Michigan Telecommunications and Technology Law Review

Volume 2 | Issue 1

1996

Comments in Response to the Patent and Trademark Office's Proposed Examination Guidelines for Computer-Implemented Inventions

Robert R. Sachs

COMMENTS IN RESPONSE TO THE PATENT AND TRADEMARK OFFICE'S PROPOSED EXAMINATION GUIDELINES FOR COMPUTER-IMPLEMENTED INVENTIONS†

*Robert R. Sachs**

Cite As: Robert R. Sachs, *Comments in Response to the Patent and Trademark Office's Proposed Examination Guidelines for Computer-Implemented Inventions*, 2 MICH. TELECOMM. TECH. L. REV. 103 (1996)
available at <http://www.mttlr.org/voltwo/sachs.pdf>

The Guidelines reflect a policy decision that computer-implemented inventions require both hardware and software elements. This policy decision and definition present several important issues. First, do the Guidelines accurately reflect and accommodate the practices of the software industry and software engineers? Second, do the Guidelines accurately reflect the current case law?

I. EXISTING POLICY AND ITS RELATIONSHIP TO SOFTWARE ENGINEERING

The policy of requiring that a computer-implemented invention be claimed as a combination of both hardware and software does not accurately reflect and accommodate the practices of the software industry and software engineers. As a result, the scope of claims to computer implemented invention may not be commensurate with the actual inventions and thereby would not provide the correct level of protection.

Computer-implemented creations, inventive and otherwise, can be broadly classified into three categories:

(1) Systems: executable computer applications embedded in computer hardware providing a complete "system";

---

† Originally submitted as a position paper for an online panel discussion.

* Mr. Sachs holds a B.A. in philosophy and B.A. in psychology from the University of San Diego and a J.D. from Yale Law School. He is presently working on an M.S. in Software Engineering. The following position paper is submitted as Mr. Sachs' personal opinion based on his understanding of patent case law, his experience with software technologies, and his experience in prosecuting software patent applications. The views expressed are his own and do not represent the opinions or positions of any institution with which he is affiliated.

(2) Applications: executable computer applications independent of computer hardware and executed on conventional computer hardware; and

(3) Tools: software code, object, kernel libraries, or other pure software entities used to create, augment, or improve computer systems and applications but, in themselves, not designed as executable systems or applications.

The second category includes both consumer and custom software applications that are provided or purchased in executable form. The third category includes application development environments, programming tools, software developer kits, and all other forms of software that are used to create computer-implemented creations in the first and second category. I will call inventions in the second and third categories "software-implemented inventions" in order to distinguish them from "computer-implemented inventions" that have been defined to incorporate hardware.

It is the increasing sophistication of software in the third category that makes possible the expanding variety and presence of computer-implemented creations in the first two categories. The growing availability of powerful or pervasive computing devices alone is not sufficient to sustain the growth of computer-implemented creations; there must be the software tools in place that enable powerful, sophisticated computer programs and systems to be readily and efficiently developed, marketed, and maintained.

However, the policy underlying the Guidelines reflects a bias toward the first category of computer-implemented inventions. As a result, it constrains the patentability of the second (and particularly the third) categories of software implemented creations by imposing a requirement on claims to inventions in these categories that does not reflect software engineers' understanding of their inventions. The requirement is that the claims must be artificially limited to a hardware embodiment even though what the inventor conceived of and created is independent of, though used on, some computer hardware. The Guidelines should employ a definition of the non-hardware elements of a computer implemented invention that properly captures the various levels and types of software entities and that allows inventions at these various entities to be properly claimed as statutory subject matter.

