Alice is Due for Reversal: Science Proves Its Reasoning Unsound

By David Reardon & Gene Quinn / March 21, 2019

"It is settled science that software transforms computer systems into new quantum states that are functionally and demonstrably different. Indeed, the physical transformation of computer systems by software is the only explanation for how computers are actually useful."

Since the 2014 Supreme Court decision in *Alice v. CLS Bank International*, patent claims including software have faced a much higher barrier for receiving patents than any other field of invention. This has also infected specialized software, such as artificial intelligence (AI), which is both distressing and sad. It also explains why Chinese AI start-ups are receiving more funding than U.S. AI start-ups, a fact that should be sending a shockwave through Capitol Hill.

Since *Alice*, patent examiners have presumptively classified software claims that can be implemented on a general computer as covering nothing more than an abstract idea, which means they are ineligible subject matter under 35 U.S.C. § 101. To overcome this rejection, applicants must show why their claimed invention is *something more* than just a mere *abstract idea*. Ironically, what constitutes something more is itself an abstract idea, and even what is an abstract idea is itself an abstract idea. In something straight from out of the *Monty Python* version of patent eligibility, these key terms – *something more* and *abstract idea* – have not been defined by the Supreme Court or the Federal Circuit. As a result, most applications with software are routinely denied, which is understandable when frontline decision makers (i.e., patent examiners) are left without objective guidance. Subjectivity prevails.

This new subjective *Alice* paradigm has created an uphill battle for software claims. There have been a growing number of cases in which software patents have been upheld following *Alice*. *See Software Patent Drafting Lessons from the Key Lighthouse Cases*. But that does not change the fact that *Alice* is fatally flawed and ripe for reversal based on scientific evidence that was not previously considered. In short, it is settled science that software transforms computer systems into new quantum states that are functionally and demonstrably different. Indeed, the physical transformation of computer systems by software is the only explanation for how computers are actually useful.

In failing to consider these facts, *Alice* is inconsistent and unsound. *See also* *Unintelligible and Unsound: Patent Eligibility in America*.

Judicial Exceptions and the Misclassification of Software as an Abstract Idea

In case law surrounding patent eligibility under Section101, the courts have made a distinction between discoveries of natural laws and abstract ideas and the invention of something that is tangibly useful. For example, the famous formula $e=mc^2$ was discovered, not invented, by Albert Einstein. It describes nature. Nuclear reactors and other devices dependent on an understanding of $e=mc^2$ are patentable but not the formula itself.

In *Alice*, the Supreme Court expanded this judicial exception to presumptively include all software operating on general purpose computers. This new standard was the direct result of court challenges and amicus curiae briefs filed by large tech companies (including Google, Apple, Amazon, Microsoft, IBM, Adobe, and Hewlett-Packard). These industry leaders successfully argued that the Supreme Court should subsume into the definition of ineligible abstract ideas any algorithm implemented on a general-purpose digital computer in the absence of some other necessary inventive concept.

According to the syllogism governing previous precedents: (a) math equations describe natural laws, and (b) natural laws are ineligible for patent protection, therefore (c) math equations are ineligible for patent protection.

But in *Alice* the Court erred in assuming that all software algorithms are math equations. It is true that mathematicians study algorithms. As do logicians. And software engineers. But computer algorithms are rarely limited to describing natural laws or the relationships between numbers.

The key difference between mathematical formula describing the natural world (or relationships between numbers) and computer algorithm is that the former do not include logic functions such as “If…then…else.” These logic steps are not math. They do not describe existing natural relationships between numbers or things. These logic steps are a human creation which exist in computer coding precisely to provide a means to create new, human-defined relationships between the input and output devices used to interact with humans in useful ways. Such algorithms do not describe naturally occurring phenomena, nor relationships between abstract number sets—they exist purely to transform computer systems into useful ends.

Fact: All Active Software is Transformative

Prior to *Alice,* the Supreme Court explained in *Bilski v. Kappos* that satisfying the so-called Machine-or-Transformation test was an important clue to patent eligibility for a method claim. It is interesting to note that the transformation prong of that test never applied to software, although from a scientific standpoint software is transformative by its very nature.

Fact: every quantum physicist agrees that any and every transformation of matter can be defined in terms of a change in the quantum state of that system. Notably, this critically important fact was not addressed, or likely even recognized, by the Supreme Court in *Alice.* But the expert testimony of anyone trained in quantum physics would have established that there is simply no question among scientists about the fact that *the quantum state of a generic computer is truly transformed by any change in the software and data contained in that computer’s operating memory*. This is a physical change.

It necessarily follows that software, once implemented and operating in a generic computer, is truly, physically transformative, evidenced by new properties and behaviors that are both tangible and measurable. Therefore, if *Diamond v. Diehr* continues to be good law as the Supreme Court repeatedly says it is, an invention that transforms or reduces an article to a different state must be patent eligible. *Diehr*, 450 U.S. 175, 193 (1981) (“patent laws were designed to protect (e.g. transforming or reducing an article to a different state or thing), then the claim satisfies the requirements of § 101.")

