, including at least 30 credit-hours of specific science-related courses, before applying for the job and taking an exam. Once you are hired, the Patent Office's in-house training takes over.

Upon hiring, every examiner gets an intensive two-week course that teaches the basics of patent examination – the standards of patentability, where to look to find prior art, and how to evaluate patent documents.

Patents may be less justified in the world of software, where major application programs can be developed by a few people working in somebody's living room.

That course is followed by four more two-week sessions over the next eight months. But the real training for a new patent examiner comes on the job. Each new examiner's supervisor has the ultimate signature authority over the newcomer's patent applications. Trainees don't get to sign their own patents until they've been at the Patent Office between three and five years.

The Patent Office has gotten quite good at recruitment and training. It has to be: few examiners stay on for more than a few years. This despite the benefits that being a patent examiner carries: high pay by federal standards ($38,000 to $75,000 a year, before overtime), flex-time arrangements, and tremendous job satisfaction.

At the same time, the job invites burnout. Each examiner has a quarterly quota that must be filled. As a result, examiners frequently put in overtime: up to 40 hours every two weeks. Since applications must be held in the utmost secrecy, examiners are forbidden to work at home, which makes long hours even less attractive.

Then there is the honey pot of private practice.

The Patent Office encourages its examiners to take courses at nearby law schools so that they can improve the job they do. Unfortunately, the idea frequently backfires: soon after examiners earn their law degrees, they frequently leave the government for a lucrative job in private practice on the other side of the patent bar.

"I know somebody who recently left the office who had been in there approximately three or four years. Their take-home pay doubled when they left the office. And they expect increases in salary of $15,000 to $20,000 over the next year," said David Clark, a patent examiner dropout.

Idea v. Implementation

No matter how good it is, you can't patent an idea in the United States: patents are only awarded in this country for usable inventions that accomplish tasks.

Computer programs, occupying that strange world between mathematical ideas and applied engineering, have posed a problem for the courts since the 1960s. On the one hand, the public has traditionally viewed computer programs as mathematical. On the other hand, programs can solve real problems.

In 1972, the US Supreme Court spoke for the first time on the subject of software patents. In the case of *Gottschalk v. Benson*, the court denied a patent on a system for converting binary-coded-decimal numbers into decimal numbers. The Court's decision was based on the notion that code was preeminently mathematical. The Court's decision said, in part, that if patents on algorithms were allowed, "the patent would wholly preempt the mathematical formula" for other uses.

Faced with the Court's ruling, patent attorneys simply bypassed the problem by framing the language of their patent applications so that software inventions seemed like hardware devices. For example, in July 1973, AT&T filed for a patent on the fundamental technique used by the Unix operating system to enforce computer security (this eventually came to be known as the SUID Patent). But instead of describing the invention as a software code, AT&T's attorney, Stephen Phillips, described the invention with a circuit diagram containing 11 chips connected by more than 40 wires. (The patent was granted in January 1979 but was dedicated to the public domain just ten months later.)

Disguising software patents with hardware implementations was a common trick, says Rick Jordan, patent counsel at Thinking Machines, a maker of supercomputers. "Instead of being up-front about the fact that their product was embodied in software, people would go through subterfuge to make it look like a product was embodied in hardware," and thus make it eligible for a patent as a device which performed a process or embodied a technique.

But Phillips needn't have bothered with his subterfuge. Just five years after the Benson case, the US Court of Customs and Patent Appeals reinterpreted the Supreme Court's decision. In a case called *In re Freeman*, the appellate court upheld a patent on a system for typesetting mathematical equations, arguing that the Supreme Court really meant that the Benson case only forbid patents on "mathematical algorithms." Unfortunately, the appellate court neglected to define the term "mathematical algorithm."

The search for the proper definition of the term "software patent" may be moot, argues Joe Dixon, a supervisory patent examiner who runs Art Unit 2312, or "Storage and Retrieval." He adds: "We do not grant 'software patents.' " What the Patent Office grants, insists Dixon, are patents on methods or processes that can be embodied in computer programs. This is the reason that words like "software," "program," and "algorithm" don't usually appear in abstracts of most so-called software patents: the law is indifferent as to whether the invention is built with a program or with a bunch of integrated circuits and wires.

