Testing Parameters for Software Patentability

Arun Kishore Narasani† and Kalyan Chakravarthy Kankanala†
Brain League Consultants, NSRCEL, IIM, Bangalore

Received 19 April 2005

Patentability of computer programs is fraught with ambiguity because of multiple reasons. One such reason is the uncertainty and inadequacy of tests to determine patentability. Courts in the US have been struggling to evolve a test that would cover the complete continuum of innovation in computer programs, which manifests in terms of form or function or both. All tests adopted by the court either focus on only form or function, thus missing out the other. It is important for the legislatures and courts to understand the existing lacuna and adopt a test that would lay emphasis on form and function. The test should be modeled to allow inventions having high function or high form or both form and function in high proportion. It should keep inventions having low form and function outside the scope of patent protection. Such a model would promote progress in the entire continuum of the invention in the software field by granting patents to worthy inventions.

This paper proposes parameters for coining a reliable test for determining patentability. A test based on parameters such as form and function would encourage innovation in the software industry without stifling progress through grant of frivolous patents. The first part of the paper expounds the law relating to software in USA and India. The second part analyses the existing patentability tests in terms of importance given to form and function of computer programs and third part discusses the usage of form and function as parameters for coining a good patentability test.

Keywords: Software, computer programs, patentability, form-function-test

Software as a field is characterized by ‘sequential’ and ‘complementary’ innovations. Owing to the nature of the industry, it is known that introduction of an uncertain software patent system has at the least stagnated the growth of innovation in the field if not reduced it. Whether or not software patents should be allowed has been a heated debate for over a decade in the US and similar debates would arise in India as India gets in compliance with the rest of the world through the WTO. About a hundred thousand software patents have already been issued in the US and if belief has to be laid in the phenomenon of industry lobby, it would be no great surprise when India allows software patents at some point in time. In the present context, what is more important is what must be done to minimize or if possible eradicate the reversing effect of the patent system on the software industry and not whether or not software patents should be allowed at all.

The primary concern about software patenting is the danger of preempting human thought process or manual method of accomplishing a particular task. Though the concern is valid, in certain instances, software development has evolved for long enough to believe that there are numerous instances where so much is being accomplished through the use of software that human being could never have thought of accomplishing through plain manual labour or thought process. For example, a human being cannot create a dynamically changing picture but that is possible through images on computers today. Though it is naïve to think that software is all about an algorithm and algorithm is just a representation of a human thought process, things in software are not so simple.

Software as a field is very complicated for patent purposes. Complications with patenting software have resulted in a host of silly and unworthy software patent grants in the US for several reasons. One of the reasons is the difficulty in applying a reliable patentability test. Courts and Legislatures are struggling to apply the existing tests to differentiate patentable inventions from unpatentable ones. The tests being applied today are either too broad or too narrow resulting in destruction of patent law's objective to protect worthy inventions for promoting innovation in the software industry.

The first part of the paper expounds the law relating to software in USA and India. The second

†Email: arun@brainleague.com and kalyan@brainleague.com
part analyses the existing patentability tests in terms of importance given to form and function of computer programs and third part discusses the usage of form and function as parameters for coining a good patentability test.

**US Patent Law and Computer Programs**

US patent law codified under Title 35 deals with patentable subject matter. It provides that 'any person who invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore...'. As Section 101 had been very broadly worded, the developments in patent law, in response to computer technologies were largely an outcome of judicial interpretation.

Though the courts dealt with the issue in 1968, the first decision dealing with computer program was given in the year 1972. The initial decisions of the courts were based on two doctrines, the 'mental steps' doctrine and the ‘function of the machine’ doctrine. The mental steps doctrine postulates that processes involving mental operations are unpatentable. The mental-steps doctrine was based upon the familiar principle that a scientific concept or mere idea cannot be the subject of a valid patent. The ‘function of a machine’ doctrine postulates that a machine is patentable but its functions are not patentable.