I will not address computer-implemented inventions in the first category since these may be claimed in a manner consistent with both the underlying policy of the Guidelines and what is actually invented. I will address the impact of the Guidelines' policy on software-

implemented inventions in the second and third categories—meaning either an executable computer program or a set of software tools or code library that can be used to create executable applications. In both categories, the computer hardware is merely the mechanism of execution and is not properly considered, from a software engineering view, as part of a software creation or, as the Guidelines state, "what the applicant has invented."[1]

Most software engineers (those of skill in the art) identify the architecture of a typical computer program, code or object library, application programming interface, and the like as an interrelated set of modules, object classes, data structures, functions, and so forth. These various software entities can be described by their structural (data and inter-module associations), dynamic (state-oriented), or functional (procedural) features. In the design of a computer program, code library, and the like, there is typically no need to expressly model computer hardware as part of the software design. The structural features are generally logical ones and are typically disclosed in a software patent application by way of block diagrams showing relationships between modules and information or event flow. The dynamic or functional features of a module are typically disclosed by way of flowcharts, data-flow diagrams, and event traces.

The problem that arises if a strict hardware-plus-software rule is imposed is this: a purely software-implemented invention would have to include, under the Guidelines, computer hardware elements in the claim language to be considered an article of manufacture or machine, or else it would be limited to a method claim. This produces an anomalous result for second and third category software-implemented inventions.

For an example of software inventions in the second category, assume a common type of software patent application—one that describes an executable computer program providing certain types of analysis or transformation of data. Examples include computer-aided design tools, financial analysis tools, and databases. Now assume a claim is written for such a computer program that casts the invention as "a computer system" and, in the body of the claim, recites selected modules of the computer program. Assume further that there is no recitation of a processor, memory, display, input/output device, or other conventional hardware since, to the inventor, the essence of the invention from which all of the "specific utility" derives is the organization and operation of the software. Assume further that the claim distinctly describes the invention such that one of skill in the art would understand the claim and

1. 60 Fed. Reg. 28,778 at 28,780 (proposed June 2, 1995).

be able to determine whether it reads on a particular computer program. Thus, the claim would serve its ultimate purpose of notifying others of the scope of the applicant's invention. However, under the Guidelines, the recitation of the modules would not recite any "physical element," and the claim would be rejected as non-statutory.

Is this the proper result? Should the Examiner strictly abide by the Guidelines, or may the Examiner impute, as one of skill in the art would, that the modules are executed by the processor and thereby define a statutory machine?

Alternatively, what problems arise if a processor, storage device, or the like is included in the claims? First, the claim does not define what the applicant invented. The applicant invented software and did not consider the invention to be a computer system, a combination of hardware and software. In my experience, where claims to software invention were always written to incorporate the computer hardware, I have had to consistently explain to the software-engineer inventors why "all that hardware" is included in the specification and the claims. Second, the scope of protection is limited to a complete system. The patentee can only sue one who "copies" the software portion as a contributory infringer, with the resulting additional complexity in the litigation. The form of the invention should dictate the patentee's enforcement options.

Next, there are article of manufacture claims. The Guidelines define an article of manufacture for computer-implemented inventions as:

> (1) a computer-readable storage medium, such as a memory device, a compact disc or a floppy disk; and
>
> (2) the specific physical configuration of the substrate of the computer-readable storage medium that represents data (e.g., a computer program), where the storage medium so configured causes a computer to operate in a specific and predefined manner.[2]

The first requirement is conceptually non-problematic. The second requirement is more difficult since it does not use the appropriate level of conceptualization of the software. The "specific physical configuration" of the substrate that "causes a computer to operate in a specific and predefined manner" is not what the inventor invented and is often not known to the inventor. That is, how a computer program is stored on a floppy diskette (the particular arrangement of object code in this or that sector) is not known, not descriptive of the invention, and not relevant to the invention.

---

2. *Id.*

What is relevant is the specific logical or physical relationship of the elements of the computer program to each other, not to the substrate. The substrate, the computer readable medium, is merely the medium of operation and transmission. To predicate patentability on the configuration of the substrate rather than the configuration of the software denies the significance of the software entities as meriting protection as inventions in and of themselves.