Obtaining software patents got easier in 1981, when the Supreme Court ruled in favor of a patent applicant in *Diamond v. Diehr*. In that case, Diehr had applied for a patent on a system for vulcanizing rubber that used a computer program to control the temperature of the rubber mold. The Court ruled that the patent application's inclusion of a computer program didn't render the application unpatentable. In essence, the 1981 decision gave the clear impression to the lower courts and the Patent Office that the Court felt software patents were OK. Since then, the number of software patents granted each year has nearly doubled.

The Problem with the Prior Art

It is famously difficult to figure out whether a software patent application describes an invention that is new or novel. Under Section 102 of the US Patent Code, the word "novel" has an exact legal definition. Specifically, an inventor may not receive a patent if the invention already exists in the prior art – that is, if the invention is patented or described in a printed or online publication anywhere in the world before the date on which the inventor files his or her patent application. In cases where two people claim the same invention (perhaps because they independently made the same discovery), Section 102 requires that the person who is awarded the patent be the first person to know and use the invention.

One thing is certain about prior art: there's a lot of it. In the *In re Hall* case (decided in 1986), a PhD dissertation on the shelf of an obscure European university library was deemed to be part of the prior art, and a patent application was thrown out on appeal by the Federal Circuit Court. A catalog sent by one French company to a few hundred customers in Germany was found by a court to be part of the prior art. Indeed, the courts have intentionally stayed away from deciding whether prior art is "good" or "bad." No matter whether it appears on the front page of *The New York Times* or in a Russian technical journal that's never been translated into English, if the prior art describes an invention, then the invention is not patentable.

In almost every field that the Patent Office covers, examiners determine whether an invention is new by searching two kinds of computerized databases: the Patent Office's own Automated Patent System, which tracks more than five million patents extending back to the 1790s, and commercially available databases of scientific literature. Got a patent application for a new drug? Check the databank. If the drug's not there, it's probably patentable. This approach is fine for tangible things like drugs. But what if we enter the Alice-in-Wonderland of software patents and try to figure out, say, if the search techniques used by the Patent Office database program itself are patentable? Good luck.

"We search the patent database, both US and foreign, and we search every commercial database," says Bruce Lehman, commissioner of patents and trademarks. "But there are many concepts that have been done which are what I call folklore. They are out there, and people know about them, but we can't find any written documentation. The examination process requires that we have a written document which we can point to which states a particular fact. Too often we can't find that documentation. Then when the patent is issued, some people say, 'Well, this is well-known, it has been in the industry for years.' "

Searching for the Needle

Roughly 80 patent applications show up on Heather Herndon's desk each year. (The applications are randomly assigned among the dozen or so examiners in Herndon's unit – a process which assures fairness and theoretically prevents the examiners in the group from overspecializing.) A new application waits for a few months, until Herndon has a free moment to crack its wrapper. The first thing she does with a new patent application is read the inventor's "claims" to figure out what kind of invention is actually being described. Over the years, the Patent Office has developed a taxonomy of inventions, an elaborate system with classes and subclasses, each with its own numeric code. Making this classification is a slow, error-prone procedure.

Once she's decided in which category the invention belongs, Herndon walks across the hallway to the terminals that are part of the Patent Office's Automated Patent System, a twenty-year, half-billion-dollar effort to automate the Patent Office's filing system. The most visible part of the system are the custom-built, two-screen terminals that let examiners cruise through the Patent Office's more than five million issued patents. Group 2300's examiners are among the first at the Patent Office to get access to the terminals. Other terminals are located on the first floor of the Patent Office's building and in various repositories around the country.

According to the Patent Office, the Automated Patent System is one of the world's largest online databases. The computers that run it have more than 400 optical drives, each with 6.4 Gbytes of online storage. The storage requirements are so immense because the system includes a complete photograph of every patent going back to patent Number 1; it also stores the text for every patent going back to the late-1970s. The examiner can view each patent on the screen or click a button and have a copy printed on a nearby high-resolution printer.

Scouting out old patents with the Automated Patent System is a lot like information-surfing on the Internet with gopher and Mosaic. For example, to do a search on Steir's hairstyle simulation patent, it takes Herndon just a few keystrokes to pop up a list of all the patents in a class 395, "Computer Graphic Applications," subclass 135, "merge overlay." With a few more keystrokes she can broaden her search to include subclass 134, "clipping"; subclass 133, "object positioning"; and subclass 137, "rotation."

