Harvard Journal of Law & Technology
Volume 9, Number 1 Winter 1996

SOFTWARE PATENTS

By Gregory A. Stobbs.[1]
New York, N.Y.: John Wiley & Sons, Inc. 1995.
Pp. 623. $125.00 (hard).

The decision by IBM and its fellow computer hardware manufacturers to unbundle their software starting in 1970 unleashed a torrent of products which has since become the software industry.[2] In the immediate aftermath of this dramatic change in the marketplace, legal protection for software inventions was quite weak. In 1980, Congress brought software under the umbrella of copyright protection, but the status of software patents remained unclear. Few software patent applications were filed in the wake of *Gottschalk v. Benson*[3] and *Parker v. Flook*,[4] decisions read by many as hostile towards software patentability. The Supreme Court alleviated some of this confusion by noting in *Diamond v. Chakrabarty* that patentable subject matter "include[s] anything under the sun that is made by man,"[5] implicitly acknowledging the inclusion of computer software within the patent regime. However, patent protection for software programs has been limited by other factors. First, the software industry's initial reluctance to apply for patent protection for its programs has rendered the Patent Office's prior art collection woefully incomplete.[6] Furthermore, many patent examiners, traditionally engineers by trade, do not possess the expertise necessary to understand software patent claims. These problems have provoked calls for reform from the software industry, which has complained that a Patent Office which does not properly understand its inventions has been issuing software patents of dubious validity.

Gregory Stobbs's book is designed to make sense of this potentially confusing area of law. While it discusses the rationales behind the relevant statutes and cases, *Software Patents* is first and foremost a

---

1. Gregory A. Stobbs is a principal in the patent firm of Harness, Dickey & Pierce, P.L.C., Troy, Michigan.

2. Before 1970, when hardware manufacturers routinely included or "bundled" software with their hardware products, the market for independent software was virtually nonexistent. Unbundling created such a market, and the growth of the resultant software industry has been explosive.

3. 409 U.S. 63 (1972).

4. 437 U.S. 584 (1978).

5. 447 U.S. 303, 309-10 (1980).

6. The Patent Office must often glean the prior art by searching industry literature, a process which is notoriously time-consuming and susceptible to error.

practitioner's handbook. Hence, the depth of Stobbs's legal analysis is not as extensive as that of a hornbook or treatise. Nevertheless, it more than adequately supports the main purpose of Stobbs's exposition: instructing the lawyer or industry practitioner how to use the patent system to protect software inventions. *Software Patents* assumes no particular background on the part of the reader; entire chapters are devoted to basic principles of law and computer science. As a result, the book could theoretically be read and understood by a novice in either field, though as a practical matter anyone relying on this handbook should have expertise in at least one of the two.

Chapter 1 begins with a description of the historical foundation upon which software patent protection has been built. Stobbs outlines the antecedents of American patent law, describing the patent monopoly's first appearance in classical antiquity, followed by the royal issuance of "letters patent" in fourteenth-century England to encourage the importation of new products. Stobbs then traces the development of the American patent system through the Patent Acts of 1790, 1793, 1836, and 1952, the creation of CONTU,[7] and the 1994 Software Patent Hearings.[8] Finally, he gives a short history of the computer industry, from the waterwheel-driven organs of the 1760s programmed with punched cylinders to modern digital computers. He also provides brief sketches of four Supreme Court cases bearing on the question of software patentability.[9]

The next two chapters address the rudiments of software design and patent practice for novices in each field. Stobbs's "software primer," an introduction to basic software concepts, is a necessarily cursory survey of a wide range of topics, including approaches to software design, software languages, communications protocols, client-server architecture, and technical aspects of Microsoft Windows. His overview of patent law in Chapter 3 is similarly ambitious, and includes a general discussion of the legal system, followed by relevant aspects of property law, the Patent Act of 1952 and its regulations, and the *Manual of Patent Examining Procedure*, the Patent Office's guidebook for examiners. The

---

7. In 1975, Congress formed the National Commission on New Technological Uses of Copyrighted Works, or CONTU, to assess the adequacy of copyright law in protecting computer-based information systems and photocopying technology.