The problem with the Guidelines' presumptions is even worse for software-implemented inventions in the third categories: software code libraries, kernels, application programming interfaces, and other tools that, in and of themselves, are not executable computer applications but are rather the nuts and bolts that software engineers can use to create computer-implemented systems and applications.

For these software inventions a "computer system" type claim is not meaningful. What the applicant invented is a particular set of code objects, interfaces, data structures, functional procedures, or the like that are not a computer system but rather aid in the creation of a variety of different computer systems that have certain structural or procedural features defined by the invented software entity. A "computer system" claim incorrectly defines the invention since the invention is not a computer system. It likely too narrowly defines the invention since it may describe the system at a level that does not properly capture the utility of the code library, code object, or other entity but only captures the utility of a complete system. Finally, it suffers from even worse enforcement problems since the patentee would have to sue the final user/maker/seller of a computer system having computer codes that include the software entity and can, again, only sue the "real" infringer for contributory infringement or inducement. This results in a very complex enforcement problem for the patentee.

Article of manufacture claims are, again, not useful for protection here. As before, it places the cart before the horse: the invention is not the floppy but what is stored on it. Second, even when claimed on a storage medium, a code library stored thereon does not in and of itself "cause[] a computer to operate in a specific and predefined manner"[3] since the code library, as including source code or even compiled object code, does not execute on its own, but rather is intended to be integrated with other codes that will cause the operation of a computer in a specific manner.

An analogy will illustrate the problems with the existing presumptions when applied to software-implemented inventions. Second

3. *Id.*

category software inventions—application level software—are the equivalent of automobile engines, while third category software inventions are the nuts and bolts used to build the engines. "Computer system" claims requiring hardware elements are like "machine claims" for engines to be cast as, "[a]n automobile, comprising: . . . wheels; tires; . . . and an engine." The invention is the engine, not the car, but could not be so claimed. If an applicant did this for an actual improved engine, the PTO would likely issue a rejection or restriction requirement. Continuing the analogy for third category software, the claim for an improved nut, piston, cam-shaft, etc. as a "automobile" would be even more obviously incorrect.

Finally, method claims do not satisfy any of the problems identified. Method claims create problems in enforcement. Who is the infringer? The end user of the software? The company that creates the code? A developer who embeds code in an application? Method claims also create problems of breadth and scope. What elements are required to be present in a computer program or code library in order to infringe the method?

II. SOFTWARE IMPLEMENTED INVENTIONS THEMSELVES ARE STATUTORY

It is respectfully submitted that allowing claims to software-implemented inventions as themselves does not produce excessively broad claims or render claims indeterminable as to their scope and would not expand § 101 beyond the scope intended by Congress. Such software-implemented inventions could be claimed, preferably as articles of manufacture in the absence of a "computer readable medium."

As noted above, the Guidelines already state that "computer program-related elements" of a computer-implemented invention may serve as the underlying structure in support of means for claims.[4] Since means-plus-function claiming is merely a formalism, it follows that the Guidelines already comprehend that software entities may provide the sufficient structural features to provide support for apparatus and article of manufacture claims. It is recommended then that the Guidelines make explicit what is already suggested: that software implemented inventions may be properly classified as statutory machines or articles of manufacture.

First, software engineers of ordinary skill in the art understand that software must execute on a computer and that the failure to disclose or

4. *Id.* at 28,779.

claim a computer processor or a computer readable medium does not render a claim overly broad or indefinite. For software-implemented inventions, a disclosure of structural and functional features of the software entity and a claim directed to those structural or functional features would, in the typical case, be readily determinable in terms of scope and breadth. That is, a claim to a software application, code module or object, data structure, interface, and so forth as an article of manufacture would still have distinctly claimed the structural or functional features of the software and, in doing so, would likely be written at the same level of description as is commonly understood and used by software engineers to describe their creations. It is likely that such a claim would be even easier to interpret than a claim drafted in the form under the Guidelines since the applicant need not couch the claim as including hardware elements and interrelating those hardware elements with software elements in order to avoid mere aggregation. Leaving out the conventional hardware elements, which are assumed by software engineers, allows the claim to focus more precisely on what the applicant invented rather than on claim-drafting formalisms.