If the Automated Patent System doesn't turn up the invention, the next stop is down the hall at the Electronic Information Center. Here a staff of two trained searchers have access to more than 900 online databases, including major American vendors like Dialog, STN International, and Mead Data Central, as well as European vendors like Data-star and Questel. The Electronic Information Center has a storage tower equipped with 56 CD-ROM drives. Searching can be expensive, says Elaine Hickey, the center's librarian, but cost doesn't really matter: "If they need it, we get it."

Unfortunately, says Hickey, there's a huge chunk of the prior art that the Electronic Information Center can't touch: books. Few publishers have allowed the full text of their books to be incorporated into online databases. Even seminal textbooks on computer graphics and algorithms are missing. That's not a problem in fields like chemistry or biology, where textbooks almost never print information that hasn't first appeared in journal articles. But in computer science, where academics commonly save up their best work for big textbook projects, it's much easier for an examiner to erroneously issue a patent.

That's probably what happened last year, when the Patent Office awarded Patent Number 5,241,671 to Compton's New Media.

The Compton's patent contained 41 claims that broadly covered any multimedia database allowing users to simultaneously search for text, graphics, and sounds – basic features found in virtually every multimedia product on the market. The Patent Office granted the patent on August 31, 1993, but it went unnoticed until mid-November, when Compton's made the unusual move of announcing its patent at the computer industry's largest trade show, Comdex, along with a veiled threat to sue any multimedia publisher that wouldn't either sell its products through Compton's or pay Compton's royalties for a license to the patent. Compton's president, Stanley Frank, stated it smugly for the press: "We invented multimedia."

The denizens of the multimedia industry thought otherwise. In dozens of newspapers around the country, experts asserted that Compton's patent was clearly invalid, because the techniques that it described were widely used before the patent's October 26, 1989, filing date. Rob Lippincott, the president of the Multimedia Industry Association, called the patent "a 41-count snow job." Even Commissioner Lehman thought that something was wrong.

"They went to a trade show and told everybody about it. They said they were going to sue everyone," says Lehman, who first learned of the Compton's patent from reading an article in the *San Jose Mercury News.*

"I try not to be a bureaucrat," he adds. "The traditional bureaucratic response would be to stick your head in the mud and not pay attention to what anybody thinks." Instead, Lehman called up Gerald Goldberg, 140▶

◀ 109 director of Group 2300, to find out what had happened.

Like Lehman, Goldberg had learned about the Compton's patent from reading the article in the *Mercury News.* "We pulled the patent file and I took a look at it," recalls Goldberg. "I spoke with the examiner. We felt the examiner had done an adequate job." In this particular patent application, says Goldberg, the Compton's lawyer had included an extensive collection of prior art citations – none of which described exactly what the Compton's patent claimed to have invented. Without a piece of paper that proved that the invention on the Compton's application was not new, the examiner had no choice but to award Compton's the patent.

At a staff meeting that day, Goldberg asked if anybody had ideas about other areas where prior art might be found that could invalidate the Compton's patent. Somebody mentioned Danny Goodman's 1987 book *The Complete HyperCard Handbook*, which described how to build the same sort of database for which Compton's had claimed a patent.

Finally, on December 14, 1993, Commissioner Lehman made the unusual move of requesting that Group 2300 reexamine the Compton's patent in light of the "new" prior art that had suddenly come to light. Three months later, the Patent Office announced that the patent application had been rejected based on "new" prior art that had come to light.

Under the law, the Patent Office's action means that the Compton's patent is now back in "prosecution," the long, expensive fight between patent attorneys and Patent Office examiners. But things aren't safe for the multimedia industry yet. Patent rejections happen all the time. They're part of the process. Even "final rejections" are common: a final rejection simply means that the patent examiner has given up on the patent, but the inventor still has the option of appealing to the Patent Office's board of appeals, then to Federal Circuit Court, and finally to the Supreme Court.

The Compton's patent is more than just a very unusual, very big mistake. It demonstrates a fundamental flaw in applying the patent system to software: examiners simply don't have access to the prior art that they are required by law to check.

Indeed, all of the Patent Office's electronic systems haven't made a dent in Lehman's "folk art" problem. That's because every program that's ever been sold, every piece of shareware that's ever been posted to a bulletin board system, and every technical report produced by every first-, second-, and third-rate computer science department in the world counts as prior art. Better databases of the prior art will certainly give examiners more tools for ruling against software patents, but they cannot, in principle, prevent the Patent Office from occasionally issuing patents that turn out to be invalid.