8. In early 1994, the Patent Office invited anyone whose work involved software patent protection to testify at a series of public hearings. Prominent members of both the patent bar and the computer industry expressed their views on the policy implications of software patents and the Patent Office's ability to issue them.

9. *See* Diamond v. Diehr, 450 U.S. 175 (1981); Diamond v. Chakrabarty, 447 U.S. 303 (1980); Parker v. Flook, 437 U.S. 584 (1978); Gottschalk v. Benson, 409 U.S. 63 (1972).

chapter's overview of the legal system concludes with a description of the various aspects of a sample patent.

The subsequent four chapters form the heart of *Software Patents*: a comprehensive examination of the process of searching prior art, "finding the invention" (a technical term for assessing sufficient differentiation from the prior art), and drafting patent claims. After a brief section on the significance of prior art searches for patent prosecution, Stobbs provides a detailed analysis of the somewhat arcane classification system employed by the Patent Office for organizing and recording its patents. He then explains how to conduct prior art searches using sources such as CASSIS (the Patent Office's CD-ROM collection), LEXIS/NEXIS, and the Software Patent Institute ("SPI") database.[10] Next, Chapter 5 discusses the rationale behind the patent specification requirement[11] and analyzes the specific regulations imposed by the governing statute. It then outlines the mechanics of specification drafting, explaining how to describe software inventions and write patent specifications with an eye towards intended readers. Chapter 6 is devoted to patent drawings: why they are helpful, when the patent law requires them, and what they should contain, as well as descriptions and examples of the various kinds of notation currently used by practitioners, such as flowcharts and pseudocode. Chapter 7 deals with claim drafting and sets forth the purpose of the claim requirement. It also discusses the various types of rejections which can arise under 35 U.S.C. § 101 (for non-patentable subject matter) and § 112 (for improperly drafted claims),[12] and offers tips for avoiding these pitfalls. The chapter concludes with detailed instructions for determining the software's particular invention and drafting its patent claim.

Stobbs organizes the next chapter as a step-by-step description of the patent examination process. He traces the path of an application through the Patent Office's complex routing system, from its arrival in the mail room, through various classification procedures, to its appearance on the examiner's desk. Stobbs next guides the reader through the filing process, highlighting common mistakes and explaining the procedures for seeking expedited prosecution. An important step is the applicant's duty to aid the Patent Office by disclosing all information "material to patentability" under 37 C.F.R. § 1.56, including the prior art references from foreign patent offices.[13]

---

10. The SPI is a non-profit organization dedicated to improving the access of both the Patent Office and the general public to prior art information.

11. 35 U.S.C. § 112 (1988).

12. 35 U.S.C. §§ 101, 112 (1988).

13. 37 C.F.R. § 1.56(a) (1992).

In Chapter 9, Stobbs treats the potentially troublesome problem of what constitutes patentable subject matter under § 101. He begins by outlining several important Supreme Court patent decisions since the nineteenth century, including *O'Reilly v. Morse*,[14] the landmark case in which the Court ruled invalid for overbreadth a claim that would have granted Samuel Morse a patent on using electromagnetic current to send encoded information. Next, the author devotes special attention to five recent decisions of particular importance for software patents: the four Supreme Court decisions discussed briefly in the first chapter and the *en banc* decision of the Federal Circuit in *In re Alappat*.[15] To reconcile these opinions, Stobbs offers his interpretation of the common bond linking these decisions: the touchstone of patentability is whether humanity can exert control over the purported invention. The chapter concludes with a description of the *Freeman-Walter-Abele* two-step test for patentability.[16]

In Chapter 10, Stobbs gives an overview of the European and Japanese patent systems. Finally, Chapter 11 lists examples of existing patents, each describing the problem addressed by the invention and providing a sample claim. Appendices to the book include a full transcript of the 1994 Software Patent Hearings, the software classification system used by the Patent Office, and an 1813 letter in which Thomas Jefferson, America's first patent examiner, elucidated his views on patentability.