The mental steps doctrine was regularly invoked by courts to deny patents to inventions consisting primarily of mathematical formula or methods of computation or in which the sole inventive contribution was a mental operation or mathematical computation. Based on the mental steps doctrine the patent office adopted guidelines on the examination of application for patents on computer programs in the year 1968, which provided that computer programs claimed as an apparatus or a process are not patentable. However, the patent office indicated that a programmed computer could be a component of a patentable process if combined with unobvious elements to produce a physical result. The Patent Office formally adopted the guidelines in 1968.

In the same year, the Court of Customs and Patent Appeals (CCPA) changed the law completely by repudiating both the 'function of a machine' and 'mental steps' doctrines in two separate decisions and expanded the scope of patentable subject matter to include computer programs. The Prater decision opened the gates for processes involving computer programs by holding that a process involving mental steps would not be unpatentable if the steps can also be performed without mental operation. In re Bernhart, the court reaffirmed Prater, and indicated that all that remained of the mental-steps doctrine was a prohibition on the granting of a patent that would confer a monopoly on all uses of a scientific principle or mathematical equation. The court also announced that a computer programmed with a new and unobvious program was physically different from the same computer without that program; the programmed computer was a new machine or at least a new improvement over the unprogrammed computer.

After the CCPA opened the door for patents on computer programs, the US Supreme Court and Federal Courts tried to clarify the issue in Benson, Parker, Diamond, Alappat and State Street Bank, by differentiating patentable computer programs from non patentable abstract ideas, natural phenomena and laws of nature. In the process, the courts designed and tried different patentability tests. The uncertainty in applying patent principles to computer programs can be seen from the decisions and the reasoning of the courts.

**Gottschalk v Benson**

The case involved an invention consisting of a method for converting binary-coded-decimal numerals into pure binary numerals. The method was claimed in a general-purpose computer of any type and was not limited to a particular art, apparatus or machinery. The claims were rejected by the patent office but sustained by the CCPA. The Supreme Court reversed the decision of CCPA and held the claimed invention to be non-patentable subject matter.

**Reasoning of the Court**

The court started by reiterating that phenomena of nature, though just discovered, mental processes, and abstract intellectual concepts are not patentable, as they are the basic tools of scientific and technological work. Based on that principle it reasoned that allowing patent protection over the method of converting binary coded decimal numerals to pure binary numerals in this case would result in patenting of ideas, because the mathematical formula involved has no substantial practical application except in connection with a digital computer. As agreeing to the patentability of the claimed method would wholly pre-empt the mathematical formula and in practical
effect would be a patent on the algorithm itself, the court held that the claimed method is not eligible for patent protection.\(^{18}\)

**Test**

The court in this case determined patentability of the invention by testing whether the computer program has a substantial practical application. The program was held not patentable, as it did not have substantial practical application. This test proved to be very ambiguous, as the phrase 'substantial practical application' was not defined.

**Parker v Flook\(^{19}\)**

The dispute in this case related to the patentability of an invention involving a method for updating alarm limits. The CCPA rejected the examiner's decision and held the invention to be patentable subject matter. The Supreme Court reversed the decision of CCPA and held the invention to be non-statutory subject matter under Section 101 of the Patent Act because the invention just identified post solution applications of the formula and as the only novel feature of the invention was the mathematical formula or algorithm.

**Diamond v Diehr\(^{22}\)**

The case related to a dispute over the patentability of a process for molding raw, uncured synthetic rubber into cured precision products. The invention involved the process of constantly measuring the temperature inside the mold and feeding the temperature measurements into a computer that repeatedly recalculates the cure time by use of the mathematical equation (Arrhenius equation) and then signals a device to open the press at the proper time. The patent examiner rejected the application on the ground of non-statutory subject matter stating that the invention involves nothing more than a mathematical equation along with conventional industrial methods. The supreme court disagreed with the patent examiner and agreed with the CCPA by holding that a process would not be non patentable subject matter simply because it employs a mathematical formula, computer program or digital computer as long as it produces an useful result.

