The Long History of Software Patenting in the United States

Gerardo Con Diaz

Comments by Lisa Larrimore Ouellette
Hoover IP² – May 18, 2018
Summary

- Rich description of effective patenting of software in 1950s by industrial research labs using strategy of patenting the computer system embodying a program.
  - 1951 – Bell Labs obtains Hamming & Holbrook patent, *Error-Detecting and Correcting System*, disclosing circuit arrangement, not program logic.
  - Followed by BTM, RCA, IBM, etc.
- Decisions by CCPA (CAFC predecessor) in 1960s validated this drafting technique.
It is worth noting that patents modeled after Hamming and Holbrook’s blend into the broader computer patenting landscape of the 1950s. The patents themselves did not use keywords such as “software” (which had not yet become a standard term), nor did they announce that a computer’s programming stood at their cores. As a result, modern discourse-based search methods fail to generate lists of these patents without also listing scores of others aimed at electronic components and equipment. This is true even for the Bessen-Hunt technique, the keyword-based Boolean search method for the Patent and Trademark Office’s patent database with which the technique’s developers identified software patents from the 1970s onward. For these reasons, the sections that follow advance not quantitative analyses of patenting trends, but instead historical examinations of patents, applications, and court opinions.
What is the conventional wisdom on the birth of software patenting?

The Long History of Software Patenting in the United States

For nearly fifty years, scholars and commentators on computing and patent law have advanced an origin story about the birth of software patenting.¹ This story begins in 1968, when a firm called Applied Data Research (ADR) obtained Sorting System, a patent issued for a program of the same name.²

Computerworld also called Bernhart & Fetter’s 1970 computer graphics patent “the first true software patent.”

What is the conventional wisdom on the birth of software patenting?

- Michael Risch, *Americas First Patents* (2012): “While there were no computers in 1839, there was one software patent ... which claimed ‘[t]he application of a prism and pattern card, to regulate the operation of the hooks or teeth or dents to produce the variations in the pattern or figure.’ The pattern card was a primitive punch card that guided the operation of the loom to make a certain rug design.”

- Adam Mossoff, IP² working paper (2014): Claim 5 of the 1840 Morse patent “is arguably the first ‘software’ patent claim issued by the Patent Office and upheld by the Supreme Court.”
What is the conventional wisdom on the birth of software patenting?

- “In reality, software patenting predates the controversies of the 1990s and, indeed, predates the software industry itself. Thus, although it might be a stretch to credit Samuel Morse with the first software patent, it is plain that Bell Labs received an important software patent in 1951 for its ‘Error-Detecting and Correcting System.’ Within the modern software industry, Martin Goetz’s 1968 patent often is regarded as the first ‘true’ software patent.”

- 1968 ADR/Goetz patent cited in legal literature only 6 other times: 4 student notes and briefly in articles by Jonathan Barnett and Peter Menell.

I would (1) be more specific about literature being refuted and (2) acknowledge more nuanced takes.
What does this history mean for patent law today?

1. Patentable subject matter

- Con Diaz: “[T]hese insights encourage a shift away from discussing whether software is patent-eligible and towards analyzing how firms have sought patent protection for their programs.”

- Yes, if goal is to understand the history of software patent protection ... but does this help with modern patent-eligibility debate?

- In *Americas First Patents*, Risch noted that early patents would fail “historical” machine-or-transformation test, but PSM doctrine no longer seems built on history.
A Very Brief History of Software Patent Eligibility

Not Patentable Subject Matter
- *Benson* (S. Ct. 1972): can’t preempt use of mathematical algorithm
- *Flook* (S. Ct. 1978): not patentable if only novel feature is program
- *Alice* (S. Ct. 2014): “generic computer implementation” not enough
- Post-*Alice*, subject-matter challenge success >60% dist. ct. & >90% Fed. Cir. (often on the pleadings)

Patentable Subject Matter
- *In re Alappat* (Fed. Cir. 1994) (en banc): general computer + software = patentable special-purpose computer
- *State Street* (Fed. Cir. 1998): transformation of data patentable b/c “useful, concrete and tangible result”
- *McRO* (Fed. Cir. 2016): specific improvement in computer animation
- *Enfish* (Fed. Cir. 2016): improvement to computer functionality itself
1950s software patents disclosed circuits to implement programs and sometimes snippets of code.

Are these more useful than disclosures in modern software patents, which rarely include code, and sometimes just have high-level flowcharts?

If so, what happened? Were any early patent apps rejected for inadequate disclosure?

What does this history mean for patent law today?
Hamming was eager to publish in 1948 “but the lawyers at the Legal and Patent Division explained to him that they would not allow the paper to be published until Hamming obtained a patent.”

Patent may have delayed paper ... but they filed in Jan. 1950, paper published Apr. 1950, patent issued May 1951.

Did Bell Labs typically allow publication of unpatented works, or did patent facilitate publication?

What does this history mean for patent law today?

2b. Disclosure outside patents
What does this history mean for patent law today?

3. Examination in new fields

- Con Diaz’s work illustrates that real action is in the USPTO long before the courts must grapple with it.
- USPTO faces difficulties with classifying new technologies and providing uniform examination standards.
  - “Perhaps what allowed Bell Labs to win such boldly phrased patents was the fact that its applications were read not by examiners in communications, and not by those specializing in registers and electrical engineering…”
  - “Patent Office has not been a unified entity that applies patent drafting standards uniformly across its art units; the disclosure standards on which assessments of patent-eligibility were grounded could vary from one unit to the next.”
- Any guidance for dealing with this recurring challenge?
What does this history mean for patent law today?

4. Changing economics of software

- Seems odd that under current doctrine it is harder to patent versatile software that can run on a general-purpose computer than to patent a less useful special-purpose machine ... perhaps the changing economics of software innovation can help explain?

- Can looking at the technology that these companies felt should be protected shed any light?
What does this history mean for patent law today?

I don’t know ... but for a chapter in a book called *What Patents Really Do: Historical Perspectives on Current Debates*, I would try to have an answer!