Although many are quick to criticize the Patent Office, few offer workable solutions to the problem of software patents.

Here's the Rub

The real problem with software patents may not be the invalid ones at all but the valid ones – the patents that really mark advances in the state-of-the-art, patents on fundamentally new inventions.

Both fans and critics of software patents have a tendency to quote from the US Constitution, which states that the purpose of the patent system is "to promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries." Is this in fact what patents are doing for the world of computer science?

One of the best-known patents in the computer industry is patent Number 4,405,829, "Cryptographic Communications System and Method," the patent on RSA public-key cryptography. This patent, which expires on September 20, 2000, covers every implementation of RSA encryption in the United States. Because the algorithm is patented, it is a violation of US law for a company to write its own implementation of the RSA algorithm and use it without a license from Public Key Partners, the company that has an exclusive license to the patent from the Massachusetts Institute of Technology, the university where the algorithm was developed. It's even illegal for a public-spirited citizen to write his or her own implementation of RSA and give it away.

Patent theory holds that patents are economically justified when the cost of developing a new product is high, but the cost of competitors' ripping off the product's essence is low. One of the standard examples is the field of pharmaceuticals, where it can cost hundreds of millions of dollars to develop, test, and bring a new drug to market, yet the actual chemical itself can be produced very cheaply. Patents, the argument goes, give drug makers an incentive to develop new cures.

Patents may be less justified in the world of software, where major application programs can be developed by a few people working in somebody's living room over just a few months. While copyright protection for a program prevents software pirates from handing out illegal copies, patent protection prevents other developers from writing their own version of a program and trying to sell it – or even give it away.

In the case of the RSA patent, one person who has violated the algorithmic monopoly is Phil Zimmerman, creator of Pretty Good Privacy (PGP), a program that implements the RSA cryptosystem (see "Crypto Rebels," *Wired* 1.2, page 54). Zimmerman says the RSA patents have put a stranglehold on the widespread public use of cryptography – a stranglehold he's tried to break by making PGP freely available over the Internet.

According to Jim Bidzos, president of both RSA Data Security and Public Key Partners, Zimmerman is nothing more than a dishonest profiteer, encouraging people to violate US patents while he builds up his own reputation as a cryptography consultant. But even without patent protection, the RSA algorithm would still have been developed and made publicly available. That is because the idea of applying for a patent on RSA didn't occur to MIT Professor Ronald Rivest until three months after he had published the first technical description of the encryption system.

Solving the Problems

Although many are quick to criticize the Patent Office, few offer workable solutions to the problem of software patents.

Hardly anyone advocates abandoning software patents entirely. "The more patents are issued, the more you set up an expectation that this is an ongoing field of patentable subject matter, and the harder it is to essentially tell the next person in line, 'You don't get your software patent,' " says Professor Merges.

For example, while many members of the Interactive Multimedia Association are opposed to "bad" software patents, the group's largest founders are the companies most actively pursuing patents of their own. Hence the association's measured stance on patents: it opposes 142 ▶

◄ 140 patents that would affect the entire industry but favors less expansive individual patents.

One group committed to the abolition of all software patents is The League For Programming Freedom, which contends that Congress could either modify the law so that the Patent Office is forbidden to issue software patents or rewrite the law so that a patent cannot be infringed merely by a computer running a program.

A less drastic solution is favored by John Preston, who headed MIT's patent office for nearly ten years. Preston would set up a system of mandatory licensing, in which patent holders would be legally required to license any software patent at a preset fixed fee. Such a scheme would prevent large companies from cross-licensing patents among themselves and then using patent pools to shut out upstart competitors, a practice common in other industries. "After a technology has been introduced, it should be made available to everyone," says MIT's Preston.

For a patent system to be fair and effective, patent searches should be cheap and easy. But in reality they are expensive and difficult – and they are likely to stay that way. Why? It's simple: costs. The Patent Office has to support itself entirely on user fees – the charges for patent applications, patent issuances, and patent maintenance fees during the patent's life. To further help make ends meet, the office generates additional revenue by selling its taxpayer-created government information to private companies, who then resell the information at a substantial profit. For example, the Patent Office's "full text file" will deliver to you the full text of every newly issued US patent (approximately 2,000 patents each week filling more than 80 Mbytes) for just $1,785 a year. Companies like Dialog drop this information into their databases and sell access to customers at a cost of $150 per hour or more.