**Reasoning of the Court**

The court started its analysis by stating that a patent is granted in USA for the invention of some practical method or means of producing a beneficial result or effect and not for the result or effect itself.\(^{23}\) As the claimed process describes a new method of producing a result, the court held that it is patentable subject matter, though it uses well-known steps.\(^{24}\) Despite the fact that Arrhenius equation is not patentable in isolation, the court opined that the process for curing rubber, which incorporates in it a more efficient solution of the equation, is patentable.

**Test**

Though the court tried to indicate a common line of reasoning with Gottschalk and Parker cases, the test adopted by it in this case is different from those cases. The test used by the court in this case is whether the invention containing the program as a whole produces a patentable result. This test removed lot of uncertainty by taking a wholistic view, but failed to differentiate between patentable and non-patentable computer programs.

**Freeman-Walter-Abele test**

After Diehr, the US Courts followed the Freeman-Walter-Abele test for some time. The test involved a two-step analysis of the claimed invention.\(^{25}\) First, the court determined whether a mathematical algorithm is recited directly or indirectly in the claimed invention.
If yes, the court then checked whether the claimed invention is directed to a mathematical algorithm that is applied to or limited by physical elements or process steps. If the algorithm were applied in one or more steps of an otherwise patentable process or one or more elements of a patentable product, the claimed invention would be patentable. If the claimed invention were limited to the mathematical algorithm, it would not be patentable. This test was abandoned after Alappat’s decision.

In re Alappat
The case relates to a dispute over the patentability of an invention, which is a means for creating a smooth waveform display in a digital oscilloscope. The examiner rejected claims 15-19 of the patent application stating the claimed invention to be non-statutory subject matter. The Federal Circuit disagreed with the examiner and held the invention to be a machine, which falls under the scope of statutory subject matter.

Reasoning of the Court
The court in this case opined that the Supreme Court’s test for patentability of computer programs was not clear. So, it made an attempt to give lucidity by interpreting Diehr's (useful result) decision to mean that any computer program producing a concrete, useful and tangible result is patentable. Based on this interpretation the court held the invention in this case, which is a computer program to create a smooth waveform in a digital oscilloscope, to be patentable as it produces a concrete useful and tangible result. The court looked at the programmed computer as a new machine, because it believed that 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.

Test
The Federal Circuit in this case coined the test being applied today by patent office and courts, which is the concrete-useful-tangible-result test. As per this test, a computer program is patentable if it produces a concrete, useful and tangible result. The court in this case looked at a computer containing a computer program as a new apparatus capable of producing a useful result. Though the test seems to be prima facie simple and clear, the meaning of the terms concrete and tangible remain unsettled.

State Street Bank v Signature Financial Group
The case involved a fight between Signature Financial Group Inc (hereinafter called ‘Signature’) and State Street Bank and Trust Co (hereinafter called ‘State Street’), over a patent directed to a data processing system for implementing an investment structure, which was developed for use in Signature’s business as an administrator and accounting agent for mutual funds. As its negotiations to get a licence from Signature failed, State Street claimed patent invalidity stating that the data processing system falls outside the scope of patentable subject matter under Section 101. The Federal Circuit reversed the ‘patent invalidity’ decision of District court of Massachusetts. In its decision, the Federal Circuit clearly stated that a process patent should not be held invalid merely on the ground that it involves a mathematical algorithm as long as it produces a concrete, useful and tangible result. It also clarified that there is no prohibition against business method patents.

Reasoning of the Court
The court construed claim 1 of the patent, which claims a machine, namely, a data processing system for managing a financial services configuration of a portfolio established as a partnership and went on to determine whether the claim fell under one of the judicially created exceptions to statutory subject matter, which are, ‘mathematical algorithm’ exception and ‘business method’ exception.

