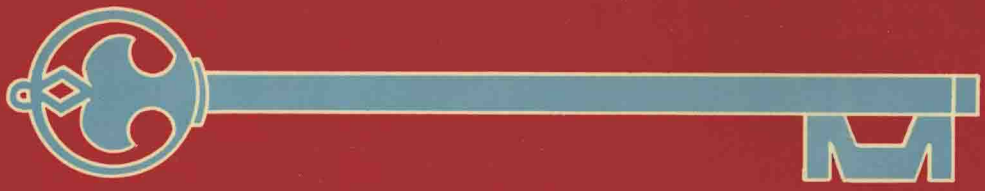
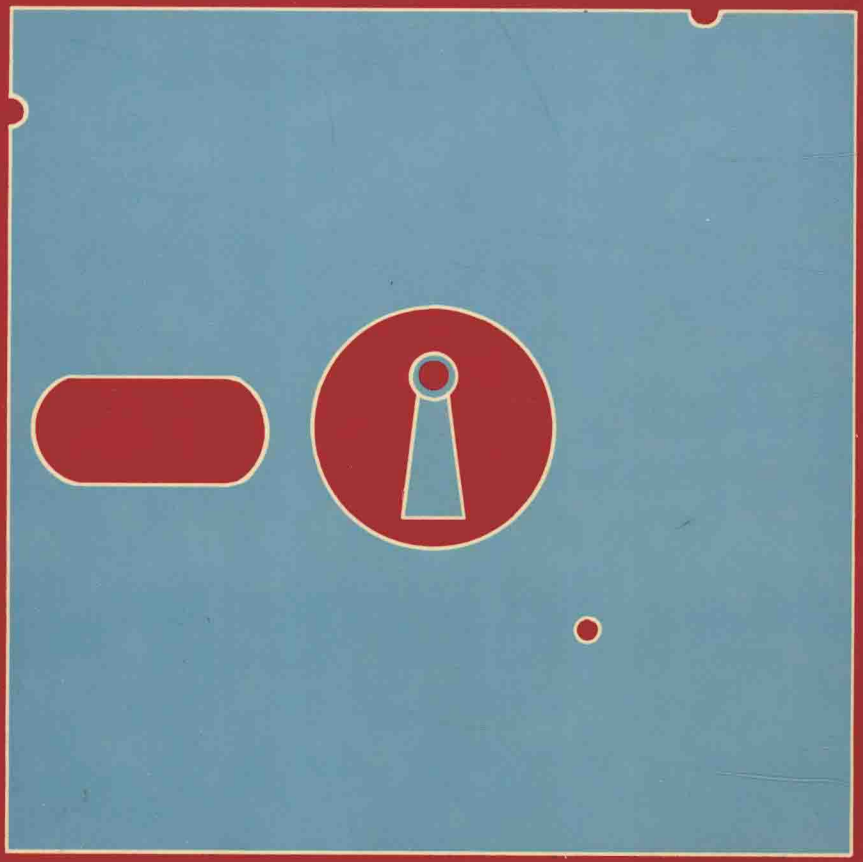
A SOFTWARE LAW PRIMER



FREDERIC WILLIAM NEITZKE



A SOFTWARE LAW PRIMER

FREDERIC WILLIAM NEITZKE


VAN NOSTRAND REINHOLD COMPANY
New York

Library of Congress Catalog Card Number: 83-23508
ISBN: 0-442-26866-1

Manufactured in the United States of America

Published by Van Nostrand Reinhold Company Inc.
115 Fifth Avenue
New York, New York 10003

Van Nostrand Reinhold Company Limited
Molly Millars Lane
Wokingham, Berkshire RG11 2PY, England

Van Nostrand Reinhold
480 La Trobe Street
Melbourne, Victoria 3000, Australia

Macmillan of Canada
Division of Canada Publishing Corporation
164 Commander Boulevard
Agincourt, Ontario M1S 3C7, Canada

15 14 13 12 11 10 9 8 7 6 5 4 3 2

Library of Congress Cataloging in Publication Data

Neitzke, Frederic William.
A software law primer.

