From: Wesley Parish
Sent: Saturday, January 05, 2013 4:55 AM
To: SoftwareRoundtable2013
Subject: Software patents(Comments): Additional topics

First, the preliminaries:

Is teaching people software development to be made illegal? That is one impression I get from the constant software patent battles that enfilade the media - if a given algorithm is to be the subject of an overly broad patent, that algorithm can no longer be the subject of education without an expensive license. When you add all the software patents applied for, you price software development education out of the picture - and thus price the American software industry out of existence. I am sure IBM, Microsoft, Google, etc, will be delighted to know just how thoroughly their software patenting sprees have made their current share prices a fraud upon their investors.

Now we have the preliminaries out of the picture, we come to the question of:

"Software by its nature is operation-based and is typically embodied in the form of rules, operations, algorithms or the like."

This is unnecessarily vague and undefined. Software takes the form of algorithms manipulating data in data structures.

Wikipedia has commonly-accepted definitions of algorithm and data structure:
http://en.wikipedia.org/wiki/Algorithm
"More precisely, an algorithm is an effective method expressed as a finite list[1] of well-defined instructions[2] for calculating a function.[3] Starting from an initial state and initial input (perhaps empty),[4] the instructions describe a computation that, when executed, will proceed through a finite [5] number of well-defined successive states, eventually producing "output"[6] and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate random input.[7]"

http://en.wikipedia.org/wiki/Data_structure
"In computer science, a **data structure** is a particular way of storing and organizing data in a computer so that it can be used efficiently.[1][2]"

The data coming into a computer may be randomly organized, which is another way of saying it may not be organized at all: once it is inside the computer, it must be organized into data structures for the algorithm to manipulate.

With relation to:
https://www.federalregister.gov/articles/2011/02/09/2011-2841/supplementary-examination-guidelines-for-determining-compliance-with-35-usc-112-and-for-treatment-of

"3. Computer-Implemented Means-Plus-Function Limitations: For a computer-implemented means-plus-function claim limitation invoking § 112, ¶ 6, the corresponding structure is required to be more than simply a general purpose computer or microprocessor. [96] To claim a means for performing a particular computer-implemented function and then to disclose only a general purpose computer as the structure designed to perform that function amounts to pure functional claiming. [97] The structure corresponding to a § 112, ¶ 6 claim limitation for a computer-implemented function must include the algorithm needed to transform the general purpose computer or microprocessor disclosed in the specification. [98] The corresponding structure is not simply a general purpose computer by itself but the special purpose computer as programmed to perform the disclosed algorithm. [99] Thus, the specification must sufficiently disclose an algorithm to transform a general purpose microprocessor to the special purpose computer."

as a guide to the USPTO, might I point out that it relies implicitly on the prior iteration of computer technology, the analog computer. To make an analog computer into a incarnation of a particular program was to make the analog computer; to change the programming of the analog computer was to remake it.

The digital computer is a general purpose algorithmic device that simulates various special-purpose analog devices through algorithmic manipulation of data in data structures in short- and long-term memory. If the algorithm as a mathematical procedure cannot be patented, then neither can the generalized algorithmic device simulating an analog device.

To be blunt - a general purpose algorithmic device simulating various special-purpose analog devices, does not become the various special-purpose analog devices while it is running the algorithms manipulating the data structures that simulate the various special-purpose analog devices - special-purpose analog devices are notorious for being special-purpose (read: not capable of being multitasked with other such devices), and general-purpose algorithmic devices are notorious for multitasking more than one simulation of special-purpose analog devices. General-purpose algorithmic devices are even capable of multitasking simulations of general-purpose algorithmic devices: talk to any IBMer in the zVM dept, for pete's sake!

And the reason for that is the general-purpose storage of data in short-term and long-term memory. This goes back to Turing at Bletchley Park and then von Neumann's work, so ignorance cannot be claimed as an excuse for ignorance ...