Mathematical Algorithm Exception
The court started its reasoning by stating that mathematical algorithm exception is an expansion of the law against patentability of abstract ideas. It reiterated the decision in Diehr, where the court explained that certain types of mathematical subject matter, standing alone, represent nothing more than abstract ideas until reduced to some type of practical application i.e. a useful, concrete and tangible result. The court further stated that the Freeman-Walter-Abele test has very little significance after the Chakrabarty and Diehr decisions of supreme court. It cited In re Alappat, where the court held the use of Freeman-Walter-Abele test to be misleading because the existence of an algorithm in an invention would not render it unpatentable even if it was not involved in a practical application. It further went on to state that when a claim containing a mathematical formula implements or applies that formula in a structure or process which, when considered as a whole, is
performing a function which the 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 patentable subject matter. The court went on to say that it is irrelevant that a claim may contain, as part of the whole, subject matter which would not be patentable by itself. "A claim drawn to subject matter otherwise statutory does not become nonstatutory simply because it uses a mathematical formula, computer program or digital computer." Finally, the court concluded that the proscription against patentability is limited to algorithms standing alone and not to any inventions involving them. Based on this reasoning the Federal Circuit held that the mere existence of an algorithm/computer program in Signature's patent would not make it unpatentable.

**Business Method Exception**

The Court held that there is no business method exception and the notion or exception is ill-conceived. It took the support of Judge Newman’s statement where he said that the business method exception is an unwarranted encumbrance to the definition of statutory subject matter, that should be discarded as error-prone, redundant, and obsolete and that it merits retirement from the glossary of statutory subject matter. The court stated that patentability does not turn on whether the claimed method does ‘business’ instead of something else, but on whether the method, viewed as a whole, meets the requirements of patentability as set forth by the Patent Act. Based on the aforementioned analysis, the court said that Signature's patent can not be held invalid based on it being a business method.

**Test**

In State Street, the court reiterated the test coined in Alapat’s case. It defined practical application as defined in Diehr to mean concrete, useful and tangible result. Courts and PTO are following this test to determine patentability of computer programs.

It can be seen from the decisions that the test for patentability of computer programs has gone a long way from the once applied mental steps and function of a machine tests to today’s concrete-useful-tangible result test. Despite genuine efforts, US Courts have not been able to arrive at a lucid and certain test to determine patentability of computer programs. All the tests, including the test being applied today are fraught with ambiguity and uncertainty.

**Indian Patent Law and Computer Programs**

Indian Patent Law is governed by the Indian Patent Act of 1970 as amended in 1999, 2002 and 2005. In India, unlike in USA, there is a provision proscribing the patentability of computer programs. Section 3 of the Indian Patent Act as amended in 2002 provides under clause (k) that a mathematical or business method or a computer program *per se* or algorithm is not a patentable invention. Indian Government made an attempt to change this in 2004 through the patent amendment ordinance by providing under Section 3 (k) that a computer program producing a technical application to industry or a computer program in combination with hardware is patentable. But the attempt was short lived as the government-repealed ordinance through an amendment Act in March 2005.

Though the section lays down a blanket prohibition on the patentability of computer programs, it extends to only computer programs standing alone (*per se*). The clause can be and would most probably be interpreted to mean that an invention would be patentable despite the presence of a computer program as one of its elements, if the invention as a whole includes something more than the computer program that is eligible for patent protection.

**Form-Function Analysis**

Parameters such as speed, automaticity, accuracy and efficiency have been recognized as elements of computer programs that can make such programs do much more than mimicking the functions of the human mind. Innovation in computer programs manifests itself in a continuum of form and function. Its expression might be in either form or function or both in variable proportions. The word ‘form’ means the construct/arrangement of the parts of the product and ‘function’ means the incremental improvement that a software program provides in any of the mentioned parameters, namely, speed, automaticity, efficiency and accuracy.

Courts in US have been laying differential emphasis on form and function for determining patentability of computer programs. Initially, the courts used the mental steps doctrine and function of a machine doctrine to determine patentability of computer programs. Both the doctrines focused on form and gave little or no importance to function, while deciding on patentability. The doctrines postulated that function performed by a computer program is least important and that the physical result,
i.e. the form it produces is important. In Prater and Bernhart, the courts eliminated both the aforementioned doctrines and looked at a computer containing an unobvious computer program as a new and improved machine. This perspective gave very high importance to function and little importance to form. Even if there was no change in form, the program was patentable as long as it performed new function. This is a total reversal from the earlier point of view.