Second, it is submitted that software in and of itself is a member of the class of "anything under the sun that is made by man."[5] While software may, on one level, be considered "intangible," it does not fit in any of the three categories to be explicitly identified as non-patentable: laws of nature, natural phenomena, and abstract ideas. Software clearly is not a "law of nature" or a "natural phenomena." Where most resistance is found is the belief that software is an "abstract idea."

Now, it is true that computer science, as a theoretical discipline, operates on and describes logical entities defined by logical or mathematical principles. But in this regard, computer science is no different than quantum mechanics or other areas of physics which rely inherently on highly complex mathematical models to describe and define reality and which attempt to prove the existence of particles or phenomena by computation. Yet, this no more makes computer programs or code abstract than it makes quarks, leptons, or wavelets abstract because they are defined by complex equations in quantum mechanics.

For example, a stack is a well-defined and, by now, conventional "data structure." Like a simple circuit, it has a particular structure and a particular set of procedures that define its operation. It is man made; it does not define a law of nature or a natural phenomena. Is it an abstract idea? A stack is abstract only to the extent that we may attempt to prove

5. Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980).

something about its operation as a matter of logical deduction or computation. Similarly, a sorting algorithm is an abstract idea to the extent we analyze its efficiency and other properties. But, a Boolean circuit comprising only a series of NOR gates and tested for the validity of its operation is also abstract and, yet, non-controversially statutory. To the extent that a particular stack is defined to cooperate with other particular entities in a particular manner, it no longer has merely abstract properties but rather is now a part of a larger structural description of a machine or part thereof.

The problem with software is that the symbolic collapses with the physical. Software is a symbolic structural or procedural definition of a machine or machine component, but it is a definition of a machine nonetheless. This is the wisdom of *In re Alappat*:

> We have held that such programming creates a new machine, because a general purpose computer in effect becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from program software.[6]

The *Alappat* court was apparently thinking only of second category software, but the same is true of third category software. Software changes the general purpose computer to a special purpose computer because it imposes a particular structural and functional arrangement on the processor and memory. But, the processor and memory are merely the mechanisms that instantiate a particular instance of the symbolically defined machine or component. In the absence of software, a general purpose processor and memory is static, of little interest, and of no utility. How many businesses or individuals purchase computers merely to have them sit dormant on a desk? It is the software that defines the useful machine, not the machine that defines the software.

In the final analysis, the fundamental questions for software inventions really should be ones of enablement, written description, and distinct claiming. A proper written description will establish what the invention was at the time of filing and, in particular, will support the distinctive claiming of the software invention over the prior art. An enabling disclosure will define the specific structural, dynamic, and/or functional features that must be present to provide the "specific utility" to the invention. These features would, under the Guidelines, be incorporated as limitations of the claims. If these criteria are met, then others may practice the software invention by a particular embodiment. They then may surmise which software systems, applications, or tools infringe and which do not.

6. 33 F.3d 1526, 1545 (Fed. Cir. 1994).

Returning then to the specific language of the Guidelines, the following recommendations are suggested:

(1) If it is decided that computer-implemented inventions must incorporate both hardware and software, then the presumption for computer-programmed apparatus as statutory machines should specifically state that at least one computer hardware element operating under the control of, or configured by, a computer program must be recited in the body of the claim.

(2) If it is decided that software alone is a sufficient structure, then the presumptions should indicate that the elements recited in the body of the claim may correspond to specific computer program elements disclosed in the specification. This would allow software claims to be directed to a precise and distinct structural or functional description of the software. This would further allow, but not require, the recitation of hardware elements in the preamble, which are operated upon by software elements in the body. The claim, however, would be limited to the software elements for purposes of scope and interpretation.