Software is as Transformative as DNA Manipulation, a Forge, or Pharmaceuticals

Arguably, the Court erred simply because the justices are not physicists and were not adequately briefed by physicists. But for those who are not familiar with quantum physics, three simple analogies are helpful.

First, when a plant seed is reprogrammed to have new traits through DNA modification, those genetically altered seeds are eligible for patents. But why is this reprogramming a seed eligible but not the reprogramming of a computer? There is no logical consistency to classifying one form of re-programming as innovative, or "something more" and the other as nothing more than merely an abstract idea that does not encompass an innovative concept.

Second, the human body has many natural states of existence: hungry, full, sleepy, alert, anxious, depressed, and more. These states of being can be altered by ingesting sugars, fats, carbohydrates, sedatives, antidepressants, poisons and more. These different ingested compounds are analogous to different programs being inputted into the working memory of a computer. As long as these ingested compounds are active in the body, the body *must react to them*. These inputs necessarily transform the state of the body. Similarly, a computer that is running a program is also transformed into an altered state of being as long as that program is present in its memory.

Third, properly understood computers are really just an elaborate form of raw material, like a few pounds of copper. Given tools, time, and energy that copper can be repeatedly transformed from a pile of screws, into gears, conduit, electrical wire or any number of useful inventions. It can even be recycled into one invention, then another. The *only difference* between transforming copper into different devices and transforming a generic computer into a clock, a gaming system, or a spreadsheet workstation, is that it takes *less* time and energy for each transformation of the computer (at least, after the first thousand hours writing the software). The transformations of both copper and computers always requires tools, time and energy, but the advantage of computers is that of efficiency, requiring less energy, time, and tools between each transformation than copper.

When the programs that transform a computer into a calculator, or a clock, or a video game, or a spreadsheet, or messaging system are described in a flow chart or printout of code, *that description of the code* is an abstraction. But *when that program is read into and actually "overlaid" into the operating memory* of a computer system, each program step truly transforms the quantum state of the computer system and purposefully reconfigures the flow of information and energy into patterns that are meaningfully and intentionally designed to produce useful results.

Most notably, all of these changes are tangible and measurable. In contrast to the realm of abstract ideas that can accommodate an infinite amount of information, in a real computer system the program steps and related data face true physical limitations. Each bit occupies physical space in the form of transistors that must be held in specific electronic states as defined and controlled by the software overlay. The space occupied is measured in nanometers, but it is truly a physical space. It is not abstract. Moreover, each space in this carefully controlled physical system must be correctly transformed since a single physical error can trigger a "blue screen of death."

Application to the Subject Matter Eligibility Test

The stumbling block for many applicants is found in Step 2A of the *Alice/Mayo* framework, which asks: "Is the claim directed to a law of nature, a natural phenomenon (product of nature), or an abstract idea?" Typically, examiners reflexively reject software patents under Step 2A as being directed to an abstract idea and define the abstract idea so broadly that the analysis made under Step 2A infects any analysis under 2B (i.e., the so-called hunt for the inventive concept). It is simply wrong to describe a software overlay that is in active computer program memory as non-transformative, much less, nothing more than an abstract idea. In fact, software truly changes a computer system in significant, measurable, and useful ways. That alone should make it impossible for any properly described software to be an abstract idea.

The Patent Office, Federal Circuit and Supreme Court should all thoughtfully consider the science and technology involved. There has been great mischief created by *Alice*, and all because no one bothered to consider what is actually occurring on a physical level.

Perhaps the software patent claims being considered by decision makers do not describe a novel innovation, or perhaps what is described would be obvious to one of skill in the art. Further still, it is certainly possible that the patent claims reviewed are not adequately supported by a written specification that describes with appropriate detail the fullness of the innovation in a way that teaches those of skill in the art. But what we know for certain is that the software, when introduced to a computer—even a general purpose computer—will make physical changes that are perceptible.

Tangible physical changes and rearrangement or an article into a different state or different thing has always been the hallmark of a patent eligible invention. The fact that the alteration in state is something that cannot be seen or isn't appreciated by the masses should not justify a *per se* rule that innovation of a certain category is simply not eligible for patent protection. As Judge Pauline Newman has explained repeatedly in her opinions, § 101 is intended to be a low threshold so that it does not prevent innovation from flourishing before it is known what may result from a particular field of endeavor.

It is recommended that these arguments (and if practical, the expert opinion of physicists regarding the change in computers' quantum states due to active software) should be presented in every prosecution of software claims, until such time as USPTO guidance acknowledges this fact.

The Author

David Reardon is an electrical engineer who has obtained numerous patents, many representing himself, in the field of computer security.