Gregory Aharonian, of the Internet Patent News Service, is trying to put together a coalition of law firms that would in turn make the Patent Office's database of patents freely available over the Internet. "In January, I floated a proposal on the Internet to raise $150,000, $80,000 of which would go to buy 80 Gbytes of disk drive, $40,000 of which would buy magnetic tapes from the Patent Office, and $30,000 of which would be applied to the Internet connection and the workstation to parcel out the connection. I got a lot of interest, but no one came forward with the check."

A group trying to solve the prior-art problem is the Software Patent Institute in Ann Arbor, Michigan. A project of the Industrial Technology Institute, the Software Patent Institute has been building a database of computer science folklore – techniques that are in use, but not widely published – so that patent examiners can stop issuing bad patents. "What we are providing is not all prior art, but the prior art that is least readily available elsewhere," says Roland Cole, the Software Patent Institute's executive director. Cole says that the institute's database consists of the full text of software-related items from old computer magazines, old computer manuals, standard computer science textbooks, and IBM's *Technical Disclosure Bulletin*. The Software Patent Institute also invites programmers who have invented things to write them down and send them for inclusion in the database.

So far, however, the institute is having problems getting the information that it needs to make its database useful. Publishers have been unwilling to have their books included in the Software Patent Institute's database, because they think the databases will cost them sales.

Cole admits that cost, as well as the difficulty of persuading people to donate material and of processing it into the database, hinders progress.

Another approach being considered, which is backed by Commissioner Lehman, is the Patent Term and Publication Reform Act, sponsored by Senator Dennis DeConcini (D-Arizona). This bill requires the Patent Office to publish applications-in-progress eighteen months before a patent is granted, which would make it possible for industry watchdog groups to bring prior art to the attention of the patent examiner before a patent is granted. The bill would also change the term of patents from the current seventeen years following the date of issuance to twenty years from the date of filing.

The most important thing about DeConcini's bill, say its supporters, is that it would eliminate so-called "submarine patents": patents that are filed early on in the development of a new technology, but stay in prosecution for up to twenty years, and then bite the companies that have built products based on the "unpatented" process. That's what happened with the REFAC patent on natural order recalculation, which was filed in August 1970 but not granted until 1983.

Nevertheless, even if the Patent Office stopped issuing "invalid" patents, and even if mandatory licensing became a reality, patents would still fundamentally change the computer industry. Free software, shareware, and small two-person start-up companies are widespread in today's computer industry for one main reason: software, once it is created, can be copied virtually without cost.

Software patents are changing that scenario. A single program can infringe a dozen or more patents – unbeknownst to the programmer. Combined with the high cost of infringement suits and the presumption by courts that patents are almost always valid, this could spell the death of the industry's small players, locking the computer industry up in the hands of giants like Microsoft, Novell, and Lotus. ■ ■ ■

@ W I R E D

Additional articles and information on patents are available via WIRED Online.
► America Online: Type keyword *Wired,* then click on the WIRED Extras icon and select 2.07 Patents-Extras for downloading;
► Gopher: Gopher over to *gopher.wired.com* in *Etext/2.07/patents.extras;*
► World Wide Web: Point your WWW client to *http://www.wired.com/Etext/2.07/patents .extras.html* (Links to the additional information will temporarily exist on the top level of WIRED Online's Gopher and WWW sites);
► E-mail: Send a message to *infodroid@ wired.com* with the words *get 2.07/features/ patents.extras* contained in the message body.

Simson L. Garfinkel will appear in the WIRED Auditorium on America Online to discuss this article on Wednesday, July 6, from 9 to 10 p.m. EST. From AOL, type the keyword *Wired* and click on the WIRED Auditorium icon.

If you have access to the Internet and would like a copy of the US Patent Office's hearings on software patents, *ftp* to the site *comments.uspto.gov*, logging in as *anonymous* with your own name as the password. In the directory *pub/software_hearings* you'll find the full transcripts of both the San Jose and the Virginia sessions. For information about the Internet Patent News Service mailing list, e-mail *patents@world.std.com.*