The test used by the court in Gottschalk gave primary importance to function by looking for substantial practical application while deciding patentability of a computer program. Parker's case used existence of inventive concept or inventive application to test patentability. By doing so, it relied on both form and function.

In Diehr, the US Supreme Court gave more importance to function performed by the computer program under the garb of using the mathematical method exception. The court explicitly mentioned that as long as a computer program produces a useful result, it is patentable. Here the court gave importance to both function and form, though more emphasis seems to have been laid on function.

The Freeman-Walter-Abele test, which was applied after Diehr reverted back to the physical result produced by a computer program. This test gave importance to both form and function. However, primary emphasis of this test was on form and function was a secondary consideration.

Finally, the concrete-useful-tangible-result test from Alappat and State Street Bank gave full importance to function and very little importance to form. This decision has opened gates to a number of software patents based on function without any relevance to form.

The Indian patent regime allows patents for only embedded software and computer programs prompting changes in hardware. This test gives paramount importance to form and very little importance to function, which makes the scope of software patentability very narrow.

Analysis of various tests for patentability of computer programs indicates that the courts have been applying form and function variably. Such an application gives rise to uncertainty and ambiguity among inventors in the software industry, potentially impeding the progress of invention in the field. Non-consideration or variable consideration of either form or function would result in closing doors to patenting of inventions manifesting in the parameter, which has not been or has been variably considered. So, it is important to give proper weightage to both form and function for testing patentability of computer programs because all inventions in the software field manifest in either form or function or both and such a test would cover all patent worthy inventions.

Form and Function Approach

A program that provides significant benefits in any one or both the parameters (form and function) can be thought of adding patentable value to the solution. As computer programs produce results in the nature of both form and function, both parameters should be considered in testing their patentability. Most tests have focused only on either form or function. Even in the tests where courts used both form and function, primary emphasis was laid on only one parameter. All tests used by courts till date failed to cover the entire innovation process in computer programs.

Form-Function Model

The form-function model proposes that any reliable test to determine patentability has to consider both form and function in order to cover the complete continuum of innovation in software. Such a test would promote progress in the industry by removing ambiguities in software patenting. The model points out approximate levels of form and function that might be sufficient for a computer program to merit patent protection. This model is proposed to be applied in addition to the other patentability requirements (patentable subject matter, usefulness, novelty, nonobviousness and specification).

The Fig. 1 shows a way of looking at an invention involving software either wholly or as a part of the subject matter for determining its patentability. In
Fig. 1 the model classifies subject matter into four quadrants based on the contribution to invention from ‘form’ and ‘function’. Patentability of any computer program depends on which quadrant it falls under.

Under quadrant 1, the contribution from both ‘form’ and ‘function’ is low. An invention would not be patentable if it falls under this quadrant because the amount of innovation would be too small to warrant patent protection. For example, the invention in Gottschalk, which involved a computer program for converting binary coded decimal numerals to decimal numerals falls under quadrant 1. This invention has very small function and no form.

An invention falling under quadrant 2 has high form and low function and vice versa for an invention falling under quadrant 3 (low form and high function). A computer program falling under both the quadrants would be patentable, as the cumulative impact of both form and function would make the innovation sufficient to merit patent protection. An example of an invention under quadrant 2 is a process for molding synthetic rubber, which involves the use of a computer program for calculating the mold time (Diamond v Diehr case). This invention has very high form and low function. The function of the program is just automation of calculations, which can be performed manually. As the manifestation of innovation is high in terms of form the invention would get patent protection under the model. Illustration of invention falling under quadrant 3 is the computer program in State Street case, which involves a program to a data processing system for implementing an investment structure. The invention has very high function, which is to make complicated calculations in order to assess profits or loses in mutual funds but no form. This computer program also would merit patent protection because the invention manifests as very high function, which is enough under the model despite non-existence of form.