Includes index.
1. Copyright–Computer programs–United States.
2. Computer programs–Patents. I. Title.
KF3024.C6N44 1984 346.7304'82 83-23508
ISBN 0-442-26866-1 347.306482

A SOFTWARE LAW PRIMER

Preface

This is a book for the layman with an interest in the legal issues inherent in the creation and exploitation of computer programs. Its makeup is different from that of most books of this sort, in that I have included many actual quotations from court opinions, rather than just stating what the court decided. I did this because many of those with an interest in these topics prefer to deal with source materials whenever possible. But don't be lulled into a false sense of security. The materials in this book barely scratch the surface of the surface; it isn't a do-it-yourself handbook. It does, however, provide sufficient information to alert you to situations requiring assistance of counsel, and to help you work with that counsel more effectively. As ever, recognizing a problem early is at least half the battle.

Frederic W. Neitzke
Melbourne, Florida

Contents

1
Patents

SOME SOFTWARE IS PATENTABLE

Justice Stevens of the Supreme Court recently stated that "the cases considering the patentability of program-related inventions do not establish rules that enable a conscientious patent lawyer to determine which, if any, program-related inventions will be patentable." I've started off with this quotation in hopes of dispelling a persistent rumor that software isn't patentable. The good news is that some software is indeed patentable. The bad news is that it is almost impossible to identify the characteristics of such programs.

In spite of these hurdles, it's worthwhile to explore the possibility of patent protection for a program, because patents offer by far the best legal protection for software. For example, a patented program can be widely distributed without increasing the likelihood that rights will be lost, because a patent owner can prevent the unauthorized use of his program by anybody, no matter how that person came into possession of the program. This is true even if the unauthorized user developed the program independently without any knowledge of the existence of the patented program. And the owner of a patented program can exploit it commercially, using techniques that would violate the antitrust laws but for the patent. For example, he could license one competitor to use his program only in Texas and another to use it only in California, which could be an illegal division of markets if the licensed software were not patented.

The materials which follow provide an overview of the United States patent system, and a discussion of the leading cases dealing with the patenting of software.

THE U.S. PATENT LAWS

Constitutional Provision

The U.S. Constitution provides that "Congress shall have the Power . . . 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."

Pursuant to this provision, Congress has enacted several patent laws including the present Patent Act, which went into effect in 1952. Basically, a U.S. patent gives an inventor the right to exclude others from making, using, or selling his invention in the United States for a period of years, in return for a full disclosure of the invention. This limited monopoly is designed to encourage full disclosure of inventions, so that after the monopoly has expired the public has the benefit and the free use of the invention. The limited monopoly offers an incentive to the inventor.

Section 101

The Patent Act (*Section 101*) defines patentable inventions as any new and useful process, machine, manufacture, or composition of matter. This is a necessary but not sufficient requirement, since patentable inventions must comply with other code requirements. Remarkably, this seemingly all-encompassing definition has represented the major obstacle to the patenting of software. The Supreme Court has held that certain programs do not constitute patentable subject matter within the scope of *Section 101*. We will defer consideration of these cases until we have examined other significant sections of the Patent Act.

Section 102

Section 102 of the Patent Act spells out certain conditions which prevent the issuance of a valid patent. Portions of this section may be of interest to programmers, since they represent a compact treatment of

a number of variables; they are reproduced below. Note that the significant considerations are:

1. When the *invention* was made [*102(a)*];
2. When the *patent application* was filed [*102(b)*];
3. *Where* certain events occur (in the United States or abroad); and
4. Who filed the application [*102(f)*].

35 U.S.C. 102

A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent, or

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of the application for patent in the United States, or

(c) he has abandoned the invention, or . . .