In general, the material in *Software Patents* is well-presented. Stobbs's exposition is lucid, and his logical organization should present practitioners with little difficulty in locating specific material. The user-friendly division of each chapter into many short titled sections creates a detailed table of contents and easy access to particular subheadings. To lighten the tone of a potentially dry subject, amusing commentary graces the text in several places. For example, Stobbs notes that in the

---

14. 56 U.S. 62 (1854). In holding that Morse could not patent the motive power of electric current in the absence of specific machinery to create such power, the Supreme Court's decision foreshadowed contemporary debate on the patentability of abstract processes.

15. 33 F.3d 1526, 1545 (Fed. Cir. 1994) (holding that a computer operating pursuant to software may represent patentable subject matter, as long as claimed subject matter meets all other statutory patentability claims).

16. This test comes from three decisions of the Court of Customs and Patent Appeals: *In re* Freeman, 573 F.2d 1237 (C.C.P.A. 1978); *In re* Walter, 618 F.2d 758 (C.C.P.A. 1980); and *In re* Abele, 684 F.2d 902 (C.C.P.A. 1982). The first step in the *Freeman-Walter-Abele* test is to determine whether a mathematical algorithm is recited directly or indirectly in the claim. If so, the second step is to determine whether the claimed invention as a whole consists of nothing more than the algorithm itself. A claimed invention found to be no more than a mathematical algorithm is nonstatutory subject matter under § 101. *See In re* Schrader, 22 F.3d 290 (Fed. Cir. 1994).

underground walkway system leading to the Patent Office, various shops may divert the attention of one's fellow travelers (p. 101). The inclusion of such passages makes for pleasant reading and provides a welcome change from the traditionally humorless tenor of most legal volumes.

The book's organization is not without its flaws, however. Most noticeably lacking in the presentation is a general overview of the statutory requirements for a valid patent. Although several chapters analyze individual requirements in some detail, the lack of a single comprehensive treatment may disappoint or confuse readers unfamiliar with the intricacies of the patent system. For example, the discussion of the patent specification in Chapter 5 clearly assumes knowledge of what a patent claim is, even though this subject is first addressed in Chapter 7. A reproduction of additional sections of the Patent Act addressing these requirements might have avoided this problem. In addition, there is some repetition of material — § 1.2 and § 3.13, for example, are largely duplicative, as are § 1.15 and § 8.17 — though the information arguably merits inclusion in each chapter.

Substantively, the primary strengths of *Software Patents* are its thorough treatment of an important area of law and its accessibility to a broad range of readers. In the inevitable trade-off between breadth and depth, this ambitious aggregation of topics comes down squarely on the side of the former. For instance, while the presentation in Chapter 2 of only the broadest outline of software concepts is informative, anyone whose practice demands a specialized working knowledge of this material will need to consult computer science textbooks or technical journals. Nevertheless, the "how-to" portions of the book are well-written and will doubtless provide practitioners with able guidance. The analysis of the patent statute and case law will likewise be of assistance, not just to the practitioner, but to anyone perplexed by this changing area of law. For more ambitious readers, sifting through the generally cogent but often repetitive Software Patent Hearing testimony in Appendix A will illuminate the policy environment likely to shape the law of software patents in the near future.

One substantive weakness of *Software Patents* is that while it treats § 101 of the Patent Act (patentable material) in sufficient depth, it largely ignores §§ 102-03 (novelty and nonobviousness). Even though § 101 does present the most vexing questions to a potential applicant, the other two sections deserve further analysis. Also, the book does not squarely address the question of how, given a particular software design and the prior art, one confirms that there is indeed an invention that is legally distinguishable from the prior art and also a nonobvious addition to it. This question is important both for applying for new patents and for avoiding infringement of existing patent rights. Although there is one

reference to the § 102 question in Chapter 8, it appears in neither the Table of Contents nor the Index.

Despite these few structural flaws and substantive omissions, *Software Patents* ably fulfills its tripartite goal of providing a foundational background which more experienced readers can skip, a good summary and analysis of the relevant statutes and decisions in a confusing and dynamic area of law, and, most importantly, enough practical guidance to be of considerable help to the patent practitioner. Moreover, it appears that *Software Patents* is among the first books devoted exclusively to its subject matter; as such, it should be a welcome addition to the library of any practitioner whose work involves this increasingly important area of law.

Robert L. Bocchino Jr.