Finally, quadrant 4 shows innovation manifestation of high form and function in the computer program. Such programs are patentable prima facie. For example, embedded software to control temperature in microwave oven. Innovation in this program manifests as high function, which is calculating temperature levels and high form, which is activating hardware to control temperatures. The function and form in this case are high enough under the model to warrant patent protection.

Though it is difficult to draw lines between different quadrants and to determine the cumulation of form and function (mathematically) required to make the computer program inventive enough to credit patent protection, the model provides an objective criteria based on which a good test of patentability can be coined. It also provides an objective basis for the courts and patent offices to make their decisions on patentability of computer programs.

Tests for patentability used in US and India lie at two ends of the spectrum. The US test for patentability (concrete-useful-tangible) covers quadrants 3 and 4 but misses out inventions falling under quadrant 2. The Indian test falls under quadrant 4 and misses out inventions falling under quadrants 2 and 3. Both US and India should think about tuning their laws and tests to cover all three quadrants. A test covering all three quadrants would encourage invention and innovation in the software industry by extending the patent regime to credible inventions, which manifest themselves in the inventive spectrum of form and function.

Conclusion
Existing tests for patentability are not effective, as they do not consider the complete continuum of form and function, which is the expression of innovation in the software field. The test under the US patent law by focusing primarily on function leaves out inventions in software, which manifest as form. Computer program patentability test followed by Indian patent office gives high importance to form and function and in the process misses out inventions having high function or high form only. The courts, legislatures and patent offices in India and USA should try and follow a test using form function parameters, which would cover the complete spectrum of innovation manifesting as form or function or both. Cumulative or combined effect of both form and function would confer patents on worthy inventions, thus promoting progress in computer technology.

References
1 The terms software and 'computer program' used in this paper shall have the same meaning
3 Tandon Radhika, Moving forward: Patentability of software and business method patents, 6 Intell Prop I Bull 1, Spring, 2001
4 Hanish Marc S, Yeung Adrienne & Ritchie David B, How are computer software inventions patentable?, 9-DEC Nev Law, 12 December 2001
5 35 USC Sec. 101, 2003
6 Diamond v Diehr, 450 US 175, 1981, 195
7 Id, 195
8 Id citing In re Bolongaro, 62 F.2d 1059, 1060, 20 CCPA (Pat.) 845, 846-847, 1933
9 Id 195
10 Diamond v Diehr, 450 US 175, 1981, 197
12 Id citing, in In re Tarczy-Hornoch, 397 F.2d 856, 55 CCPA (Pat.) 1441, 1968 (overruled 'function of a machine' doctrine) and In re Prater, 415 F.2d 1378, 56 CCPA (Pat) 1360, 1968 (overruled 'mental steps' doctrine)
13 Diamond v Diehr, 450 US 175, 1981, 198
14 417 F.2d 1395, 57 CCPA (Pat) 737, 1969
15 Id, 1400, 57 CCPA (Pat), 744
16 409, US 63, 1972
17 Gottschalk v Benson, 409 US 63, 1972, 71
18 Gottschalk v Benson, 409 US 63, 1972, 72
19 437 US 584, 585-86, 1978
20 Parker v Flook, 437 US 584, 1978
21 Id, 585
22 450 US, 175, 1981
23 Diamond v Diehr, 450 US 175, 1981, 183
24 Diamond v Diehr, 450 US 175, 1981, 184
25 In re Freeman, 573 F.2d 1237, 197 USPQ 464 (CCPA 1978); In re Walter, 618 F.2d 758, 205 USPQ 397 (CCPA 1980); In re Abele, 684 F.2d 902, 214 USPQ 682 (CCPA 1982).
26 33 F.3d 1526 (Fed. Cir. 1994)
27 149 F.3d 1368 (Fed. Cir. 1998)
28 Section 3, Indian Patent Act, 1970 as amended in 2002
29 Jur Strobos, Stalking the elusive patentable software: Are There Still Diehr or was it just a flook? Harvard Journal of Law and Technology, 6, Spring Issue, 1993