(f) he did not himself invent the subject matter sought to be patented, or

(g) before the applicant's invention thereof, the invention was made in this country by another who had not abandoned, suppressed, or concealed it.

In my experience, the following situations are most commonly the reason that a patent cannot be obtained:

1. The invention has already been patented.

2. The invention has been offered for sale or been in public use for more than a year before the inventor seeks a patent. This frequently occurs when the inventor is not familiar with the Patent Act and seeks to test-market his invention before investing in patent protection.

3. The invention has been described in a publication more than one year before a patent is sought. In this case, too, the inventor is usually ignorant of patent law.

SECTION 103

If an invention represents patentable subject matter under *Section 101* and doesn't run afoul of the procedural requirements of *Section*

102, it still has one other hurdle to clear which is set out in *Section 103.* The invention must not be obvious in view of the existence of similar inventions, whether or not the other inventions are patented. This is, at best, a subjective determination. *Section 103* reads as follows:

35 U.S.C. 103

> A patent may not be obtained . . . if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.

Section 103 usually first comes into play when a patent application is acted upon by a patent examiner. He searches the Patent Office files for similar inventions disclosed in earlier patents or publications. If he locates any "prior art" that in his opinion makes the invention obvious, he rejects the claims of the patent anticipated by the prior art. If the applicant believes that his invention is not obvious in view of the prior art found by the examiner, he engages the examiner in a stylized dialogue, seeking to convince him that the claims, as originally filed or amended, should be allowed.

The examiner uses the results of his search for prior art to evaluate the obviousness of applicant's invention. Since the examiner has a finite amount of time to devote to the search, he may not be able to locate all the relevant prior art. In fact, since the Patent Office files are notoriously incomplete, the examiner probably does not have access to all pertinent prior art.

Consideration of the obviousness of an invention does not end when a patent issues. If the patent owner sues an infringer, the defendant will almost certainly seek to invalidate the patent by finding prior art not considered by the patent examiner, or by quarreling with the decision of the examiner with regard to the prior art that he did consider. It is then up to a judge or jury to evaluate the obviousness of the invention.

Fortunately, we do not need to review the intricate rules developed by the courts and the Patent Office regarding obviousness under *Section 103,* because there have been very few decisions pertaining to

the *obviousness* of software. Judicial pronouncements regarding the patentability of software have centered on the issue of whether or not software constitutes patentable subject matter under *Section 101.*

SOFTWARE PATENTABILITY CASES

ENIAC, the first electronic general-purpose computer, was built in 1946. Reprogramming ENIAC required manual rewiring. MANIAC I, the first digital computer capable of operating upon stored programs, was completed in 1952. Twenty years later the Supreme Court first addressed the issue of the patentability of computer programs in *Gottschalk* v. *Benson.* In the decade since *Benson* there have been a number of decisions pertaining to the patentability of software. The principal players in this evolving area of the law have been:

1. *The Patent Office,* which opposes the patenting of software. This opposition is couched in terms of principle (software is not patentable because of legal axioms that predate ENIAC), but reflects the reality that the Patent Office is not equipped to deal with an avalanche of software applications, having neither the personnel nor records to do so.
2. *The Court of Customs and Patent Appeals (CCPA),* which has championed the cause of software patentability.
3. *The Supreme Court,* which has frequently overruled the CCPA but has not established any clear guidelines regarding the patenting of software. The Court has tried to prod Congress into settling the matter by legislation, but to date no statutes have been passed.

In 1968, the Patent Office issued software patentability guidelines which denied patent protection to computer programs. The programmed computer could be a *component* of a patentable process, but software alone was not patentable. The CCPA then set about eliminating the legal precedents on which the Patent Office guidelines were based.

GOTTSCHALK v. *BENSON*

Gary Benson and Arthur Tabbot filed a patent application which claimed a method for converting binary-coded decimal numerals to binary numbers. The

claims were not limited to any particular technology, apparatus, or end use. They covered any use of the claimed method in a general-purpose digital computer. The claims in issue read as follows:

Claim 8

The method of converting signals from binary coded decimal form into binary which comprises the steps of

(1) storing the binary coded decimal signals in a re-entrant shift register,

(2) shifting the signals to the right by at least three places, until there is a binary "1" in the second position of said register,

(3) masking out said binary "1" in said second position of said register,

(4) adding a binary "1" to the first position of said register,

(5) shifting the signals to the left by two positions,

(6) adding a "1" to said first position, and

(7) shifting the signals to the right by at least three positions in preparation for succeeding binary "1" in the second position of said register.

Claim 13

A data processing method for converting binary coded decimal number representations into binary number representations comprising the steps of

(1) testing each binary digit position "1," beginning with the least significant binary digit position, of the most significant decimal digit representation for a binary "0" or a binary "1";

(2) if a binary "0" is detected, repeating step (1) for the next least significant binary digit position of said most significant decimal digit representation;

(3) if a binary "1" is detected, adding a binary "1" at the (i + 1)th and (i + 3)th least significant binary digit positions of the next lesser significant decimal digit representation, and repeating step (1) for the next least significant binary digit position of said most significant decimal digit representation;

(4) upon exhausting the binary digit positions of said most significant decimal digit representation, repeating steps (1) through (3) for the next lesser significant decimal digit representation as modified by the previous execution of steps (1) through (3); and

(5) repeating steps (1) through (4) until the second least significant decimal digit representation has been so processed.

These claims were rejected by the Patent Office as not being drawn to patentable subject matter under *Section 101*. The CCPA reversed, holding that the claims *were* drawn to patentable subject matter. Specifically, the court held that claim 8 was drawn to a method to be practiced on a particular apparatus, a re-entrant shift register, and that it was within the "machine" and "process" categories of *Section 101*. Claim 13, which was not drawn to a specific apparatus, was held to constitute a "process" under *Section 101*.

In a unanimous opinion, the Supreme Court reversed the CCPA. The Supreme Court defined an algorithm as a procedure for solving a given type of mathematical problem, and held that

> the mathematical formula involved here has no substantial practical application except in connection with a digital computer which means that *if the judgment below is affirmed, the patent would wholly preempt the mathematical formula and in practical effect would be a patent on the algorithm itself.* . . . If these programs are to be patentable, considerable problems are raised which only committees of Congress can manage [emphasis added].

The *Benson* Court said that it was *not* saying that software is not patentable, but its murky decision did not provide much guidance as to what software programs might be patentable. The Supreme Court's next foray into the software thicket took place in 1977.

After *Benson,* the CCPA continued to reverse the Patent Office rejection of certain software patent applications, developing a two-step analysis for software claims. This two-step test involved first determining if the claim recited an algorithm in the *Benson* sense. If no algorithm was found, the claim was acceptable under *Section 101.* If an algorithm was found, the patent was analyzed to determine whether the claim preempted the algorithm. If the claim did not preempt the algorithm – that is, if the patent claim covered only a specific use for the algorithm, leaving it otherwise free for anyone to use – then the claim was directed to patentable subject matter.

These tests led to even more confusion. The Patent Office adopted a broad definition of an algorithm, in one instance using a definition taken from a computer dictionary which read as follows: "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."

The CCPA considered this definition overly broad, since it was not limited to expressions in mathematical terms but rather included expressions in natural language. (The CCPA also criticized the Supreme Court's definition of an algorithm.) The CCPA held that an algorithm, by itself, could be patentable subject matter. But in *Parker* v. *Flook* the Supreme Court disagreed.

PARKER v. *FLOOK*

Dale R. Flook applied for a patent on a method for updating alarm limits during a catalytic conversion process. The only novel feature of the method was a mathematical formula. The patent claim covered a broad range of potential uses of the method, but did not cover every conceivable application of the formula. A representative claim reads as follows: