

*Comment*

*Developing a Comprehensive Software Claim Drafting Strategy for U.S. Software Patents*

*Keith E. Witek* †

**TABLE OF CONTENTS**

I. INTRODUCTION

- A. The Increased Issuance of Software Patents
- B. Challenges Facing Software Patent Practitioners
- C. Overview

II. CONVENTIONAL STRUCTURE AND PROCESS CLAIMS

- A. The Early Software Patent Case Law
- B. Software Structure/Apparatus Claims
- C. Software Process Claims with Structural Limitations or Practical Application Limits

III. CONVENTIONAL MEANS-PLUS-FUNCTION CLAIMS

- A. The Case Law Approving of Means-Plus-Function Claims for Software Innovations
- B. *Alappat*-Type Means-Plus-Function Software Claims

IV. LIMITATIONS OF CONVENTIONAL SOFTWARE CLAIMS

- A. Conventional Software Claims Are Less Likely To Be Infringed
- B. Contributory Infringement Limitations of Conventional Software Claims

V. ARTICLE OF MANUFACTURE CLAIMS

- A. Introduction to *Beauregard* Software Article of Manufacture Claims
- B. Allowance of Article of Manufacture Claims Over 35 U.S.C. § 101
- C. USPTO Rejection Techniques Under 35 U.S.C. §§ 112 and 103
- D. Summary of *Beauregard* Article of Manufacture Claims

VI. METHOD OF MANUFACTURE CLAIMS

- A. Comparison Between a Conventional Process and a Method of Manufacture Claim

B. Advantages of Software Method of Manufacture Claims

VII. DATA STRUCTURE CLAIMS

A. Using a Software Patent to Protect Software Data Structures

B. Barriers to Data Structure Patentability

C. The Advantage of Data Structure Claims

VIII. A COMPREHENSIVE SOFTWARE CLAIM DRAFTING SCHEME

A. Cost Analysis of the Comprehensive Software Claim Drafting Scheme

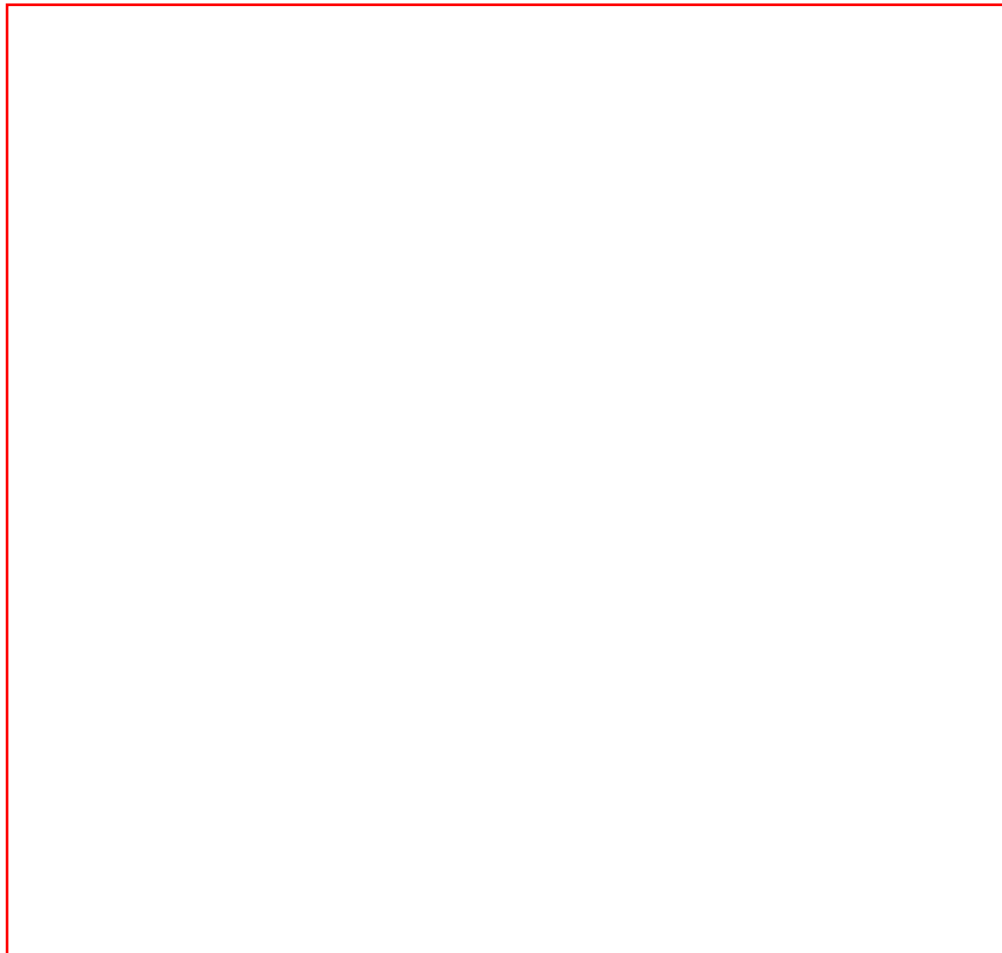
B. Disclosure Required to Support the Comprehensive Software Claim Drafting Scheme and to Provide Backup Positions During Prosecution

IX. CONCLUSION

**I. Introduction**

**A. The Increased Issuance of Software Patents**

In the face of rapid expansion and heightened competition within the software industry, software patent activity has increased significantly in recent years. This change is marked by the increased granting of software patents by the United States Patent and Trademark Office (USPTO) and by their growing acceptance in the Court of Appeals for the Federal Circuit (CAFC).<sup>1</sup> The following figure illustrates the exponential growth in software patent issuance from 1971 to 1994:<sup>2</sup>



This exponential growth suggests that attorneys will need to pursue aggressive software patent portfolios on behalf of software

industry clients in order for those clients to compete effectively. Multimillion-dollar software patent law suits have already begun to emerge.<sup>3</sup> A well-drafted software patent could make a crucial difference in the amount of royalty payments awarded in software licensing or litigation. This paper examines how best to claim and disclose software patents before the USPTO. The methods discussed herein will help patent practitioners create software patent portfolios which will effectively serve the contemporary offensive and defensive patent needs of the software industry client.

## **B. Challenges Facing Software Patent Practitioners**

Software patents are difficult documents to draft when compared to other technology patents. Software is an abstract manifestation and is not easily represented in two-dimensional form. In many instances, flowcharts, pseudocode and other graphical methods for illustrating software concepts are archaic and ineffective. A software patent practitioner must understand both electronic computer hardware and software programming from a technological, business and legal perspective in order to serve the client effectively. These challenges are aggravated by the fact that the software industry is at least partially hostile to software patents.

Software claims must take into account the dynamic and ever-changing software industry so that software utility patents, which can be enforceable for up to twenty years, have a maximum useful enforcement duration. This further complicates the software drafting process. In the 1970s and early 1980s, most high technology corporations believed that profit was primarily derived from the hardware. Software was viewed as part of a secondary market that could be left to the smaller players.<sup>4</sup> In the modern market, however, many companies reap billions of dollars in revenue from software products alone. Many companies now rely on software as their premier products, and are pushing more aggressive software campaigns because they believe that the future lies in software technical advancement and market share.<sup>5</sup> As market perceptions and business strategies shift, software patents and their claims must be carefully tailored for the client's business needs. A valuable and long-lived software patent can only be obtained by developing software claim drafting strategies that protect the added value that these patents represent.

The biggest challenges that have faced the software patent practitioner are the barriers to software patentability under the software patent case law and USPTO practice. The early case law on software patents created significant barriers to patentability.<sup>6</sup> In particular, 35 U.S.C. § 101 created problems for the software patent attorney with respect to whether software is patentable subject matter.<sup>7</sup> Even if software is patentable, can the practitioner get some claims through the USPTO without too much trouble? The attorney's main goal with respect to the early 35 U.S.C. § 101 case law was ensuring that he did not become too greedy with his claims, thereby resulting in an uphill battle in the USPTO trying to satisfy the statutory subject matter provision. To overcome 35 U.S.C. § 101 rejections under the early case law, patent attorneys simply drafted safe structural software/computer claims, process software claims with significant physical steps, or the recently accepted means-plus-function software claims under 35 U.S.C. § 112, sixth paragraph.<sup>8</sup>

Traditionally, the case law and USPTO practice regarding software patenting have been vague, largely form over function, constantly in flux and inconsistent. This situation creates an uncertainty that causes even longer drafting times for software patents as compared to other technology patents. A software patent already takes 25% longer to draft than an electrical system patent application, 35% longer to draft than a chemical or composition of matter patent application, and 50% longer to draft than a manufacturing process or method of manufacture patent application due to the issues mentioned above.<sup>9</sup> These longer drafting times are also caused by the need to disclose unnecessary figures and text in a software patent specification in order to overcome specific problems of statutory subject matter,<sup>10</sup> obviousness,<sup>11</sup> and enablement and best mode.<sup>12</sup> In particular, the figures in a software patent specification should include both hardware and software figures (a double disclosure) in order to avoid these problems during prosecution, litigation and licensing.<sup>13</sup>

Further, many more claims are needed in software patent applications than in applications for other technologies. These additional claims are necessary to provide fall-back positions due to 35 U.S.C. § 101 attacks and to provide complete coverage for the software innovation. A typical software patent application easily contains in excess of twenty total claims, with roughly seven independent claims and a proper number of dependent claims as are needed to vary claim scope from broad claims to more narrow claims.<sup>14</sup> In comparison, fewer claims are needed for patent applications for other technologies, such as chemical compositions, manufacturing methods, electrical apparatus and mechanical systems, because claim styles used for these art areas have been set in stone for decades and will be unquestionably valid under 35 U.S.C. § 101. In addition, the USPTO, the software industry and the federal courts are unsure of the optimal disclosure and claim style required for software applications. For this reason, a broad mix of technical figures, explanatory text, and claim styles must be integrated into one software patent application to ensure that the software patent has a decent chance of surviving licensing and litigation opposition.

In recent years, the software patent case law and USPTO practice have indicated an increased willingness to allow software to be patentable.<sup>15</sup> With these changes, the current issues facing the software patent practitioner become "what type of claims should I draft for my client's software invention?" and "how will these claims affect both the prosecution of the application and the client's financial

and business interests?" Emerging software patent law has opened the door for new, extremely advantageous types of software claims.<sup>16</sup> As this new law takes hold, it may be helpful for an attorney to focus on drafting software claims which are designed to overcome the problems associated with contributory infringement suits and increase the royalty base of the software patent. It may also be advantageous for an attorney to draft software claims that capture a larger number of potential infringers for the client, while optimizing the client's position in patent licensing or litigation. At the very least, a few claims in each software patent should be dedicated to these very valuable goals, even at the risk of raising a 35 U.S.C. § 101 rejection under USPTO examining procedures and federal court holdings.

What all of this means for the software patent practitioner is that already complicated software patent drafting may become even more complicated, albeit advantageous for the software industry, by virtue of new CAFC case law and USPTO software patent examination policy. Understanding software claim drafting strategies, styles, and emerging opportunities may go a long way to reducing practitioner anxiety, drafting time, prosecution time, and client discomfort with her software patents.

### C. Overview

This comment discusses the advent of "conventional" software claims and their strengths and weaknesses. Conventional software claims include structure claims, process claims with physical step limitations or field of use limitations, and means-plus-function claims. The comment explains why conventional software claims are not capable of fully preventing or deterring some competitors from making, using, offering to sell, selling or importing the client's patented software invention in the modern software market. This comment then introduces and justifies article of manufacture claims, method of manufacture claims, and data structure claims as valuable supplemental software claims over and above conventional software claims. The set of software claims introduced are further analyzed in terms of patent procurement costs, overall patent portfolio strategy, patent specification disclosure, and software prosecution methodology with the hope of shedding light on the software patent prosecution experience. In many cases, software patent practitioners will find that these three new types of software claims are essential in order to protect fully the rights of the individual software inventor or software corporation.

## II. CONVENTIONAL STRUCTURE AND PROCESS CLAIMS

In order to comprehend the inherent limitations and strengths of conventional software claims, one must look to the historical advent of conventional software claims and their evolution. This section explores the early software patent case law which led to the use of structure and process claims as conventional formats for software inventions. Examples of these claim formats are provided to illustrate the advantages of using such claims.

### A. The Early Software Patent Case Law

#### 1. *GOTTSCHALK v. Benson: The First Modern Software Patent Decision*

Software became viable as an independent industry in the late 1960s to early 1970s.<sup>17</sup> In 1972, the Supreme Court decided its first case dealing directly with the issue of software patentability. In *Gottschalk v. Benson*,<sup>18</sup> the Court held that method claims reciting a process for transforming a binary coded decimal (BCD) value to a binary value were unpatentable subject matter under 35 U.S.C. § 101. This nonstatutory determination was made even though claim 8 of Benson's application contained hardware limitations in the form of a recited "reentrant shift register."<sup>19</sup> In the wake of *Benson*, software patent practitioners adopted software structure claims (e.g., electrical hardware claims reciting structure such as a CPU, memory, display screen and circuitry) as the standard claim style for patenting software inventions. Software process claims were nearly entirely abandoned. The rationale for this widespread adoption of the software structure claim are quite clear once the text and reasoning of *Benson* are scrutinized.

In *Benson*, the Court was concerned that the claimed process was "not limited to any particular art or technology, to any particular apparatus or machinery, or to any particular end use."<sup>20</sup> The Court found that a broad patent on a mathematical formula was bad for the public, and should be limited to some tangible and specific machinery.<sup>21</sup> The Court further stated that "[w]hile a scientific truth, or the mathematical expression of it, is not a patentable invention, a novel and useful structure created with the aid or knowledge of scientific truth may be."<sup>22</sup> It is therefore easy to understand why software inventors and attorneys would believe that hardware-laden computer structure claims would be preferable over a broad math-like process software claim for software inventions in light of this interpretation of 35 U.S.C. § 101. The *Benson* Court believed that "[i]f there is to be invention from such a discovery [of nature], it must come from the application of the law [of nature] to a new and useful end."<sup>23</sup> This rationale reinforced the Court's desire for structural limitations in software claims before allowing software to become patentable subject matter under 35 U.S.C. § 101.

In addition to favoring structural limitations to avoid the patenting of pure ideas and math, the *Benson* Court viewed software as an

intangible law of nature in a manner similar to the intangible law of gravity.<sup>24</sup> The Court seemed to follow a line of reasoning wherein gravity is not patentable subject matter, but the pendulum which simply relies on gravity for proper operation is patentable subject matter. The difference between gravity and the pendulum is that the former is math-like and intangible while the latter is a tangible apparatus.<sup>25</sup> It was therefore easy for the Court in *Benson* to draw a similar conclusion between the intangible software concepts of the early 1970s and the tangible computer hardware systems of that same period. The result in *Benson* was that any machine, apparatus, or computer implementation including software as a component is more likely to be patentable subject matter as a novel structure under 35 U.S.C. § 101 than nature-like software concepts. The *Benson* Court stated, "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."<sup>26</sup> The *Benson* Court clearly wanted to insert structural limitations into software technology because the Court construed software as falling into the realm of laws of nature, mental processes, and abstract ideas.

Although the method at issue in *Benson* was found to be nonstatutory, the *Benson* Court did not hold that software could never be patentable: "It is said that the decision precludes a patent for any program servicing a computer. We do not so hold."<sup>27</sup> The Court clearly sent the message that software would be more deserving of patentable subject matter status if software practitioners disclosed a large amount of hardware along with the software and included substantial hardware limitations in the claims. Such specifications would render the claims less like laws of nature, less like abstract ideas, and less like a mathematical algorithm. Therefore, the result from the holding in *Benson* was largely a form over function change in patent drafting.

Following *Benson*, clever software practitioners and inventors of the 1970s realized that software could not be totally abstract and void of hardware. The technique commonly employed was to avoid stating anything about "algorithms" or "software" in a specification and simply show the software as a hardware system both textually and graphically. It became difficult to distinguish software patents from hardware patents unless the patents were scrutinized in some detail. Software patent practitioners soon discovered that they could disguise software innovations as hardware inventions by disclosing significant computer hardware details along with the software code within the patent specification. The software had to be linked to some computer structure or memory system in order to operate. An applicant therefore could disclose and claim this structure without creating a serious claim limitation. With this required hardware in the software claims, the federal courts or the USPTO would be more likely to find the software claims protectible under 35 U.S.C. § 101.

Following this line of reasoning, the software patent practitioner of the 1970s determined that she needed to illustrate and claim the software with a central processing unit (CPU) to execute instructions; memory (either magnetic tape, a magnetic drum, magnetic disks, CDs, optical storage, RAM, ROM, EEPROM, EPROM, flash memory, and/or like storage media) to store executable code and data; and conductive interconnects to allow portions of the software to communicate with other portions of the software and to allow the CPU to execute the software that is resident in memory.<sup>28</sup> In other circumstances, depending upon the type of invention, the practitioner would illustrate the software as including further hardware components such as a printer, modem, keyboard, mouse, display screen, disk drive, register, sensors, motors, controllers, machinery, assembly line, or some other tangible object in order to properly process information, manufacture items, receive input, provide output, or execute software code. Claiming these tangible structural items via a structure software claim format rendered the mysterious and intangible software subject matter statutory as an apparatus under the *Benson* rationale. Any determination that these claims were nonstatutory under 35 U.S.C. § 101 would have invalidated tens of thousands of electrical systems, circuits, and like patents consistently issued by the USPTO for decades. In summary, *Benson* forced structural limitations into both process and structure software claims, and discouraged math-intensive software inventions. Residue from these two themes still exists in current federal court opinions and current USPTO examination practices.<sup>29</sup>

## 2. PARKER v. Flook: Case Law in the Wake of Benson

In *Parker v. Flook*,<sup>30</sup> the progress of software patenting seemed to worsen. *Flook* reinforced the hardware structure claim as a predominant claim format for U.S. software patenting. The case concerned software method claims which recited the general steps of: (1) measuring the present value of a process temperature variable; (2) using a mathematical equation or algorithm to calculate an updated alarm-limit value; and (3) adjusting the actual alarm limit to the updated value.<sup>31</sup> The Court found that the steps of measuring and using a mathematical equation did not contain patentable subject matter under 35 U.S.C. § 101, and that post-solution physical steps of storing the alarm limit to some physical location was not enough to render the claim statutory under 35 U.S.C. § 101.<sup>32</sup>

In keeping with *Benson*, the Court wrote in *Flook*, "[T]his Court has only recognized a process as within the statutory definition when it either was tied to a particular apparatus or operated to change materials to a 'different state or thing.'"<sup>33</sup> This reinforced the notion that only process claims with substantial physical limitations and/or structural hardware claim limitations would be patentable under 35 U.S.C. § 101 for software innovations. The Court further held that post-solution activity in a software process claim, even if physically limiting in substance, would not render the software claim patentable: "[t]he notion that post-solution activity, no matter

how conventional or obvious in itself, can transform an unpatentable principle into a patentable process exalts form over substance."<sup>34</sup> By arriving at this conclusion, the *Flook* Court made it even more difficult to obtain a software patent in the U.S.

*Flook* also implies that hardware or physical claim limitations other than the software need to be novel to survive 35 U.S.C. § 101. The Court determined that if the point of novelty was present only in the algorithm or software, and not in the overall process, then the claim would not be patentable subject matter under 35 U.S.C. § 101.<sup>35</sup> The Court further stated, "We think this case must also be considered as if the principle or mathematical formula were well known."<sup>36</sup> By assuming that the software limitations or algorithm portions in a software claim are well known in every software innovation, no software would ever be classified as patentable unless accompanied by novel hardware. If there is novel hardware, why would anyone bother to limit the already novel hardware with unneeded software limitations that would be given no patentable merit in the USPTO but would be construed as limitations to infringement by the courts?

The *Flook* Court made it more difficult to obtain a software patent in other ways as well. In addition to the constraints on patentable subject matter under 35 U.S.C. § 101, the software claim must also adhere to 35 U.S.C. §§ 102 (novelty) and 103 (nonobviousness) before being considered as patentable subject matter.<sup>37</sup> The *Flook* majority states that "[t]his case turns entirely on the proper construction of § 101 of the Patent Act . . . . It does not involve the familiar issues of novelty and obviousness that routinely arise under §§ 102 and 103 when the validity of a patent is challenged."<sup>38</sup> The majority seems to suggest that issues of novelty and obviousness will not be visited to determine patentable subject matter under 35 U.S.C. § 101. However, the *Flook* majority then contradicts itself by saying, "Respondent's process is unpatentable under § 101, not because it contains a mathematical algorithm as one component, but because once that algorithm is assumed to be within the prior art, the application, considered as a whole, contains no patentable invention."<sup>39</sup> Assuming portions of the claim are in the prior art brings novelty and obviousness into the determination of patentability under 35 U.S.C. § 101. In other words, the *Flook* Court believed that a 35 U.S.C. § 101 determination must include some aspects of novelty under 35 U.S.C. §§ 102 and 103.<sup>40</sup>

In summary, the 35 U.S.C. § 101 hurdle for software patent claims was significantly raised by the decision in *Flook*. Software process claims began to contain significant structural limitations far beyond simple post-solution activity in order to overcome the decision in *Flook*. Structural software patent claims became an even more useful tool for avoiding 35 U.S.C. § 101 rejections than software process claims of similar scope.<sup>41</sup> Most software patent practitioners of this period continued to draft apparatus and structure claims for software inventions which appeared at first glance to be hardware inventions in order to avoid an adverse 35 U.S.C. § 101 ruling. This response from patent attorneys was exactly what the Court was trying to avoid.

The Court did not want form over function to become the rule in software patents. In *Flook*, the Court stated that "[allowing implementation of a principle in some specific fashion] would make the determination of patentable subject matter depend simply on the draftsman's art and would ill serve the principles underlying the prohibition against patents for 'ideas' or phenomena of nature."<sup>42</sup> In other words, the Court did not want a claim with mechanically added non-novel post-solution activity to be patentable, while the same claim without the non-novel post-solution activity is rendered unpatentable under 35 U.S.C. § 101. The Court was trying to discourage clever attorneys from using their skill to hide software claims among a sea of irrelevant non-novel limitations.

In light of *Flook* and *Benson*, it was true that simple additions to claims would not fool the courts or the USPTO into believing that the claims were statutory under 35 U.S.C. § 101. Instead, in order to fool the courts and the USPTO, practitioners needed to hand-craft and custom-tailor the entire software patent application to look and feel like hardware. Practitioners no longer assumed that simple structural additions to select claim segments would suffice. For example, software patents needed hardware block diagrams of detailed computer schematics, electrical timing diagrams, and structural claims. "Software" was rarely mentioned in the specifications because "software" invited 35 U.S.C. § 101 rejections. One need only inspect the "software" patents of the late 1970s and early 1980s to see that the distinction between software and hardware was significantly blurred by hardware disclosure that was intended to avoid an adverse ruling under *Flook* and/or *Benson*.<sup>43</sup> Instead of just claim drafting becoming form over function, the entire art of software patent drafting became form over function far beyond the fears of the *Flook* Court. The *Flook* and *Benson* decisions rendered the entire operation of drafting a software patent application an exercise in form over function mastery, for which software clients would pay their attorneys dearly.<sup>44</sup> *Flook* marked the low point in software patenting, where the patent attorney either made the whole patent look, smell, and feel like hardware or the client lost software protection under 35 U.S.C. § 101.

### 3. Further Case Law in the Wake of *Benson* and *Flook*

About the same time *Flook* was decided, the Court of Customs and Patent Appeals (CCPA) rendered the use of other types of structural limitations in software claims ineffective to overcome 35 U.S.C. § 101 rejections. In *In re Christensen*<sup>45</sup> and *In re Richman*,<sup>46</sup> software claims containing structural limitations that simply provided operands, arguments, or inputs to an algorithm or software routine were rendered unpatentable. These holdings have received recent support from the 1989 decision in *In re Grams*.<sup>47</sup>

The holdings in *Flook*, *Christensen*, *Richman* and *Grams* suggest that when software claims contain "input-providing" hardware limitations and/or post solution activity "output" hardware limitations surrounding a software algorithm, these claims do not meet the requirements of 35 U.S.C. § 101. However, if the software claim is laden with several hardware limitations, physical effects, or limited-field-of-use language throughout, that claim becomes statutory.<sup>48</sup>

In *Diamond v. Diehr*,<sup>49</sup> the Court began to pull back from the holding in *Flook*, a process that is still continuing today. The *Diehr* Court determined that claims must be considered as a whole to determine patentability and that claims may not be considered unpatentable simply because they contain algorithms.<sup>50</sup> "It is inappropriate to dissect the claims into old and new elements and then to ignore the presence of the old elements in the [35 U.S.C. § 101] analysis."<sup>51</sup> Claims have a synergistic effect, so that if a claim contains elements A, B, C, and D, it is not proper to attack each of A, B, C, and D in isolation as four separate elements. The USPTO and the courts must view the claim as a whole under 35 U.S.C. § 101.

Further, the *Diehr* Court reaffirmed the general policy behind *Benson* by maintaining that pure ideas, laws of nature, and mathematical algorithms are not patentable,<sup>52</sup> suggesting again the necessity of claiming an object or other limitation that renders the software claim patentable as a whole. The Court did not directly state what objects or limitations are required in the software claims to render the software claims patentable, but did indicate what claim strategies will not overcome a 35 U.S.C. § 101 rejection. *Diehr* stated that merely limiting an algorithm or software innovation to a specific "technological environment" or adding insignificant post solution activity to a software claim will not be enough to overcome a 35 U.S.C. § 101 rejection.<sup>53</sup> Thus, while *Diehr* pulled back from *Flook* in that claims were to be considered as a whole and not subject to 35 U.S.C. §§ 102 and 103 criteria, *Diehr* still held that significant additional physical or structural content over and above the software algorithm must be included in the software claim to render the software invention patentable.<sup>54</sup> Therefore, *Diehr* further reinforced the use of structure claims or apparatus claims for software inventions along with the use of process claims with significant physical limitations.

About the time *Diehr* was decided, the CCPA and other federal courts began using a two-part test for determining the patentability of software claims under 35 U.S.C. § 101. This test became known as the *Freeman-Walter-Abele* test<sup>55</sup> and is still recited with some reverence today in some USPTO rejections of software patent applications and in federal court opinions. The *Freeman-Walter-Abele* test involves two parts:

It is first determined whether a mathematical algorithm is recited directly or indirectly in the claim. If so, it is next determined whether the claimed invention as a whole is no more than the algorithm itself; that is, whether the claim is directed to a mathematical algorithm that is not applied to or limited by physical elements or process steps. Such claims are nonstatutory. However, when the mathematical algorithm is applied in one or more steps of an otherwise statutory process claim, or one or more elements of an otherwise statutory apparatus claim, the requirements of section 101 are met.<sup>56</sup>

The *Freeman-Walter-Abele* test did little to change the software patent practice. Once again, practitioners got around an adverse decision under the *Freeman-Walter-Abele* test by drafting apparatus or structure claims or process claims with structural or physical limitations.

#### *4. Summary of the Response to the Early Software PATENT Case Law*

The cases discussed above reflect nearly two decades of patent case law and USPTO policy. These cases resulted in software patent practitioners drafting significant structural and physical computer limitations in both software structure claims and software process claims. This extreme bias toward the need for structural limitations in software patents is still hampering software patent practitioners today. *Benson*, *Flook*, *Diehr*, and the *Freeman-Walter-Abele* test either are currently in full force or have not been totally overruled as of this writing. For instance, a software process claim which contains significant structural limitations may still be rendered nonstatutory subject matter under the current USPTO Examination Guidelines.<sup>57</sup> If the claim "encompasses any and every computer implementation of a process," the claim will be analyzed as a process claim (i.e., the structure may very well be ignored).<sup>58</sup> This is in line with case law holdings such as those involving the doctrine of post-solution activity, which have even rendered software claims unpatentable under 35 U.S.C. § 101 in spite of structural limitations.<sup>59</sup> Process claims will only be statutory if the claim recites "a physical transformation outside the computer" or is "limited by the language in the claim to a practical application within the technological arts."<sup>60</sup> Thus, the current USPTO Examination Guidelines indicate that even today significant structural limitations must be used in software claims to avoid a 35 U.S.C. § 101 battle. Because of the above case law, the drafting of software structure claims and process claims that contain significant structural limitations or physical constraints has been firmly entrenched in software patent prosecution.<sup>61</sup>

## B. Software Structure/Apparatus Claims

### 1. EXAMPLES OF SOFTWARE STRUCTURE/APPARATUS CLAIMS

In order to draft effective structure claims that avoid 35 U.S.C. § 101 rejections in accordance with the pertinent case law, it is best to study a few structure/apparatus claims and structure claim strategies from issued U.S. patents.

#### a. A First Style for Claiming the Algorithm as a Computer Structure

One typical type of software structure claim, the hypothetical machine claim,<sup>62</sup> recites the software subroutines or software functions as individual objects, as though pieces of the software themselves are tangible. These claims may be interpreted in light of the specification. The lack of enough hardware detail disclosed in the specification may result in a loss of the claim under 35 U.S.C. § 101. Because the claim recites software in a structural form, it may not fool the USPTO. This may result in the claim being classified as an abstract idea, making the claim per se nonstatutory.<sup>63</sup> Therefore, a good practice for the software patent practitioner is to disclose a lot of structure and hardware diagrams in the application to avoid this result.

One example of this type of software structure claim is found in claim 19 of U.S. Patent No. 5,303,146 (1994). This claim recites pure software in a manner that makes intangible items, like a spreadsheet on a computer screen and software code, seem tangible:

Structure Claim Example 1. A system for managing different versions of a *data model*, the system comprising:

an electronic *spreadsheet* for expressing the *data model* as a plurality of *interrelated information cells*;

*means* for specifying a *reference set* of said *information cells* which serves as a *base case* for the *data model*;

*means* for creating a *new version of the data model* by modifying at least one of said *information cells* in the *reference set*; and

*comparison means* for determining which ones of the *information cells* in the *new version* have changed from the *reference set*.

All of the italicized references in the claim above recite limitations that are either pure software or are arguably intangible abstract concepts. This method of drafting software claims, while potentially troublesome under 35 U.S.C. § 101, should not be overlooked. The practitioner can always argue that the limitations of "electronic" and "system" in the above claim render the claim, when considered as a whole, patentable subject matter under 35 U.S.C. § 101 in view of *Flook*<sup>64</sup> (there is more than just post solution activity), the *Freeman-Walter-Abele* test<sup>65</sup> (the claim is more than the algorithm itself), and *In re Alappat*<sup>66</sup> (a new machine is recited). This method of drafting structure claims is also more valuable when the patent specification discloses some hardware diagrams initially. These hardware diagrams allow the patent to be read on either software or hardware embodiments to whatever extent is necessary to avoid invalidation of the claims in the federal courts under 35 U.S.C. § 101, while still rendering useful software claim protection. The main strategy here is that the patent attorney will attempt to read the intangible or abstract objects as broadly as possible (i.e., on software alone), and limit only certain strategic terminology in the claims to structural embodiments and limitations in the specification as a last resort to avoid adverse 35 U.S.C. § 101 rulings.<sup>67</sup>

Another example similar to the above claim style is taken from claim 6 of U.S. Patent No. 5,337,233 (1994):

Structure Claim Example 2. An apparatus for preparing language text to be used by a text processing system, where said language comprises more than 256 characters, said apparatus comprising:

a) a filter device for capturing an input stream of characters which represent said language comprising;

a word filter for separating said input stream of characters into strings of characters which represent words, wherein said word filter comprises

a grammar analyzer to facilitate the separation of said characters into strings of said characters which represent words, wherein no special characters in said input stream of characters is required to distinguish one word or character set from another, and wherein character strings comprising more than two bytes of data are called compound words;

b) a mapping device coupled to said word filter, for mapping said strings of characters which represent words into unique strings of single-byte ASCII characters; and

c) an output device coupled to said mapping device, for passing said unique strings of single-byte ASCII characters which represent words to said text processor.

Once again, the structure-sounding limitations in this claim (e.g., "input stream," "grammar analyzer") may be read on pure software (which a court may state is intangible), or alternatively may be read on disclosed hardware block diagrams if proper support for both is provided in the specification. The hardware diagrams therefore function as a safe harbor to which the software patent practitioner can turn either in licensing, litigation or prosecution to render a claim patentable.

In addition, when an Examiner is being tough under 35 U.S.C. § 101, the applicant can argue that there are several direct structural limitations in the above claim, such as "output device," "text processor," and "two bytes of data" (arguably memory locations). Such an argument is always available to a patent practitioner who discloses significant structure in the specification, making the normally abstract items no longer intangible. For example, the practitioner may argue that the normally intangible item is instantiated in memory of a specific CPU illustrated in the figures, or that it is stored in a cache within a CPU. These limitations can always be added without battling problems of new matter.<sup>68</sup> Under the new matter doctrine, anything that is not described in the original specification is new matter.<sup>69</sup> However, when the material added to a patent is inherently disclosed by the application,<sup>70</sup> the material does not constitute new matter.<sup>71</sup> It is therefore always a good idea when using structure claims, as well as process claims, to provide many structural fall-back positions in the specification even if they may never be needed during prosecution.

#### b. A Second Style for Claiming the Algorithm as a Computer Structure

Another approach for drafting software structure claims provides an alternative to reciting portions of the software code itself as structure. Under this approach, a claim may recite the conventional and widely used structure of a computer which executes the novel software from memory locations. In the alternative, if the software is executed in a larger system containing more hardware than just CPU and memory, the software patent practitioner may claim the larger hardware system. Accordingly, the claim will recite a CPU or computer, memory, peripherals, and/or other computer or system technology, and then recite that the memory coupled in the computer system contains novel software that is executed within the computer system.

An example of this claim type is taken from claim 1 of U.S. Patent No. 5,461,488 (1994):

Structure Claim Example 3. A data processing system for processing facsimile (FAX) transmissions, the data processing system comprising:

a serial communication device having an input for receiving facsimile data from a serial communication line and an output for providing the facsimile data;

a control computer coupled to the output of the serial communication device;

a default computer which is coupled to the control computer for receiving the facsimile data if the data processing system cannot correctly determine where to route the facsimile data, a memory portion coupled to the control computer comprising:

software for receiving the facsimile data from the serial communication device and storing the facsimile data in a first data file in a first format;

software for converting a portion of the first data file to a second format; and

software which uses the second format to determine where to route the facsimile data within the data processing system; and

wherein at least one of the software for receiving, the software for converting, or the software which uses the second format interfaces with control software to keep a log file, the log file storing information regarding the facsimile data, the information being selected from a group consisting of: facsimile data length, time of receipt of the facsimile data, time of routing of the facsimile data, a destination to which the facsimile data was routed, and where or who the

facsimile data was transmitted from.

This claim is significant because it actually mentions "software" as part of its claim language. In the pre-1994 time period, the mere mention of "software" created trouble under 35 U.S.C. § 101. This first claim indicates how much times have changed in software patenting.

A second example of a useful claim approved by the CCPA is provided by *In re Noll*:<sup>72</sup>

Structure Claim Example 4. A computer graphics system for displaying in a multi-line, multi-point-per-line format images corresponding to a sequence of input display commands comprising

(A) a programmable data processor operating under the control of a program to convert said display commands into data entries in an array of multi-bit data words, each entry is said array corresponding to a discrete point in the image to be displayed,

(B) a scanned-raster display device for generating illuminated points on a display surface in response to applied data signals and

(C) means intermediate said data processor and said display device and cooperating with said data processor for sequentially accessing said words in said array for presentation to said display device.

This claim is an older claim that can be compared to the first claim provided above. This claim is indicative of the type of claims that were allowed during the period of hostility against software patents in the 1970s. The structure claim examples above illustrate alternate ways of reciting claims so that the combined barriers of *Benson*, *Flook*, *Diehr*, and the *Freeman-Walter-Abele* test can be readily overcome in both the USPTO and the federal courts.

## 2. Advantages of Software Structure Claims

Software structure claims, a few of which have been illustrated above, continue to be a popular claim choice for software patent practitioners. These claims appear in roughly half of all software patents issued in the U.S.<sup>73</sup> The value of software structure claims is that these claims are easy to interpret or amend to overcome 35 U.S.C. § 101 barriers in either the licensing, litigation, or prosecution of the patent.<sup>74</sup> In addition, having allowed claims when on appeal to the USPTO Board of Appeals or the CAFC is a definite tactical advantage, since a patent with a few allowed claims is more likely to be issued in some form than a patent with all claims rejected. The software structure claims are also most likely to be allowed as statutory subject matter under 35 U.S.C.

§ 101 earlier in the prosecution history. From personal experience, in some cases, structure claims are allowed long before process claims are allowed under 35 U.S.C. § 101. Therefore, structure claims can be used in a relatively safe manner to test the specific USPTO Examiner assigned to your patent application. This strategy may allow a practitioner to determine easily the point at which this particular Examiner or art group is drawing the distinction between patentable subject matter and unpatentable subject matter under 35 U.S.C. § 101. The software structure claim may also be used to identify language which will favor 35 U.S.C. § 101 patentability, and to identify language that is best avoided for that particular art group or Examiner.

In summary, there are two preferred structural software claim types. The first type of structural software claim requires that the software practitioner break the software program into distinct functional pieces or blocks. Most programmers write code in easy-to-identify discreet segments. In fact, most programming languages used today are structured to reinforce this practice.<sup>75</sup> The practitioner first parses the program into the different code segments, function calls, procedures, etc., then drafts the claim to make these software routines appear as hardware. In essence, the different portions of memory that contain different software routines are each presented as separate physical elements. Each element includes at least a portion of the CPU and electrical busing as needed for proper software execution. For example, a software routine that adds is an "adder," a software routine that sorts is a "sorter," and a software routine that rasterizes is a "rasterizer." One way to get this point across to the inventor is to ask the inventor to design a customized CPU containing circuitry optimized to her algorithm. This block diagram is usually the block diagram of the functionality of the software and defines the boundaries for which the patent attorney is looking. This model for the software claim looks like the following:

1. A computer system comprising:

<first software segment described as an object>;

.....

<last software segment described as an object>.

The second type of software structure claim is the general software structure claim. This claim recites conventional computer technology, with the memory storing the software for execution on the computer system. The general format is as follows:

1. A computer system comprising:

a central processing unit;

<additional hardware disclosure for the circuitry coupled to the CPU, as is prudent>;

a memory unit comprising:

<insert novel software here>.

The use of one or both of these two structure claim formats is advantageous when battling 35 U.S.C. § 101 barriers in litigation, licensing, or prosecution.

### **C. Software Process Claims with Structural Limitations or Practical Application Limits**

Software process claims are the most common form of software claim found in recently issued U.S. software patents. These claims recite methods for performing some operation where the elements of a process claim are individual steps in the process. Software process claims can be found in roughly 85% of all issued U.S. software patents.<sup>76</sup> The following is a brief study of software process claims in issued U.S. patents. This study is intended to aid practitioners in drafting software process claims.

#### *1. Examples of Software Process Claims: claiming the algorithm as a method of operation*

The software process claims typically found in issued software patents contain either limitations which are structurally significant or those which restrict the software process to a practical real-world application, which is usually external to the computer system. Specifically, software process claims must be either: (1) drafted with substantial structural limitations; (2) drafted to contain physical post-processing activities; or (3) be limited in scope to a practical application within the technological arts (i.e., have some external physical use or effect outside of the computer itself).<sup>77</sup>

An example of a process claim that conforms to these limits is taken from claim 15 of U.S. Patent No. 5,339,424 (1994):

Process Claim Example 1. A method for compiling and editing a source program including a first program division written in a first programming language and a second program division written in a second programming language, comprising the steps of:

compiling said second program division to produce a second module,

providing said first program division with reference information for referencing said second module said reference information comprising an identifier assigned by an operating system to a first module obtained by compiling the first program division;

compiling said first program division to produce the first module and to obtain the identifier of the first module, and

registering said second module so that said second module can be referenced by using the identifier assigned to said first module by the operating system.

Notice that this claim recites structural limitations such as "module," "division," "identifier," and "information," which may simply be binary bits of memory. These structural limitations render this software claim statutory subject matter under 35 U.S.C. § 101 as interpreted by the case law previously discussed.<sup>78</sup>

Another software process claim found to be statutory under 35 U.S.C. § 101 comes from *In re Chatfield*.<sup>79</sup> This case illustrates that

significant structural limitations in a process claim may render the claim statutory under 35 U.S.C. § 101, even if they are contained only within the preamble:<sup>80</sup>

Process Claim Example 2. A method of operating a computing system upon more than one processing program concurrently for improving total resource utilization, said computing system comprising at least one central processing unit, having a logic and main memory function and an interrupt capability, and a plurality of peripheral resources capable of functioning in parallel with the central processing unit, comprising steps for:

(1) accumulating system utilization data for at least one processing program for at least one resource, said system utilization data comprising resource activity and/or resource degradation data;

(2)(a) at spaced intervals interrupting the processing programs and analyzing the system utilization of at least one processing program;

(2)(b) based on this analysis regulating resource access by assigning an individual resource access priority and/or preventing resource access altogether in an unlike manner to at least two resources for at least one processing program to increase throughput;

(3) resuming the operation of the computing systems on the processing programs; and,

(4) continually repeating steps (1) to (3).

The above claim from *Chatfield* is of a form that will be allowed over 35 U.S.C. § 101 as interpreted by case law discussed above.<sup>81</sup>

The following example, taken from claim 1 of U.S. Patent No. 4,888,683 (1989), illustrates that a limitation of "memory" throughout a software claim may be enough to render a claim statutory subject matter under 35 U.S.C. § 101:<sup>82</sup>

Process Claim Example 3. A method for loading a program in a distributed processing system including a plurality of information processing units interconnected by a transmission system, comprising the steps of:

storing at least one program and its identifier in a memory of at least one of said information processing units;

retrieving from said memory at least one stored program based on information relating to a program to be loaded

transmitting said at least one stored program retrieved from said memory and its identifier; and

receiving a receiving program based on said identifier of said at least one stored program being transmitted and loading the received program in a memory in at least one information processing unit other than said information processing unit transmitting said stored program and its identifier.

The next example, taken from claim 37 of U.S. Patent No. 5,461,488 (1994), suggests that if an algorithm is limited to a field of technical use and an entire algorithm is not preempted, the USPTO will allow the claim:

Process Claim Example 4. A method for routing facsimile transmissions through a computer system, the method comprising the steps of:

(a) receiving a facsimile transmission wherein the facsimile transmission is received in a first data format;

(b) converting the first data format of the facsimile transmission to a second data format which is textual, the converting performing human language translation when needed;

(c) scanning the second data format in a textual manner to identify a matching alphanumeric string within the second data format which matches at least one predetermined string stored in memory, if no exact match is found between the second data format and the at least one predetermined string, the step of scanning performs an error analysis to find a most-closely associated string matching string to use as the matching string or assigns a default matching string as the matching string if error analysis passes a predetermined error threshold;

- (d) setting a facsimile destination based upon the matching string from step (c);
- (e) electronically communicating the facsimile transmission to the facsimile destination;
- (f) logging information from the steps (a) through (e) in a log file; and
- (g) repeating steps (a) through (f) for a plurality of facsimile transmissions.

The process claims described above are not presented because they are exceptional in form or scope. They are merely claims which have been found by a federal court or by the USPTO to be statutory under 35 U.S.C. § 101 due to a proper degree of structural limitations or application limitations. The differences in claim style and language illustrated by the examples above are often a matter of taste and not a matter of law.

## 2. Advantages of Software Process Claims

Using process claims in a software patent application has several advantages. It is easy for a software inventor and a software patent attorney to view software as a process rather than an apparatus. In many cases, the software engineer or software programmer who is the client will know a great deal about software development or programming while knowing little or nothing about hardware design. It is not unusual for the software inventor to fully understand portions of a patent which include pseudocode, process flowcharts, and a C code appendix, while on the other hand have minimal knowledge about computer structure, electrical block diagrams, hardware peripherals, and electrical/logic circuits. It does not take an auto mechanic to drive a car, and likewise, it does not take an electrical engineer to program a computer in a novel manner. Therefore, process claims are a natural prosecution choice for an inventor to fully understand and appreciate what is being claimed, and for the patent practitioner to work effectively with the software inventor.

From personal experience, software process claims are usually less likely to pass 35 U.S.C. § 101 muster than are software structure claims. Nonetheless, process claims provide good common ground for a practitioner to understand the technical details of the software invention, while allowing the inventor a first exposure to claim drafting. Process claims may also offer the advantages of not requiring notice and/or marking in order to obtain a full 6-year statute of limitations for infringement damages.<sup>83</sup> Therefore, software process claims are useful additions to most software patent applications.

## III. Conventional Means-Plus-Function Claims

### A. The Case Law Approving of Means-Plus-Function Claims for Software Innovations

The CAFC recently approved the use of means-plus-function claims<sup>84</sup> within software patent applications.<sup>85</sup> The elements of a means-plus-function claim will be of the form "means for <insert function/action performed>." Historically, the USPTO attempted to view the means-plus-function claim as a mere permutation of a process claim format.<sup>86</sup> Accordingly, the Examiner would state that the claim recited an "algorithm" and reject the means-plus-function claim outright.

However, in *Iwahashi*, the court clearly indicated that any process can be termed an algorithm and that this categorization should not bar patentability.<sup>87</sup> The court in *Iwahashi* also wrote that means-plus-function claims must be analyzed as a whole, and that "claims shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof."<sup>88</sup> Thus, the claim at issue in *Iwahashi* was interpreted to read on the ROM structure disclosed in the specification.<sup>89</sup> Accordingly, the structural support in *Iwahashi*'s disclosure was enough to warrant a finding of patentable subject matter under 35 U.S.C. § 101.

In a similar holding, the CAFC further approved of means-plus-function claims in software patent applications in *In re Alappat*.<sup>90</sup> In *Alappat*, the court held that "the appealed decision should be reversed because the appealed [means-plus-function] claims are directed to a 'machine' which is one of the categories named in 35 U.S.C. § 101."<sup>91</sup> The court continually noted that novel software on a known computer creates a "new machine" and this "new machine" is statutory subject matter under 35 U.S.C. § 101.<sup>92</sup> For example, the court argued that "[a]s is evident, claim 15<sup>l</sup><sup>93</sup>] unquestionably recites a machine, or apparatus, made up of a combination of known electronic circuitry elements."<sup>94</sup> The court further explained that "claim 15 would read on a general purpose computer programmed to carry out the claimed invention" and "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."<sup>95</sup>

Many software practitioners applauded the decision in *Alappat* as a sweeping affirmation that software was indeed patentable.<sup>96</sup>

However, much of the language in *Alappat* seems to suggest that the machine and software are together patentable as a combination, but the software standing alone, even though novel, is not patentable without the "new machine" limitation. *Alappat* clearly can be interpreted to suggest that any means-plus-function claim in a software application is limited to the specific hardware implementation illustrated in the patent figures or disclosed in the patent specification. *Alappat* can also be interpreted as allowing means-plus-function claims to read on software only when the software is installed on a computer system (i.e., the "new machine"), even though the software is novel independent of the computer technology used.<sup>97</sup> Under this interpretation, many currently issued software means-plus-function claims would arguably not be capable of being read directly on software products or articles of manufacture. This interpretation would have an adverse impact in terms of royalty base, contributory infringement, and the like, which will be addressed later in this paper.<sup>98</sup>

## **B. *Alappat*-Type Means-Plus-Function Software Claims**

### *1. Examples of MEANS-PLUS-FUNCTION SOFTWARE CLAIMS*

*Alappat*-type means-plus-function software claims contain a preamble followed only by two or more means-plus-function clauses that describe the software in a functional manner.<sup>99</sup> Because the means-plus-function claim provided a convenient way in which to hide the intangible software invention or abstract software concepts in a structural form, means-plus-function claims were commonly found in patents issued by the USPTO long before the *Alappat* decision was published.<sup>100</sup> The following is claim 15, found statutory under 35 U.S.C. § 101 in *Alappat*.<sup>101</sup>

Means-Plus-Function Claim Example 1. A rasterizer for converting vector list data representing sample magnitudes of an input waveform into anti-aliased pixel illumination intensity data to be displayed on a display means comprising:

- (a) means for determining the vertical distance between the endpoints of each of the vectors in the data list;
- (b) means for determining the elevation of a row of pixels that is spanned by the vector;
- (c) means for normalizing the vertical distance and elevation; and
- (d) means for outputting illumination intensity data as a predetermined function of the normalized vertical distance and elevation.

Another *Alappat*-type means-plus-function claim is provided by claim 39 of U.S. Patent No. 5,461,488 (1994):

Means-Plus-Function Claim Example 2. An electronic facsimile communicator comprising:

means for receiving a facsimile transmission and storing the facsimile transmission in a first data format;

means for transforming the first data format to a second data format;

means for translating the second data format or the first data format from a first human language to a second human language wherein: (1) the first human language is a language used in the facsimile transmission; (2) the first human language is not capable of being scanned to determine a recipient; and (3) the recipient of the facsimile data cannot understand the first language but can understand the second human language;

means for scanning the second data format to determine the recipient of the facsimile transmission out of a plurality of potential recipients in a local area network, if no direct recipient is determined, a default recipient or a recipient identified by the means for scanning as being the most likely intended recipient of the facsimile transmission is set to be the recipient;

means for electronically routing the facsimile transmission to the recipient chosen from the plurality of potential recipients by the means for scanning; and

means for storing log data to keep a history of past electronic routings of facsimile data.

### *2. PRACTITIONERS' USE OF MEANS-PLUS-FUNCTION CLAIMS*

Software practitioners initially used *Alappat*-type means-plus-function claims to hide an algorithm in a structure-like claim format. Their hope was that the means-plus-function claim would read on the pure software and subroutines of the software without the need for the software to be installed on a computer before infringement occurred. *Alappat* claims are easy to draft in a broad and sweeping manner. For example, if the novel software contained the three subroutines A, B, and C, one would simply draft a claim reciting an apparatus comprising: means plus function doing A, means plus function doing B, and means plus function doing C. The practitioner usually hoped that the means-plus-function claim would read on the pure software. However, to be safe, most experienced practitioners typically disclosed electrical block diagrams and the like in the specification, where the means plus function doing A, B, and C were different segments of programmed memory, possibly along with portions of a CPU or specifically disclosed hardware. In some cases, a specially designed hardware system or custom-designed CPU could be illustrated by the software specification to support further means-plus-function claims. In essence, means-plus-function claims were used as a special multipurpose structure claim which practitioners hoped would transcend the conventional structural limitations of the structure claims.

Practitioners who include sufficient structure and hardware diagrams in their figures and detailed descriptions in their specification are now found to have statutory means-plus-function software claims that are most likely limited in scope to the disclosed hardware and equivalents thereof by virtue of the "new machine" doctrine of *Alappat*.<sup>102</sup> Those who do not include significant structure in their software patent applications run a significant risk of losing their means-plus-function software claims under 35 U.S.C. § 101 for failure to comply with the "new machine" requirements of *Alappat*, *Trovato*, and *Iwahashi*.<sup>103</sup> In light of these cases, means-plus-function software claims are strongly tied to disclosed structural features. Accordingly, these claims arguably obtain no more patent coverage, and possibly less coverage, than conventional software structure claims.

#### **IV. LIMITATIONS OF CONVENTIONAL SOFTWARE CLAIMS**

Even though the conventional software claim styles discussed above have some advantages, these claim styles may create significant problems if an applicant relies solely upon them in her patent application. Some problems include loss of scope of infringement, loss of royalty base, less patent value in licensing, loss of remedies against foreign companies, being forced into contributory infringement positions as opposed to direct infringement positions, and loss of remedy in the absence of direct infringement. This section discusses these problems and suggests solutions via creative claim drafting techniques.

##### **A. Conventional Software Claims Are Less Likely To Be Infringed**

The numerous court opinions discussed above require either significant structural limitations or narrow/specific application for conventional software claims. This is true not only for structure and process claims, but also for *Alappat*-type means-plus-function claims. The holding in *Alappat* clearly could be construed to limit the "means" recited in the claims to the structural disclosure of the specification, consequently giving no more coverage than software structure claims in many circumstances.<sup>104</sup> These limitations decrease the scope of the software innovation, thereby rendering conventional software claims less likely to be infringed than an analogous hard-wired electronic design claim of similar scope and function. Accordingly, the limited scope for these types of claims results in lower royalties and an overall lower patent value for licensing. Therefore, the structure-heavy interpretation of software claims is by itself a disadvantage that software patent practitioners should avoid whenever possible.

##### **B. Contributory Infringement Limitations of Conventional Software Claims**

Conventional software claims also lead to inherent contributory infringement barriers that do not arise when most non-software U.S. industries assert U.S. patents. If a practitioner can avoid these contributory infringement barriers or reduce their severity, the value of the client's software patent will increase significantly.

###### *1. A HYPOTHETICAL SITUATION Between IBM and Microsoft*

To illustrate one of many possible contributory infringement problems with conventional software structure claims, software process claims, and software means-plus-function claims, please consider the following example. Suppose a first company manufactures both hardware platforms and software products for commercial use. This company, to show the reality of the problem, could be any one of Apple, IBM, HP, Motorola, Sun, AT&T, Texas Instruments, Intel, or many other foreign or domestic companies. Assume, for example, that the first company is IBM. Suppose that a second company manufactures only software (e.g., Microsoft, Lotus, Borland, Adobe, Stac or Netscape). Assume the second company is Microsoft.

IBM sells hundreds or thousands of hardware platforms or computers to a third company (assume GM for example). IBM has a significant number of hardware patents on the computers sold. IBM also has patents on various software packages, such as word

processors, OS2(tm), and like software that will execute on the computers sold to GM. All of these IBM software patents contain only structure claims, process claims with structural limitations, and means-plus-function claims, in accordance with federal case law. All of IBM's structure claims contain limitations like "data cache," "CPU," "control bus structures," "computer memory," and "electrical signal." The process software claims are likewise limited to physical applications, structure, and occurrences that can only result after installation of the software onto a computer platform.

Instead of buying the IBM software for the IBM computers, GM would like simply to buy the IBM hardware platforms from IBM and purchase the software from Microsoft. Therefore, GM buys Windows95(tm) instead of OS2(tm), Word(tm) instead of the IBM word processor, etc., and installs the Microsoft software on the IBM computers. Process steps such as sorting, reading, transmitting, compiling, writing, and encoding, are not performed by Microsoft but are performed by GM after the software is installed onto a computer. Assume that the installed and executed Microsoft programs infringe several of IBM's software patents and IBM sues Microsoft.

## *2. IBM Can Only Sue Microsoft For Contributory Infringement*

Microsoft states that it is not a direct infringer, even though direct infringement occurred when GM installed the program and/or executed it via a CPU. Microsoft argues it does not make any hardware and that it cannot infringe the hardware limitations in the structure claims until GM installs or executes the software. Accordingly, GM is the direct infringer of the software structure claims. Microsoft then argues that it is not a direct infringer of the process claims because operations like "sorting," "scheduling," "reading," and "transmitting" are performed by GM and not by Microsoft. Instead, Microsoft ships CDs and magnetic disks that GM purchased and subsequently installed on the IBM computers, which then performed these operations and infringed various IBM patents. Therefore, contributory infringement and not direct infringement must be proved to find that Microsoft infringed IBM's patents.<sup>105</sup>

The simple threat of a more complex contributory infringement suit may reduce a patent's value in licensing, reduce royalty base, disrupt customer relationships, and in general be wholly undesirable. This is especially true if we consider that the whole issue could be diffused by the patent drafter for roughly a \$100 increase in filing fee.<sup>106</sup> By adding one or more of the claims discussed herein for the roughly \$100 fee (plus the cost of drafting), direct infringement theories can be more easily pursued with an increased likelihood of success.<sup>107</sup> However, the loss in this example is far greater than the cost of increased litigation fees, inconvenience or reduced patent revenues. Under contributory infringement, the plaintiff must show: (1) knowledge on the part of the defendant; (2) direct infringement; and (3) that the Microsoft algorithm or software in suit is not staple and has no substantial noninfringing uses.<sup>108</sup> All three of these elements pose additional hurdles that IBM must now overcome in a patent suit. The cost of addressing these issues in licensing or litigation will be far more expensive than a \$100 claim drafted by a practitioner. Furthermore, these contributory infringement issues may result in further revenue losses to the patent holder due to a weakened settlement position or an adverse judgment.

## *3. Microsoft's Arguments Against Contributory Infringement*

Microsoft can formulate many arguments by which it may avoid liability under a contributory infringement theory. Microsoft may argue that GM purchased the computers from IBM, and therefore, theories of implied license,<sup>109</sup> exhaustion,<sup>110</sup> and implied warranty of fitness/merchantability<sup>111</sup> dictate that GM cannot be liable for infringement of those software patents owned by IBM that contain structural limitations. The structural limitations, such as "data cache," "CPU," "control bus structures," physical steps, and so on, in the IBM software claims could not possibly be directly infringed by GM. As long as one limitation in a claim is not infringed by GM, there is no direct infringement. If there is no direct infringement, there is no contributory infringement.<sup>112</sup> Accordingly, under this argument Microsoft may have no liability since the IBM patents contain only structure claims, process claims containing structural limitations, and means-plus-function claims (arguably limited to hardware as discussed above). This result hardly seems equitable for IBM, given U.S. patent policy.

Microsoft may also pursue another theory. As software becomes more of an integral part of our society, some algorithms and software creations may possibly become staple or commonplace over time. Consequently, these algorithms or software creations may have substantial noninfringing uses. Think of algorithms and creations such as the pointer, the window, a spreadsheet, a merge sort, do loops, a binary search, ASCII, FFT algorithms, the hash table, object oriented concepts, Internet communication protocols, the tree structure, the array, and encoding, that have become commonplace and used in a variety of different software settings. All of these items were once an invention, but have over time arguably become staple or commonplace in either all software programs or large portions of the software industry. Are common software inventions such as these now as common as nails in a picket fence?<sup>113</sup>

## *4. IBM's arguments For Contributory Infringement*

IBM will assert several arguments to rebut the above theories of Microsoft. For instance, the purchase price was not a reasonable royalty for the many thousands of infringing copies of software that Microsoft sold. Also, a general purpose computer is capable of many uses other than that provided by Microsoft's software, and that for this reason a court should find contributory infringement. Further, IBM will argue that GM is liable for direct infringement and Microsoft is liable for contributory infringement under the doctrine of equivalents as applied in interpreting the means-plus-function claims.<sup>114</sup> Under this argument, *Alappat* merely stands for the proposition that if the means-plus-function claim reads on some type of structure in the specification, then the claim is unquestionably acceptable under 35 U.S.C. § 101. Then IBM will argue that the *Alappat* court did not intend to exclude pure software embodiments from the infringement scope of means-plus-function claim interpretation.<sup>115</sup> In the alternative, assuming IBM's software patents disclose both hardware block diagram figures and software flowcharts, even if *Alappat* limits the interpretation of the means-plus-function claims to hardware embodiments, the doctrine of equivalents extends the claimed "means" to the disclosed software.<sup>116</sup> IBM will attempt to argue that one of ordinary skill in the art would realize that the software and hardware solutions are interchangeable, and that one is equivalent to the other.

In response, Microsoft will take the position that the *Alappat* court did intend to limit the means-plus-function claims to the hardware embodiments (i.e., the "new machine") in the specification, and to allow the means-plus-function claim to read on pure software violates 35 U.S.C. § 101 and the policy of invalidating patents that read on abstract ideas. Further, Microsoft will state that when applying *Hilton Davis* to the contributory infringement issue, software and hardware are radically different from each other.<sup>117</sup> Any function-way-result argument will favor Microsoft because software is arguably very different functionally from hardware, and software is operated and developed in a substantially different manner.<sup>118</sup> Microsoft will therefore argue that there is no direct infringement of any of IBM's claims. Microsoft will claim that it has a complete defense and no liability under any interpretation of the claims in the IBM software patents.

Who would win in the above scenario is not readily apparent, even though Microsoft seems to have a stronger position because of the arguments made above. At any rate, the IBM software patents are reduced in value, harder to license, and more likely to be litigated by an accused infringer—all due to the exclusive use of only conventional structure, process and means-plus-function software claims.<sup>119</sup> The above hypothetical clearly indicates that conventional software claims may not adequately protect a software invention from infringing activity in a timely and cost-effective manner.

## **V. Article of Manufacture Claims**

An effective way for IBM and other companies like it to avoid both contributory infringement problems and case law requiring extensive structural limitations is to draft article of manufacture claims for software innovations. An article of manufacture claim for software is any claim style that allows the claim to be read onto computer-readable media without requiring a CPU or computer to be present, and without requiring that the software code be executed to be infringed. Article of manufacture claims read on software stored on a CD, floppy disk, tape, RAM, and like computer-readable storage media. In addition, software article of manufacture claims, if properly drafted, can cover the passive transmission of software on the information superhighway. Article of manufacture claims will consequently allow software corporations to assert direct infringement claims against competitors that may not be possible using conventional structure, process and means-plus-function claims. Article of manufacture claims for software innovations will also render a software patent more valuable in licensing, less likely to require litigation to prevent infringement, and less costly to enforce.

### **A. Introduction to *Beauregard* Software Article of Manufacture Claims**

Most people believe that article of manufacture claims for software achieved legitimacy for the first time in 1995 via the USPTO's position in *In re Beauregard*.<sup>120</sup> *Beauregard* is not judicial precedent, but it clearly indicates the USPTO's willingness to accept article of manufacture claims for software inventions in some form, even though subsequent USPTO policy may limit this form. The new software USPTO Examination Guidelines, released in January of 1996, indicate that article of manufacture claims may be allowed.<sup>121</sup> However, it will probably be a fight to get them issued.<sup>122</sup> Nonetheless, the theory that software embodied on a tangible storage medium is patentable subject matter under 35 U.S.C. § 101 is logically consistent with the reasoning used in previous CAFC cases.<sup>123</sup> If software creates a "new machine" simply by virtue of its presence within a computer storage area, then it must follow that computer code resident on a transportable storage media should likewise result in a statutory article of manufacture. After all, both the article of manufacture and the machine are enumerated specifically in 35 U.S.C. § 101 as permissible subject matter. Furthermore, the court in *In re Warmerdam*<sup>124</sup> determined that even though a claim (claim 1) was not statutory subject matter,<sup>125</sup> it became statutory when a dependent claim (claim 5) added a simple limitation of a "memory."<sup>126</sup> If "memory" is a critical limitation that renders non-patentable subject matter patentable, then a claim drafted to a computer storage memory or media containing software code should be patentable subject matter.

The debate regarding patentability of article of manufacture claims using *Alappat* and *Warmerdam* as precedent may now be moot

because the exchange in *Beauregard* is right on point with the issue of whether or not software article of manufacture claims constitute patentable subject matter.<sup>127</sup> Furthermore, the USPTO conceded that article of manufacture claims can be statutory subject matter in view of 35 U.S.C. § 101.<sup>128</sup> However, the issue still remains, to what extent will these claims be patentable through the USPTO?

## **B. Allowance of Article of Manufacture Claims Over 35 U.S.C. § 101**

The USPTO's guidelines for 35 U.S.C. § 101 restrictions on software article of manufacture claims are discussed in the recently published "Examination Guidelines for Computer-Related Inventions."<sup>129</sup> Unfortunately, the new guidelines are not crystal clear. The USPTO suggests that software claims that encompass any article of manufacture in a generic manner should be patentable under 35 U.S.C. § 101 only if the underlying recited process in the software claim is patentable.<sup>130</sup> For instance, the Examiner must look to physical constraints or limited field of use in the process portion of the claims for a favorable subject matter determination.<sup>131</sup> The USPTO states, however, that any article of manufacture claim that is limited to a specific manufacture is statutory regardless of the nature or scope of the process limitations.<sup>132</sup> This position is similar to prior-stated USPTO examining positions.<sup>133</sup>

An example of an article of manufacture claim is taken from claim 38 of issued U.S. Patent No. 5,461,488 (1994):

Article of Manufacture Claim Example 1. An electronic facsimile communicator stored via storage media, the storage media comprising:

a first plurality of binary values for receiving a facsimile transmission and storing the facsimile transmission in a first data format;

a second plurality of binary values for transforming the first data format to a second data format;

a third plurality of binary values for scanning the second data format to determine a recipient of the facsimile transmission out of a plurality of potential recipients in a local area network, if no direct recipient is determined, a default recipient or a recipient identified by the third plurality of binary values as being the most likely intended recipient of the facsimile transmission is set to be the recipient;

a fourth plurality of binary values for electronically routing the facsimile transmission to a recipient chosen from the plurality of potential recipients by the scanning performed by the third plurality of binary values; and

a fifth plurality of binary values for storing log data to keep a history of past electronic routings of facsimile data.<sup>134</sup>

The claim above is not limited to any particular medium. The recited "storage media" can be magnetic tape, optical disc, compact disc (CD), hard disk, floppy disk, ferroelectric memory, electrically erasable programmable read only memory (EEPROM), flash memory, EPROM, read only memory (ROM), static random access memory (SRAM), dynamic random access memory (DRAM), ferromagnetic memory, optical storage, charge coupled devices, smart cards, and any sort of storage media that is properly disclosed in the specification. The USPTO may state that the claim thus reads on any article of manufacture and accordingly will handle the claim as stated in the Examination Guidelines.<sup>135</sup> The use of other physical limitations and field of use limits, such as "local area network," or "FAX transmission," will result in the claim passing muster under USPTO procedures, since the claim does not wholly preempt an algorithm in the abstract but is instead a claim limited to a specific technology.<sup>136</sup>

If an Examiner rejects the article of manufacture claim with 35 U.S.C. § 101, the software patent practitioner may use two possible avenues to rebut the rejection. First, within the originally filed patent specification, one can provide detailed figures<sup>137</sup> and textual description.<sup>138</sup> These detailed sections will lend enabling support to claims that recite specific substrate configurations for the major software storage media (CDs, magnetic disk, magnetic tape, and transistor-configured IC memory) commonly used in the computer art.<sup>139</sup> This way, any broad claims rejected under 35 U.S.C. § 101 for reading too generically on any computer-readable manufacture will come into compliance if amended to include "specific manufacture" claims for each widely-used computer-readable medium.<sup>140</sup> However, this means that the practitioner must: (1) draft at least four total claims to cover the four most commonly used storage media,<sup>141</sup> (2) pay additional USPTO fees,<sup>142</sup> (3) lose (arguably forever) some exotic media coverage from the claims (e.g., optical storage, ferroelectric storage), and (4) make the claims narrower. Furthermore, under changes made to implement the General Agreement on Tariffs and Trade (GATT), the patent term could last nearly twenty years.<sup>143</sup> Unfortunately, specific/narrow claims drafted to specific storage media, such as a CD or a floppy disk, may be obsolete in fewer than ten years, effectively reducing the valuable life of the software patent.

The second way to approach a 35 U.S.C. § 101 rejection of an article of manufacture claim is to attack the validity of the USPTO's decision to ignore the article of manufacture limitations and to investigate only the process limitations for a determination of patentability.<sup>144</sup> It is probably desirable to attack the process described in the Examination Guidelines<sup>145</sup> as being improper in light of previous federal case law before succumbing and amending the article of manufacture software claims to include specific media configurations disclosed in the specification. The practitioner should amend the claims to include specific computer readable media only if she cannot convince the Examiner that either the USPTO Examination Guidelines are improper or that the claims already contain adequate structure to render them patentable subject matter in view of 35 U.S.C. § 101.

The example below illustrates how a practitioner might amend the Article of Manufacture Claim Example 1 above to describe a specific storage embodiment, such as a compact disc (CD):

Article of Manufacture Claim Example 2. An electronic facsimile communicator stored [via storage media] on a compact disc (CD) having a top programmed surface, the [storage media] compact disc comprising:

a first plurality of [binary values] trenches formed within a first portion of the top programmed surface which are spatially configured to provide a first set of binary values for receiving a facsimile transmission and storing the facsimile transmission in a first data format;

a second plurality of [binary values] trenches formed within a second portion of the top programmed surface which are spatially configured to provide a second set of binary values for transforming the first data format to a second data format;

a third plurality of [binary values] trenches formed within a third portion of the top programmed surface which are spatially configured to provide a third set of binary values for scanning the second data format to determine a recipient of the facsimile transmission out of a plurality of potential recipients in a local area network, if no direct recipient is determined, a default recipient or a recipient identified by the third plurality of binary values as being the most likely intended recipient of the facsimile transmission is set to be the recipient;

a fourth plurality of [binary values] trenches formed within a fourth portion of the top programmed surface which are spatially configured to provide a fourth set of binary values for electronically routing the facsimile transmission to a recipient chosen from the plurality of potential recipients by the scanning performed by the third plurality of binary values; and

a fifth plurality of [binary values] trenches formed within a fifth portion of the top programmed surface which are spatially configured to provide a fifth set of binary values for storing log data to keep a history of past electronic routings of facsimile data.

The strategy outlined above should adequately protect software article of manufacture claims from rejection under 35 U.S.C. § 101. Because persisting USPTO practice for examining software inventions and the tests from the CAFC are largely form over function, a good software practitioner will disclose a substantial degree of hardware and physical substrate configuration in the software specification. The practitioner will not initially claim these; rather, she will first attempt to obtain broad software process and article of manufacture claims. If this technique becomes too expensive or takes too much time, and thus loses too many years of protection for the client under the new 20-years-from-filing patent term, the practitioner should then gradually amend claims to contain more hardware and physical limitations. The practitioner can continue this process until the claims are either allowed with reasonable scope or the USPTO/CAFC begins setting precedent that would invalidate every hardware/computer patent ever issued in U.S. history. It is unlikely that the USPTO or the CAFC would ever go that far.

### **C. USPTO Rejection Techniques Under 35 U.S.C. §§ 112 and 103**

To limit the patentability of article of manufacture claims, the USPTO can, and probably will, use a three-front attack on these claims. The USPTO may use: (1) 35 U.S.C. § 101; (2) 35 U.S.C. § 112; or (3) 35 U.S.C. § 103, or any combination of the above, to attack the software article of manufacture claims. The rejection techniques under 35 U.S.C. § 101 have been discussed above. The following sections discuss the USPTO attacks using 35 U.S.C. §§ 112 and 103.

#### *1. The 35 U.S.C. § 112 Rejection of ARTICLE OF MANUFACTURE Claims*

In addition to challenges under 35 U.S.C. § 101, the USPTO has been, and may still be, ready to reject any *Beauregard* article of

manufacture claims as not enabled under 35 U.S.C. § 112, first paragraph. The USPTO's position would be "you claim an article of manufacture, but you do not teach how to make a disk, SRAM, tape or CD in the specification or the drawings, and therefore your disclosure is not enabling."<sup>146</sup> Such a rejection may lead to all sorts of new matter problems, specification amendments, and so forth. Fortunately, the practitioner can avoid this type of rejection with some initial preventative specification drafting. For instance, the practitioner can find technical papers (e.g., IEEE papers), patents, and/or books that teach the technology the practitioner wants to enable peripherally in the specification and incorporate these documents into the specification by reference.<sup>147</sup> It is therefore critical to disclose significant article of manufacture structure limitations in the specification and/or drawings not only to overcome 35 U.S.C. § 101 rejections as discussed above but also to overcome or prevent the assertion of 35 U.S.C. § 112 nonenablement rejections.

## *2. The 35 U.S.C. § 103 Rejection of aRTICLE OF MANUFACTURE Claims*

A third manner in which the USPTO will attack article of manufacture claims is under the obviousness test in 35 U.S.C. § 103. The Examiner may claim that one of ordinary skill in the art would know to place software onto storage media and therefore the article of manufacture claims are obvious. This position not only violates common sense, but it also violates decades of Federal Circuit opinions construing and interpreting the scope of 35 U.S.C. § 103 and the meaning of obviousness.<sup>148</sup> First, common sense dictates, for instance, that Intel's new microprocessor (Pentium(tm) Pro) may either be patentable or contain patentable subject matter, even though engineers have known for decades to place digital circuitry onto semiconductor wafers or printed circuit boards. The fact that a known medium or substrate is used to instantiate a novel procedure, function, or configuration should hardly render a claim obvious. New microprocessors, even though manufactured on known substrates, will contain patentable subject matter because the functions and/or configuration of the new microprocessors are novel. A rejection under 35 U.S.C. § 103 cannot be justified solely by the fact that the invention is manifested in tangible media. If it is true that a new processor and new program code would not be patentable because they were made on a known starting material, then with equivalent logic, the USPTO could state that it is known to use elements of the chemical periodic table to make tangible objects, and therefore every potential tangible object made by man is obvious.

Decades of federal case law clearly indicate that 35 U.S.C. § 103 should not be interpreted in this manner. For example, the following arguments can be made under federal case law: (1) the novel combination of old elements is patentable when new and useful functionality is obtained over the prior art;<sup>149</sup> (2) all claim limitations must be considered, especially when missing from the prior art;<sup>150</sup> and (3) the prior art does not teach the problem or a source of the problem solved or addressed by the software.<sup>151</sup> Further, it is hard to believe that: (1) software for determining a cure for every cancer for individual patients once blood data is provided to the software program; (2) Internet software which transmits newly released theater movies directly to your home for viewing; (3) intricate control software on disk which enables the manufacture of room temperature superconductor material that is not brittle; or (4) software which enables life support for a manned space flight to Mars, could all be deemed obvious on the ground that it is known to place software onto storage media.

A word of caution is in order. Some 35 U.S.C. § 103 rejections are warranted. Taking an algorithm well known in hardware and performing it in software, and vice versa, is obvious in most circumstances. Taking the process of balancing a checkbook as done by hand for many years and programming this same process into a computer is obvious, absent some revolutionary new additions required by the software embodiment. However, a new function or new method implemented in software should be just as patentable in view of 35 U.S.C. § 103 as a new function or new method implemented in hardware, even if known pieces or known steps are utilized.

Accordingly, with proper study of the case law and proper specification drafting before filing the patent application, a practitioner can minimize costs effectively, minimize time lost in prosecution, avoid abandonment, and avoid much aggravation when placing software article of manufacture claims in a software patent application. A software patent practitioner should attack a 35 U.S.C. § 103 rejection on two grounds. First, the rejection should be attacked technically, i.e., the prior art combination does not arrive at what the claim currently states. Second, the rejection should be attacked legally, i.e., the case law states that this is an improper 35 U.S.C. § 103 rejection. These two techniques should be used before 35 U.S.C. § 103 secondary considerations, claim amendments, and file wrapper estoppel comments are used to overcome a 35 U.S.C. § 103 rejection.<sup>152</sup>

### **D. Summary of *Beauregard* Article of Manufacture Claims**

In summary, the software article of manufacture claim will allow the patent holder to: (1) improve the value of his or her patent when licensing; (2) sue and obtain judgments against infringers over whom the patent holder would otherwise not prevail; (3) decrease the chances of lengthy litigation due to an increased probability of success on direct, induced, and contributory infringement case law; and (4) provide alternative portfolio strategies for the client. The prosecution obstacles created by article of manufacture claims under 35 U.S.C. §§ 101, 103, and 112 may well be worth the trouble in order to obtain issued U.S. software patents. Article of manufacture claims are less troublesome if the practitioner provides adequate specification disclosure initially. This procedure also inherently provides

some fall-back hardware limitations for the claims.

## VI. Method of Manufacture Claims

In many circumstances, the innovative software is not very expensive to purchase and/or produce, or the software has little monetary value when claimed alone. However, one or more infringing corporations may use even inexpensive software to manufacture or control the manufacture of millions or billions of dollars in product per year. For these software innovations, the true royalty base lies not in the software itself but in the method of manufacture that the software performs. In some cases, the software enables the making of a product where prior techniques could not. In other cases, the new and novel software may improve product yield or reduce scrap, reduce manufacturing cost, improve time to market, provide greater throughput so that more articles are manufactured per hour than previously possible, provide for greater manufacturing accuracy and tighter tolerances, improve profits, and so on. In these circumstances, if the software practitioner were to draft only: (1) structure claims limited to the computer and the software in combination; (2) process claims limited to the software alone; (3) *Alappat* means-plus-function claims limited to a "new machine" or a computer; and (4) *Beauregard* article of manufacture claims, the client's patent might not produce maximum royalty potential. A patent holder can arguably improve the software patent royalty base by including applicable method of manufacture claims in a software patent application.<sup>153</sup>

### A. Comparison Between a Conventional Process and a Method of Manufacture Claim

For example, assume software for monitoring and controlling plasma intensity in a semiconductor deposition chamber to improve deposition film thickness, yield, film integrity, and composition may be purchased from a vendor for \$5,000. The claim for this invention may read:

Conventional Process Claim Example. A computer-implemented process for adjusting plasma luminosity comprising the steps of:

reading an input luminosity which defines an initial luminosity value, the input luminosity being read from a memory using a CPU;

monitoring the input luminosity value over time and force adjusting the input luminosity over time via control signals output from the CPU such that the input luminosity is maintained within a standard deviation of .001% from the initial luminosity value;

setting error flags in the memory to warn of gross variations in the standard deviation of the input luminosity; and

setting binary values for shutting off power to power supplies which supply power to create the input luminosity if a serious process fault is detected by the CPU.

This process claim may be interpreted as being limited to the software alone and/or the hardware system on which it is executed. Nothing in the claim recites the products made, and the royalty base may therefore be limited to the software and/or inexpensive computer hardware. Furthermore, all other claims in the patent are likely to include structure claims limited to the computer hardware, *Alappat* claims limited to the "new machine" or computer, and article of manufacture claims under *Beauregard* which will have a one-time sales price of \$5,000 per software purchase (and may very well be a single purchase or at least less than several hundred purchases per infringing corporation). Due to the choice of strategy used during prosecution, a potential infringer would argue that the reasonable royalty for these claims is dependent upon the \$5,000 purchase price for the software, and is not based upon any method of manufacture using the software. A royalty equal to a small percentage of \$5,000 is hardly a reason to file suit in federal district court.

However, the patent holder will have a better royalty position if the practitioner views the invention in a slightly different light. If the practitioner views the new software as an aspect of a method for depositing a film on a semiconductor wafer in order to mass manufacture semiconductor wafers, the practitioner could draft additional claims that would arguably increase the client's royalty base by orders of magnitude:

Method of Manufacture Claim Example. A method for manufacturing a semiconductor wafer comprising the steps of:

placing the semiconductor wafer into a deposition processing chamber;

creating a plasma environment in the chamber to begin deposition of a layer of material onto the semiconductor wafer;

reading an input luminosity provided from the chamber which defines an initial luminosity value, the input luminosity being stored in memory and read by a CPU;

monitoring the input luminosity value over time and force adjusting the input luminosity over time such that the input luminosity is maintained within a standard deviation of .001% from the initial luminosity value;

setting error flags in memory to warn of gross variations in the standard deviation of the input luminosity;

setting binary values for shutting off power to power supplies which supply power to create the input luminosity if a serious error is detected by the CPU;

terminating the plasma once the layer of material is completely deposited onto the substrate; and

removing the wafer from the chamber.

With this type of claim, the patent holder now has a system he or she can license for higher royalties, rather than just a software package to sell at a fixed rate.

With a proper mindset, a software patent attorney can characterize any software that performs a manufacturing operation as a method of manufacture. For example, a compiler for C code is a method for mass producing compiled computer code onto computer readable media; a method for positioning bolts into a widget using a robot is a method for manufacturing a widget containing the bolts; software for controlling the temperature in a chemical reaction chamber is a method for making the chemical; a computer aided design (CAD) tool for designing an integrated circuit (IC) is a method for designing and mass manufacturing integrated circuits; a computer aided manufacturing (CAM) tool for jet engine design is a method for making jet rockets; software for setting up splines in memory and sending x, y, and z coordinates to an I/O port for motor control is a method for cutting a contour into an object to produce paneling for home building, and so on. The use of method of manufacture claims has many advantages, which are discussed below.

## **B. Advantages of Software Method of Manufacture Claims**

Three advantages result from using method of manufacture claims in a software patent. First, since method of manufacture claims contain physical limitations, specific application limitations, more-than-post solution-activity limitations, and the like, the CAFC case law cited above and USPTO policy outlined in the Examination Guidelines will result in these claims being easily allowed over 35 U.S.C. § 101. A second advantage is that the royalty base may improve, given current federal case law on infringement damages. Third, the method of manufacture claim may enable additional actions against foreign software-using companies which may not be possible when claims are limited only to the software routines. The latter two advantages will be analyzed in further detail below.

### *1. Damage AdvantageS of Using Software Method of Manufacture ClaimS*

The software patent inventor or assignee may obtain larger damage awards when claiming the method of manufacture than when relying exclusively on claims directed purely to the software in isolation. A patent holder has three primary ways in which to collect damages for infringement, each of which must be investigated in light of software method of manufacture claims.<sup>154</sup> One method for obtaining damages is to develop an established royalty base through licensing or other similar law suits.<sup>155</sup> For instance, if the software patent owner were to obtain more than one monetarily high royalty payment or acquiescence from more than one licensee for the software claim reciting the integrated method of manufacture, then subsequent courts would be likely to enforce this royalty as reasonable. In the above Conventional Process Claim Example, an infringer would be unlikely to be willing to pay much more than \$5,000 for infringing the claim, and the infringer would go to court if the patent holder demanded millions in royalty payments over the years.<sup>156</sup> However, an infringer would be more likely to pay a percentage of manufactured product profits if confronted with a claim such as the Method of Manufacture Claim Example given above.

Where an established royalty is not available for review by the court, the court may use the concept of reasonable royalty to calculate infringement damages.<sup>157</sup> Arguably, given the current reasonable royalty case law, the Method of Manufacture Claim Example above will provide for greater reasonable royalties than the Conventional Process Claim Example. Since the method of manufacture claim will cost roughly \$100 to add to a U.S. patent<sup>158</sup> plus the cost of drafting, the slight additional up-front prosecution cost is well worth even a slight possibility of massively increasing the royalty base for the client. The practitioner may even be able to add method of manufacture limitations to the software claim strategy via dependent claims, which could further reduce patent filing costs and

restriction requirements for the client, while still providing an extremely valuable royalty base position in licensing and litigation.<sup>159</sup>

The third method for awarding damages is to award the plaintiff lost profits for infringement by its competitors. Lost profits are a reasonable measure of damages in patent infringement suits.<sup>160</sup> A plaintiff is entitled to lost profit damages if she can prove: (1) demand for the product in suit; (2) absence of acceptable noninfringing substitutes to the patent in the suit; (3) the plaintiff's ability in terms of capacity to meet the demand brought on by the infringing goods; and (4) the amount of profit that the plaintiff would have made on the product in suit, absent the infringement.<sup>161</sup> For the above Conventional Claim Example, the lost profits on the sale of the software itself may be much less than the \$5,000, the purchase price for the software. However, for the Method of Manufacture Claim Example, the revenue lost to infringing competitors engaged in the mass manufacture of semiconductor wafers over many years using the infringing software may easily climb into billions of dollars per infringer.<sup>162</sup> Under the third *Panduit* factor listed above, a software innovator will only be entitled to these large damage awards if it would have been able to exploit the demand for the product. Thus, a corporation that manufactures only software will never be able to get lost profits on manufactured products.<sup>163</sup>

## 2. Importation and I.T.C. Advantages when Using Software Method of Manufacture Claims

To illustrate an additional benefit to method of manufacture claims in U.S. software patents, assume that a client has obtained a better-than-average software patent, containing structure claims, process claims, *Alappat* means-plus-function claims, and *Beauregard* article of manufacture software claims. The client's competitor develops similar software in Japan. The Japanese competitor uses this infringing software to manufacture widgets, which it then imports into the U.S., thereby swallowing your client's market share in the U.S. The client would like to file an International Trade Commission (ITC) action and obtain an injunction to stop the importation of the goods manufactured by the Japanese competitor.<sup>164</sup> These injunctions are only available on infringing products imported to or exported from the U.S. However, the actual infringing devices embodying the software structure claims and the means-plus-function claims are made in Japan and are never imported to or exported from the U.S. in any manner. With respect to *Beauregard* article of manufacture claims, any disks, tapes, or memory integrated circuits containing the software are also not imported to or exported from the U.S. Therefore, these three types of software claims are probably useless to your client in the ITC and are useless to stop importation of the Japanese widgets into the U.S.

Further, with respect to the Conventional Process Claim Example given above, the competitor may argue that this claim is a process claim that recites memory and a plasma as the products it produces. The plasma and the memory elements that the software code modifies are not imported from Japan into the U.S. Therefore, the pure software process claims may be difficult to assert in the ITC for injunctive purposes. Further, process claims like the Conventional Claim Example are more likely to be found invalid under current U.S. case law because they are more likely to contain abstract or intangible limitations, as opposed to the more tangible and "physical" method of manufacture claims.<sup>165</sup> Software method of manufacture claims, however, do provide a way to get an ITC injunction. Products imported into the U.S. that are made by a process patented in the U.S. are deemed to be infringing under 35 U.S.C. § 271(g). The patent holder is thus entitled to an injunction to prevent importation of these products. Therefore, the client's ability to get quick relief from foreign use of infringing manufacturing software may hinge entirely on the software patent draftsman's ability to secure software-implemented method of manufacture claims for the client.

In sum, method of manufacture claims provide increased chances of validity under 35 U.S.C. § 101, a greater chance for improved monetary compensation, and improved positions when attempting to obtain injunctive relief against importation by foreign corporation.

## VII. Data STRUCTURE CLAIMS

Obtaining claim coverage for data structures in addition to software algorithms may be very advantageous for a client. A data structure is "a physical or logical relationship among data elements, designed to support specific data manipulation functions."<sup>166</sup> Under federal case law, obtaining patent protection for a machine and hardware-limited software algorithm is easier than obtaining patent protection for a data structure.<sup>167</sup> This section will explain why data structure claims should be statutory subject matter.

### A. Using a Software Patent to Protect Software Data Structures

In the recent case *In re Lowry*,<sup>168</sup> the CAFC stated that data structures that comprise sequences of bits stored in the memory of a computer as electrical (or magnetic) signals that represent information are acceptable under 35 U.S.C. § 101. In *Lowry*, the court found the following claim to be statutory under 35 U.S.C. § 101:

A memory for storing data for access by an application program being executed on a data processing system, comprising:

a data structure stored in said memory, said data structure including information resident in a database used by said application program and including: a plurality of attribute data objects stored in said memory, each of said attribute data objects containing different information from said database;

a single holder attribute data object for each of said attribute data objects, each of said holder attribute data objects being one of said plurality of attribute data objects, a being-held relationship existing between each attribute data object and its holder attribute data object, and each of said attribute data objects having a being-held relationship with only a single other attribute data object, thereby establishing a hierarchy of said plurality of attribute data objects;

a referent attribute data object for at least one of said attribute data objects, said referent attribute data object being nonhierarchically related to a holder attribute data object for the same at least one of said attribute data objects and also being one of said plurality of attribute data objects, attribute data objects for which there exist only holder attribute data objects being called element data objects, and attribute data objects for which there also exist referent attribute data objects being called relation data objects; and

an apex data object stored in said memory and having no being-held relationship with any of said attribute data objects, however, at least one of said attribute data objects having a being-held relationship with said apex data object.<sup>169</sup>

Not only did the court find that the above claim was patentable subject matter because it recited memory and physical data structure elements, the court further held that the inclusion of a database, a central processing unit (CPU) means for processing the application program, and a memory means for holding the claimed data structure would clearly render any data structure claim patentable as an electrical system or article of manufacture under 35 U.S.C. § 101.<sup>170</sup> Further, *Lowry* held that the USPTO and the courts must consider printed matter (e.g., the database) contained in a data structure claim when they evaluate data structure claims under 35 U.S.C. §§ 102 and 103.<sup>171</sup> In summary, *Lowry* held that if a machine is programmed in a certain new and nonobvious way via a data structure, then the machine is physically different from the machine without that structure because memory elements in the machine are arranged differently.<sup>172</sup> If the claim is in any way more than a mere abstraction, such as a data structure stored in specific electrical or magnetic structural elements in a memory, the claim is patentable under 35 U.S.C. § 101.<sup>173</sup>

Further, the USPTO Examination Guidelines also address how to make data structures statutory. Data structures are "descriptive material" according to the USPTO, much like music, literary works and pictures stored on a CD or a disk are descriptive.<sup>174</sup> "Descriptive material" is nonstatutory per se when the material is non-functional.<sup>175</sup> Examples of non-functional descriptive material include music, literary works and a compilation or mere arrangement of data.<sup>176</sup> However, descriptive material that is "functional" may be statutory in some cases.<sup>177</sup> The USPTO states that functional descriptive material when recorded on computer readable medium becomes structurally and functionally interrelated with the computer readable medium, and is thus statutory.<sup>178</sup> Because the USPTO defines data structures to be functional,<sup>179</sup> when these data structures are claimed to define structural and functional interrelationships with other claimed aspects of the invention, thereby permitting the data structure's functionality to be realized, such claims should be statutory.<sup>180</sup>

An example of a successful data structure claim is taken from claim 24 in U.S. Patent No. 5,414,701 (1994):

Data Structure Claim Example. A data structure for performing address compression in an asynchronous transfer mode (ATM) system, the data structure comprising:

a link array, the link array containing a plurality of entries wherein one of the entries is accessed for each incoming ATM data cell, each entry in the plurality of entries comprising a virtual path pointer;

a plurality of virtual path tables wherein each virtual path pointer in the link array is used to address a unique one virtual path table in the plurality of virtual path tables, each virtual path table in the plurality of virtual path tables comprising at least one virtual path entry, the at least one virtual path entry comprising a virtual channel pointer; and

a plurality of virtual channel tables wherein each virtual channel pointer in the plurality of virtual path tables is used to address a unique one virtual channel table in the plurality of virtual channel tables, the unique one virtual channel table comprising a plurality of entries wherein one of the entries in the plurality of entries comprises an ingress connection identifier for routing ATM data in the asynchronous transfer mode (ATM) system.

## **B. Barriers to Data Structure Patentability**

Despite the holding in *Lowry* and statements in the USPTO Examination Guidelines that data structures can be patentable, this author has faced numerous rejections from the USPTO for data structure claims. The USPTO seems to have adopted a position that data structures do not interrelate with computer-readable media as does software code, and are therefore non-functional.<sup>181</sup> Thus, the USPTO seems to view data structures more like music, text, or pictures stored on a CD or a disk, which are non-functional and not patentable. For the reasons discussed below, data structures should be considered no different from software and just as functional.

First, it is improper for the USPTO to draw a connection between data structures and common forms of non-functional material. Pure music, binary pictures and readable text are passive information that may be displayed and stored by a computer; however, this information does not instruct a computer to perform an operation (i.e., is not executable in most cases) and does not alter computations in a computer in a functional manner (i.e., is not read as functional input data in most cases). Consider the following: suppose someone digitizes a Mozart song onto a magnetic disk and places the magnetic disk in a computer. The user presses "play" and the computer executes functional software to play Mozart's music from the disk. The computer will perform the same executable function if a Van Halen song was placed onto the magnetic disk instead of Mozart. There arguably is no difference. The functioning of the computer is arguably the same since the song is passive information/data and the computer does not care about or substantially change functionality depending upon what digital music data is stored on the disk. In addition, a GIF picture of the Rocky Mountains will be displayed via executable computer code in a functional manner that is identical to the functional manner used by the executable computer code to display a GIF of Captain Kirk on the Starship Enterprise.

On the other hand, when software code and/or software data structures are altered on a computer by changing disks or the like, the machine will functionally perform a completely different operation since software and data structures are active features and not passive data within the computer system. One must be careful not to confuse a data structure with data; one is patentable and the other usually is not. Also consider *Lowry*, which states that "[t]he printed matter cases have no factual relevance where 'the invention as defined by the claims requires that the information be processed not by the mind but by a machine, the computer.'"<sup>182</sup> Music, English text, and visual pictures are information that are presented in a form for easy and efficient processing by the human mind. I can readily comprehend a picture on a 17" screen, case law text displayed by Westlaw(tm), or music played from a speaker. However, I cannot easily understand and traverse binary code, B trees, linked lists, hash tables, hierarchical arrays, multi-level feedback queues, and FIFOs, since these items are data structures designed for easy traversal and processing by a computer and not a human mind. Both software and data structures are functional and both should be considered statutory subject matter for this reason.

Second, data structures should actually be more patentable than software because software is arguably more like printed matter.<sup>183</sup> Unless a programmer has written self-modifying executable software code, the software code is fixed in memory and never changes. The software instructs the computer to do something and to that extent is "functional." However, the software itself can be viewed like a statue, which once created is fixed for all time. In comparison, if software programs are unchanging statues, then data structures are moving and changing machines. They can be totally destroyed, moved to different locations, changed over time, be expanded in size and/or contents, reduced in size and/or contents, changed in value, created from smaller pieces, and so on. Data structures change radically every millisecond in a functional manner. Thus, data structures should be even more patentable than software because they are typically more functional and more dynamic in form than the executable code that the courts have so widely embraced as patentable subject matter.

Third, separating what is executable software code from what is a data structure in computer storage is very difficult. When executing a compiler to compile code, the compiler A compiles high level computer code B into executable object code C. To the compiler, the code B and code C are data structures that require manipulation as do any other textual or numerical data structure. To another program or hardware, the same code portions B and C are executable software. Verilog(tm) code of a CAD design tool, which represents an integrated circuit executable code, can be viewed as simulation-dependent code, a data structure, or a little of both. Wingz (tm) code written within a Wingz(tm) spreadsheet may be viewed either as a data structure within the Wingz(tm) application or as software executed by the Wingz(tm) application. The code used to play a computer game is functional software when it is executed by a processor, but is a data structure when it is broken into pieces and transmitted over the Internet. Sometimes executable software code is a data structure and other times a data structure is executable software code. To say that one is patentable subject matter while the other is not cannot be justified.

The USPTO has recently adopted a position whereby an item that is restricted in claims to "specific" machines or manufactures will be more likely to be statutory under 35 U.S.C. § 101 than an item which is not restricted to "specific" machines or manufactures.<sup>184</sup> This position should also render data structures more patentable than executable software code. For example, static and unchanging software executable code can be stored in ROM memory.<sup>185</sup> Most data structures, on the other hand, are dynamic and cannot be stored in ROM memory because user-initiated write operations to memory are almost always required in order to maintain a data structure. EPROM and EEPROM memory are likewise more feasible for software than data structures due to their long write/erase times, which render dynamic changes inefficient.<sup>186</sup> In addition, case law aside from *Lowry* supports the conclusion that both software and data

structures are patentable subject matter. If software, a series of functional 0s and 1s, can render memory, disks, and computers patentable subject matter under *Warmerdam*, *Alappat*, *Trovato*, and *Beauregard*, then a data structure, which is a series of functional 0s and 1s, should likewise be considered a new machine or new article of manufacture.<sup>187</sup> For these reasons, data structures are arguably more limited to specific storage media than software. They are more similar to new machines or articles of manufacture than software, and should therefore be more likely to overcome 35 U.S.C. § 101 rejections.

### C. The Advantage of Data Structure Claims

The advantage of the data structure claim is clear. A novel data structure typically can be made by more than one process (i.e. software algorithms). In addition, a novel structure can perform many different functions. These functions and many methods of formation may allow a potential infringer to avoid infringing a software patent that contains only structure claims, process claims, *Alappat* means-plus-function claims, and the like. However, a practitioner can draft a data structure claim so that the patent is infringed regardless of both the use of the data structure and the method used to form the data structure. The final structure that results from any of these uses and methods of formation is recited in the data structure claim. This realization is not new in patent claim drafting. For instance, if one invents a new device (say a new transistor) using a novel process, it is universally understood that many other processes or process steps, either now known or later developed, may be used by others to arrive at the same transistor. In this case, a process patent may be limited in infringement potential, especially if there are 300 different processes that are capable of achieving the same structure. However, a single data structure claim will cover all 300 different processes in one claim, since all the processes will result in the formation of the same structure in the end. Therefore, data structure claims for novel data structures are required, where applicable, to provide adequate patent protection for a software innovation.

### VIII. A Comprehensive Software Claim Drafting Scheme

Given the above sections, it is possible to list the types of claims a practitioner should be aware of when drafting a software patent application. In order to protect a software inventor's interest properly a patent attorney should:

(1) Draft at least one software structure/apparatus claim, which will:

- (a) provide a claim that will maximize the likelihood of compliance with 35 U.S.C. § 101;
- (b) provide a claim that allows the attorney to determine how hard the particular art group or Examiner is going to press 35 U.S.C. §§ 101, 103, and 112 rejections; and
- (c) provide another common claim format that courts and the USPTO are comfortable having prosecuted.

(2) Draft at least one software process claim, which will:

- (a) provide a common ground for the inventor and the attorney to begin the claim drafting process;
- (b) provide a claim that many courts are comfortable with, because these claims are used in a large percentage of all software patents; and
- (c) potentially provide notice, marking, and damages advantages under 35 U.S.C. §§ 286 and 287.

(3) Draft at least one *Alappat* means-plus-function claim, which will:

- (a) provide an alternative claim style to the structure claim;
- (b) provide a claim style that was recently provided as a safe harbor for software claims in *Alappat*; and
- (c) provide equivalents arguments that will allow the "means" to be read on both disclosed hardware and disclosed software-the court need interpret only one of the "means" in the claim as hardware to render the claim statutory, while it can interpret the rest of the "means" as intangible pure software due to the disclosure of both hardware and software in the specification.

(4) Draft at least one *Beauregard* article of manufacture claim, which will:

- (a) allow the client to avoid contributory infringement barriers and instead assert direct infringement; and
- (b) provide for more potential infringers.

(5) If applicable, draft at least one method of manufacture claim, which will:

- (a) provide a process claim the courts are more likely to allow than the process claim of item (2) above;
- (b) increase the client's royalty base in litigation and licensing; and
- (c) provide for more adequate injunctive remedies against foreign competitors.

(6) If applicable, draft at least one data structure claim, which will:

- (a) provide for infringement of the formation of a data structure independent of hardware limitations or software process constraints; and
- (b) provide collateral protection for the client's software algorithms.

The practitioner should use at least two of the above six software claim types for claims within the patent application in order to vary the scope of the software claims among narrow, medium, and broad scope. By drafting claims of at least two differing scopes of protection, the patent claims will be harder to invalidate as a whole, while infringement of the claims will be maximized.

#### **A. Cost Analysis of the Comprehensive Software Claim Drafting Scheme**

Unfortunately, due to the way in which the software patent law has developed, it is very expensive to file and prosecute software patents properly. The default filing fee for a patent for a large entity at the time of this writing is \$770, which includes up to three independent claims and twenty claims total.<sup>188</sup> However, a U.S. software patent is likely to contain seven or more independent claims and an average of at least forty total claims if the practitioner drafted the patent to protect the client's innovation fully, as discussed in this comment. Thus, a typical software patent may have double the minimum USPTO filing costs (roughly a \$1500 filing fee).

In addition to an increased filing fee, an attorney will usually have to bill more hours for a properly prepared software patent. Drafting a software patent application requires an attorney to have more extensive knowledge and breadth of understanding of the patent law, a greater understanding of both hardware and software technology, and a firm understanding of the client's software business in order to be comprehensive. The textual specification and drawings needed to support fully a comprehensive software claim drafting strategy are far more intense than those required to support other technologies adequately.

Further, the chance of receiving a two-way or three-way restriction requirement for a software patent is high. An N-way restriction requirement occurs where the USPTO claims that you filed one patent application that contains N inventions, where N is an integer greater than one.<sup>189</sup> You can battle the restriction requirement during prosecution but will usually lose. If you lose or acquiesce to the restriction requirement, you must file N patents for your client to obtain the same claim coverage you were hoping to get in one patent. This restriction will either force the client to compromise patent coverage in favor of reduced cost, or force additional costs onto the client for proper protection of the software innovation. With luck, the USPTO will not issue a restriction, as was the case for U.S. Patent No. 5,461,488 (1994). This patent contains structure claims, process claims, means-plus-function claims, and article of manufacture claims in one issued patent application. However, a restriction requirement is more likely than not to occur when attempting to claim fully a software invention using all of the claim styles discussed herein.

An attorney should address all of these cost issues up front with the software client to facilitate determination of maximum cost/benefit impact. The attorney may then attempt to minimize costs by drafting the various claim styles discussed herein in dependent form, or some styles may be eliminated from a client's software patent strategy with minimal negative impact to the client. This informed reduction of claim styles based on financial concerns and business perspectives may be required for certain clients. Once the client and attorney have discussed thoroughly all legal rationales of software patenting, business impact of software to the client, the financial

position of the client, and technological aspects of the client's innovations, the attorney can optimize the patent acquisition costs in an informed manner.

## **B. Disclosure Required to Support the Comprehensive Software Claim Drafting Scheme and to Provide Backup Positions During Prosecution**

The specification disclosure required for software inventions is significant. Structure claims require one or more of: (1) electrical system block diagrams illustrating a basic computer;<sup>190</sup> (2) block diagrams of a custom CPU designed to perform the software algorithm exclusively in hardware;<sup>191</sup> (3) circuit schematics; and (4) electrical signal timing diagrams in order to enable the invention/claims under 35 U.S.C. § 112, first paragraph. The process claims are usually supported by one or more of: (1) executable code or pseudocode sections as figures or as specification tables,<sup>192</sup> (2) one or more flowcharts;<sup>193</sup> and (3) an example of the algorithm as it affects objects over time (an illustration of the process at work).<sup>194</sup> Furthermore, any device claimed in an application must be illustrated in drawings.<sup>195</sup> *Alappat* means-plus-function claims are easily enabled by the above disclosure itemized in structure claim elements (1) through (4). For article of manufacture claims, it is prudent to illustrate the article of manufacture for at least one computer storage embodiment. However, textual discussion in the specification after disclosure of an electrical system seems to suffice. Method of manufacture claims may require additional flowcharts, and data structure claims should include a block diagram illustration of the data structure and its physical configuration on storage media, if possible.

The practitioner should adequately describe these figures in the specification as required under 35 U.S.C. § 112. Furthermore, it might be useful to cite technical papers, text books, and/or prior patents to overcome 35 U.S.C. § 112, first paragraph, enablement rejections. These rejections are sometimes directed to electrical system figures which contain conventional items (e.g., CPU, printer and display screen), conventional computer readable storage media (e.g., RAM, CDs and floppy disks), the manufacturing technology needed to form the electric system, and integrated circuits and storage media, when these items are not adequately described in the specification. If this information is not provided, the practitioner may be forced to prove them during prosecution and may be exposed to a new matter rejection.<sup>196</sup> Fighting new matter in the USPTO is not an easy battle; these problems are better diffused up front with adequately drafted specification disclosure containing incorporations by reference where necessary.

As can be seen, the combined skills needed in terms of hardware understanding and software understanding may be difficult to find even when taking into account the combined knowledge of both the software patent draftsman and the inventor.

## **IX. Conclusion**

Software patenting is a difficult art to master. Software patent law is complicated, delicate, form over function, and constantly in flux. The drafting of a software patent application requires extensive knowledge of the software patent law, extensive understanding of the client's business, understanding of the client's technical and financial goals and capabilities, and vast technical knowledge of both computer hardware (electrical/computer engineering) and software (computer science and programming). The prosecution of a software application is also difficult because rejections relating to software applications tend to be longer, more difficult to address, and more hostile in character than rejections related to other technological areas. The fact that the software industry is somewhat hostile to software patents and that many clients do not understand the need for software patents is cause for further frustration. In general, it is not easy to develop all of the skill sets needed to be truly effective at software patent prosecution, procurement, and portfolio management.

However, one easy and important step that any software patent practitioner can take in a short period of time to improve software drafting effectiveness is to understand the software claim weapons at his or her disposal. The practitioner should learn to wield these weapons with maximum advantage for a software client. Even without a broad understanding of the software business, the software technology, the computer technology, the software patent law, and the financial position of the software industry over time, the claim styles discussed in this comment (structure claims, process claims, means-plus-function claims, article of manufacture claims, method of manufacture claims, data structure claims) can provide the essential elements for a comprehensive software patent strategy designed to ensure improved patent portfolio benefit for the client.<sup>197</sup> However, these claim styles, when deftly wielded by a technically competent and experienced software patent practitioner, become even more powerful and useful for a client. In general, by understanding the claim styles herein and their respective strengths and weaknesses, the practitioner can spend a few extra dollars and a few additional hours when initially drafting and prosecuting a software patent application to either derive millions more dollars in value from an issued software patent, or avoid millions of dollars in unnecessary litigation expenses over the issued software patent.

† Practicing Patent Agent #37,475, Motorola, Inc., Semiconductor Product Sector; third year law student, University of Texas at Austin School of Law; B.S., Electrical and Computer Engineering, University of Wisconsin, Madison; B.S., Computer Science, University of Wisconsin, Madison.

Special thanks to Professor Mark Lemley, University of Texas at Austin School of Law; Bill Koch, Intellectual Property Attorney, Motorola, Inc., Semiconductor Product Sector, Phoenix, AZ; Bruce Hayden, Software Intellectual Property Attorney, Motorola, Inc., Semiconductor Product Sector, Austin, TX; and Dan Perez, Partner, Warren & Perez, Dallas, TX, for their comments and critiques regarding the contents of this paper.

The opinions expressed herein are my views and not necessarily the views of Motorola, Inc.

1. See discussion *infra* parts III, V, VI, VII.
2. This figure contains software patent issue data for two art classifications-compilers and file techniques-collected from the USPTO and using Internet search engines available at <http://sunsite.unc.edu/patents/intropat.html> and <http://www.spi.org/> on the World Wide Web.
3. See, e.g., *Stac v. Microsoft*, 38 F.3d 1222 (Fed. Cir. 1994).
4. See generally PAUL CARROLL, *BIG BLUE(S): THE UNMAKING OF IBM* (1993). The birth of Microsoft, for example, was largely due to big companies like IBM "farming-out" software projects to smaller companies in the late 1970s and early 1980s.
5. See generally BILL GATES ET AL., *THE ROAD AHEAD* (1995); see also Stephen Kreider Yoder, *Microsoft Again Signals Growth Is Leveling Off*, *THE WALL STREET JOURNAL*, July 24, 1992, at B4.
6. See discussion *infra* part II.A. (discussing the case law history of software patents).
7. 35 U.S.C. § 101 (1994) (statutory subject matter).
8. See discussion *infra* parts II.B., II.C., III.B.
9. Data derived from more than one hundred patent applications drafted by the author.
10. 35 U.S.C. § 101 (1994).
11. 35 U.S.C. § 103 (1994).
12. 35 U.S.C. § 112, para. 1 (1994).
13. Typical software patent applications must disclose the computer hardware or electrical computer system which incorporates the software in a manner similar to a typical electrical system patent. In addition, the software application must enable and provide the best mode of the actual software code and algorithms via flowcharts, psuedocode, or the like. This results in a lengthy patent application containing hardware disclosure and software disclosure, i.e., a double disclosure.
14. See, e.g., U.S. Patent Nos. 5,386,375 (1995) (51 claims), 5,341,320 (1994) (26 claims) 5,517,506 (1996) (26 claims).
15. See *supra* figure accompanying note 2; see also discussion *infra* parts III.A., V.A., VII.A. (discussing software claim types that may be allowable before the USPTO).
16. See discussion *infra* parts V, VI, VII (discussing article of manufacture claims, method of manufacture claims and data structure claims).

17. See generally GREGORY A. STOBBS, SOFTWARE PATENTS (1995) (discussing the evolution of software and the software industry in the 1960s and 1970s).
18. *Gottschalk v. Benson*, 409 U.S. 63, 71-73 (1972).
19. *Id.* at 73.
20. *Id.* at 64.
21. *Id.* at 65, 71-72; see also *id.* at 65 (suggesting that patents on mathematical abstract ideas could stop one from thinking or writing math); *id.* at 67 (stating that math, which can be performed without a computer, must be made available to the public as a basic tool of scientific work). One unfortunate result from this decision is that the USPTO and the federal courts continue to blanket reject any software claims which are math-intensive under 35 U.S.C. § 101. While some programs are more math-intensive than others (e.g., Mathematica(tm) or the modified Booth binary multiplication of two operands are arguably more math-intensive than Windows95(tm) or a digital storybook from Humongous Entertainment), math always exists in any software program if you look hard enough. This is why all microprocessors manufactured for computers contain one or more of an adder unit, a subtractor unit, a multiplier unit, a divider, and like integer and/or floating point math units. Every piece of software must have these math units in order to do math. This author cannot think of a single piece of significant software which does not use math to some extent, and 35 U.S.C. § 101 determinations should not be predicated upon "the degree of math" or function performed by the software. This is especially true where hardware limitations are recited in the claim. Hardware is hardware and renders a claim statutory whether the software limitations print graphics, balance a checkbook, find the square root of a number, or compile a program. A dichotomy between software with math and software without math is an impossible and a nonsensical slippery slope.
22. *Id.* (quoting *MacKay Co. v. Radio Corp.*, 306 U.S. 86, 94 (1939)).
23. *Id.* (quoting *Funk Bros. Seed Co. v. Kalo Inoculant Co.*, 333 U.S. 127, 130 (1948)).
24. See *id.* (citing *Funk Bros.*, 333 U.S. at 130 ("he who discovers a hitherto unknown phenomenon of nature has no claim to a monopoly of it which the law recognizes"))).
25. For a similar line of reasoning rooted in history, see *Eibel Process Co. v. Minnesota & Ontario Paper Co.*, 261 U.S. 45 (1923) (mere obvious and simple application of known natural forces is not patentable whereas a complex machine using natural forces is patentable).
26. *Benson*, 409 U.S. at 67.
27. *Id.* at 71.
28. See, e.g., U.S. Patent Nos. 4,370,983 (1981) and 3,930,146 (1975) (exemplifying software patents in the wake of *Benson* that employ hardware descriptions and hardware claim limitations).
29. Reference to *Benson* in USPTO file wrappers for software-related inventions are commonplace, and *Benson* has not been specifically overruled to date. Many modern cases discussed herein cite to *Benson* when considering software patent policy.
30. 437 U.S. 584 (1978).
31. *Id.* at 585.
32. *Id.* at 590. "Post-solution activity" is any physical step following mathematical algorithm steps in a claim.
33. *Id.* at 589 n.9.
34. *Id.* at 590.

35. *Id.* at 591.

36. *Id.* at 592; *see also id.* at 594 for similar reasoning.

37. *Id.* at 593.

38. *Id.* at 588.

39. *Id.* at 594.

40. This author agrees with Stewart's dissent stating that §§ 101, 102, 103 and others should all be separate inquiries under U.S. patent law. *Id.* at 600 (Stewart, J., dissenting). Generally, there is a difference between patentable subject matter and a patentable invention. Patentable subject matter is an invention which passes muster under 35 U.S.C. § 101. Even an innovation which is patentable subject matter under 35 U.S.C. § 101 may not be eligible for patent protection under 35 U.S.C. §§ 102, 103, and 112 (i.e., it may not be a patent-deserving invention, but still be patentable subject matter). For example, software which performs a binary search on textual objects or performs a select sort on numbers would be patentable subject matter even given the state of technology in 1996. However, these software programs would not be given 1996 patent protection due to the abundance of prior art under 35 U.S.C. §§ 102 and 103 in the binary search and select sort area. For example, Mathematica(tm) is software that can be currently disclosed and claimed as patentable subject matter under 35 U.S.C. § 101 given current federal case law by using proper hardware limitations. However, the functions performed by Mathematica(tm) are probably not patentable over the prior art (*see* 35 U.S.C. §§ 102 and 103) existing at the time Mathematica(tm) was created. After all, Mathematica(tm) simply performs math operations, which were previously known and previously performed by hand, on a computer.

41. *See, e.g.*, U.S. Patent No. 4,241,412 (1980) (illustrating the typical software structure claims which were considered statutory over *Flook*). To see the difference between apparatus or structure claims and process claims with structural limitations, *see infra* parts II.B., II.C.

42. *Flook*, 437 U.S. at 593.

43. *See, e.g.*, U.S. Patent Nos. 4,120,030 (1978), 4,086,470 (1978). Both of these "software" patent examples contain hardware diagrams, apparatus illustrations, memory device drawings, and circuitry. These "software" patents are indistinguishable from hardware patents.

44. It is amusing that many opponents of patent protection cite enormous software patent procurement costs as being one intolerable area of the whole idea of software patenting. By battling software patents so severely, these opponents create higher procurement costs by complicating the law. Worse, they complicate the law without hope of ever being able to achieve the goal of stopping software patents from being issued in some form or another. More complicated software patent law means more required billable hours per software patent application, which means only the rich or big corporations can afford an extensive software patent portfolio to the detriment of start-up software ventures. *See generally* The League for Programming Freedom, *Software Patents: Is This the Future of Programming?*, DR. DOBB'S J. OF SOFTWARE TOOLS, Nov. 1990, at 56 (opposing software patents).

45. 478 F.2d 1392, 1394 (C.C.P.A. 1973) (saying that "[g]iven that the method of solving a mathematical equation may not be the subject of patent protection, it follows that the addition of the old and necessary antecedent steps of establishing values for the variables in the equation cannot convert the unpatentable method to patentable subject matter").

46. 563 F.2d 1026, 1029-30 (C.C.P.A. 1977) (noting that where the claims merely recited limitations which provided values for variables used in mathematical formulae used in making the calculations, such antecedent steps would not suffice to render the claimed methods, considered as a whole, statutory subject matter for a patent).

47. 888 F.2d 835, 839-40 (Fed. Cir. 1989) (holding that the claimed method was nonstatutory subject matter where all but one of the steps of the claimed method were in essence mathematical algorithms and a remaining step merely provided data for the mathematical algorithm).

48. *See also In re Chatfield*, 545 F.2d 152, 156 n.5 (C.C.P.A. 1976) (holding that algorithm claims that are limited in application or by hardware are patentable since it would be unnecessarily detrimental to our patent system to deny inventors patent protection on the

sole ground that their contributions could be broadly termed an "algorithm"); *In re Noll*, 545 F.2d 141, 148 (C.C.P.A. 1976) (illustrating that structure-laden software claims will be viewed more favorably by the USPTO and the courts).

49. 450 U.S. 175 (1981).

50. *Id.* at 187.

51. *Id.* at 188.

52. *Id.* at 185.

53. *Id.* at 191-92.

54. *See id.* at 192. The Court stated, "[W]hen 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 § 101." *Id.* (emphasis added)

55. *In re Freeman*, 573 F.2d 1237 (C.C.P.A. 1978); *In re Walter*, 618 F.2d 758 (C.C.P.A. 1980); *In re Abele*, 684 F.2d 902 (C.C.P.A. 1982).

56. *Arrhythmia Res. Tech., Inc. v. Corazonix Corp.*, 958 F.2d 1053, 1057 (Fed. Cir. 1992) (discussing the modern application of the *Freeman-Walter-Abele* test).

57. *See Examination Guidelines for Computer-Related Inventions*, 51 PATENT, TRADEMARK & COPYRIGHT J. 422, 429 (1995) [hereinafter *Guidelines*]. These guidelines, though not binding as substantive law, were designed to assist the USPTO in the examination of applications drawn to computer-related inventions.

58. *Id.*

59. *See Flook*, 437 U.S. at 590.

60. *Guidelines*, *supra* note 57, at 430.

61. This section primarily discussed early case law which was hostile to software patents. However, it should be noted that the 1970s contained some cases which supported software patenting as long as structural limitations or physical steps were readily apparent in the claims. *See, e.g., In re Johnston*, 502 F.2d 765, 771 (C.C.P.A. 1974) (reversing USPTO's rejection of claims under 35 U.S.C. §§ 103 and 101 due to the significant structural content); *In re Freeman*, 573 F.2d 1237, 1246 (C.C.P.A. 1978) (reversing USPTO 35 U.S.C. § 101 rejection of method claims because the claims did not merely recite a procedure for solving a mathematical problem, but rather described new, useful and nonobvious processes for operating a computer display system); *In re Chatfield*, 545 F.2d 152 (reversing § 101 rejection since the claims pertained to method of operating a physical computer system and did not involve abstract and unapplied mathematics).

62. *See Guidelines*, *supra* note 57, at 429-30.

63. *Id.* at 427.

64. *Flook*, 437 U.S. at 590.

65. *See supra* text accompanying notes 55-56.

66. 33 F.3d 1526, 1545 (Fed. Cir. 1994) (en banc); *see discussion infra* part III.

67. Interpretation and scope given to the claims can be controlled by the amount and content of technical disclosure in the

specification. *See Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 979 (Fed. Cir. 1995) (specification and prosecution history are used to determine the scope of claims and the language they contain); *Hilton Davis Chem. Co. v. Warner-Jenkinson Co.*, 62 F.3d 1512, 1524 (Fed. Cir. 1995) (en banc) (the specification is used to determine the breadth given claim language when the claim language contains functional limitations or means-plus-function elements). Given *Markman* and *Hilton Davis*, a specification which contains only software may be capable of claiming only software, a specification containing only hardware may be limited to hardware-interpreted claims, and a specification containing both hardware and software can be interpreted as either software or hardware (assuming no file wrapper estoppel, equivalents, or prior art problems).

68. No amendment shall introduce new matter into the disclosure of an invention. 35 U.S.C. § 132 (1994); *see also* 37 C.F.R. § 1.118 (1996); Manual of Patent Examining Procedure §§ 608.04(a)-(c), 706.03(o) (1994) [hereinafter MPEP].

69. *Triax Co. v. Hartman Metal Fabricators, Inc.*, 479 F.2d 951, 956-57 (2d Cir. 1973).

70. The doctrine of inherency states that if an application discloses an object that inherently includes a function, property or advantage, then this function, property or advantage is necessarily disclosed by that application even though the application says nothing directly about the inherent function, property or advantage. *Technicon Instruments Corp. v. Cole Instrument Inc.*, 255 F. Supp. 630, 640-41 (7th Cir. 1966).

71. *See Paperless Accounting v. Bay Area Rapid Transit System*, 804 F.2d 659, 661 (Fed. Cir. 1986) (holding that added material to a patent is not new matter if it was inherent and/or well-known to the public or one of ordinary skill in the given Applicants' original disclosure); *In re Kline*, 474 F.2d 1325, 1328 (C.C.P.A. 1973) (if one of ordinary skill in the art, given Applicants' original disclosure, would clearly see that Applicants had possession of the added matter at the time of original filing, then that added matter would not be new matter).

72. 545 F.2d 141, 144 (C.C.P.A. 1976).

73. Data based on a randomly-selected sample of approximately 500 software patents studied by the author. For a few specific examples of structure or apparatus claims reviewed by the author, see U.S. Patent Nos. 5,349,518 (1994) (claim 3), 5,349,680 (1994) (claim 1), 5,359,724 (1994) (claim 18), 5,361,361 (1994) (claim 1), and 5,386,375 (1995) (claim 9). Other patents studied for this comment are on file with the author.

74. *See supra* note 67.

75. Most languages such as Lisp, PASCAL, FORTRAN, C, C++, Prolog, Parlog, assembly language, Basic, ML, Smalltalk, Ada, and COBOL, are designed to allow for code creation using the common software methodology of "divide and conquer."

76. Data based on a randomly-selected sample of approximately 500 software patents studied by the author. For a few specific examples of method or process claims reviewed by the author, see U.S. Patent Nos. 5,349,518 (1994) (claim 1), 5,349,680 (1994) (claim 6), 5,359,724 (1994) (claim 1), 5,361,361 (1994) (claim 7), and 5,386,375 (1995) (claim 1). Other patents studied for this comment are on file with the author.

77. *Guidelines, supra* note 57, at 430-32.

78. *See supra* part II.A.

79. 545 F.2d 152, 154 (C.C.P.A. 1976)

80. Claim preamble limitations have and have not been interpreted by courts as being material limitations to infringement of a claim. *Compare Derman v. PC Guardian*, No. 95-1263, 1995 WL 746237, at \*1 (Fed. Cir. Dec. 15, 1995) ("cassette tape chamber" in preamble was read as a material limitation to the scope of the claim since it was also referenced in the body of the claim); *In re Paulsen*, 30 F.3d 1475, 1479 (Fed. Cir. 1994) (terms appearing in a preamble may be deemed limitations of a claim when they "give meaning to the claim and properly define the invention" (citations omitted)); *with Loctite Corp. v. Ultraseal Ltd.*, 781 F.2d 861, 868 (Fed. Cir. 1985) (claim preamble language is not a limitation when it simply states a purpose or intended use of the invention); *Vaupel Textilmaschinen KG v. Meccanica Euro Italia S.P.A.*, 944 F.2d 870, 880 (Fed. Cir. 1991) (where a reference to the preamble in the claim body is to a term indicating a reference point for claimed structure, then the preamble does not become a claim limitation). It

seems reasonable that a claim limitation relied upon in the preamble to overcome a 35 U.S.C. § 101 rejection would estop the patent holder from subsequently arguing that the limitation is not material.

81. *See supra* part II.A.

82. *See also In re Warmerdam*, 33 F.3d 1354, 1360-61 (Fed. Cir. 1994) (dependent claim 5 held to be patentable subject matter while base claim 1 was unpatentable subject matter; the only structural addition to claim 5 was "memory").

83. *See* 35 U.S.C. §§ 286, 287 (1994) (describing limitations on damages); *see also* *Wine Ry. Appliance Co. v. Enterprise Ry. Equip. Co.*, 297 U.S. 387 (1936) (no duty to mark or give notice in lieu thereof if the patentee is using his/her patented process). However, for a CAFC opinion that may affect the breadth of interpretation given the *Wine Ry.* Supreme Court decision, see *Devices for Medicine, Inc. v. Boehl*, 822 F.2d 1062 (Fed. Cir. 1987) (where there is a combination of both structure and process claims, a lack of marking and notice may bar increased damage recovery for the process claims). The hypothetical situation is when Company A is infringing Company B's software patent from 1988 to 1996, where Company B's patent contains method claims and apparatus/structure claims. Assume that Company B was not marking disks with the many software patents Company B has in its patent portfolio. Company B finds out about the infringement by Company A in 1996. Company B can begin damage accumulation under 35 U.S.C. §§ 286 and 287 from the software apparatus/structure claims only from the time of notice to Company A or from the time a complaint is filed. Regarding the software process claims, an argument can be made under *Wine Ry.* that the infringement of the software process claims should be awarded damages from 1990 to 1996 under 35 U.S.C. § 286.

84. Means-plus-function claims are authorized in 35 U.S.C. § 112, para. 6 (1994).

85. *See In re Iwahashi*, 888 F.2d 1370, 1374 (Fed. Cir. 1989); *In re Alappat*, 33 F.3d 1526, 1540-41 (Fed. Cir. 1994) (en banc).

86. *See Alappat*, 33 F.3d at 1539, which is one example of this phenomenon where process claims and means-plus-function claims are similarly treated.

87. *Iwahashi*, 888 F.2d at 1374 (every step-by-step process, be it electronic or chemical or mechanical, involves an algorithm in the broad sense of the term, and it would be detrimental to our patent system to deny inventors patent protection on algorithms).

88. *Id.* at 1375 (quoting 35 U.S.C. § 112, para. 6 (1994)).

89. *Id.* at 1373.

90. 33 F.3d 1526 (Fed. Cir. 1994) (en banc); *see also In re Trovato*, 60 F.3d 807 (Fed. Cir. 1995) (en banc). *Trovato* is a significant case following *Alappat* because the CAFC initially held the claims at issue to be unpatentable under 35 U.S.C. § 101. *In re Trovato*, 42 F.3d 1376 (Fed. Cir. 1994). However, in a rehearing en banc, the court reversed the earlier finding and held for patentable subject matter in accordance with *Alappat*. *Trovato*, 60 F.3d 807.

91. *Alappat*, 33 F.3d at 1536.

92. A "new machine" is not pure software. A "new machine" is a computer, CPU, or memory, not just the pure software embodied as electrical signals on a disk, electrons in an SRAM cell, functional 0s and 1s on paper, charge on a telephone line, or a set of computer opcodes and bytes.

93. Claim 15 is reproduced *infra* part III.B.

94. *Alappat*, 33 F.3d at 1541.

95. *Id.* at 1545 (citations omitted). It is important to note that significant structure was disclosed in the specification of *Alappat*.

96. *See* Maria T. Arriola, *In re Alappat and Beyond: A New Approach to the Patentability of Mathematical Algorithms and Computer Programs in the United States?*, 5 FED. CIRCUIT B.J. 293 (1995); C. Mark Kittredge, *The Federal Circuit and Non-Patentable Subject Matter Under In re Alappat and In re Warmerdam*, 11 COMPUTER & HIGH TECH. L.J. 261 (1995); Sang Hui Michael Kim, *In re Alappat: A Strict Statutory Interpretation Determining Patentable Subject Matter Relating to Computer Software?*, 13 J.

MARSHALL J. COMPUTER & INFO. L. 635 (1995); James R. Goodman et al., *Toward a Fact-Based Standard for Determining Whether Programmed Computers are Patentable Subject Matter: The Scientific Wisdom of Alappat and Ignorance of Trovato*, 77 J. PAT. & TRADEMARK OFF. SOC'Y 353 (1995).

97. See generally Jonathan N. Geld, *General Does Not Mean Generic: Shedding Light on In re Alappat*, 4 TEX. INTELL. PROP. L.J. 71 (1995). Historically, this author used software means-plus-function software claims as a mechanism to obtain patent claims which would read on floppy disks, CDs, ICs, pure software code, etc., which contained the novel algorithm(s).

98. See discussion *infra* part IV.

99. See 35 U.S.C. § 112, para. 6 (1994).

100. See, e.g., U.S. Patent Nos. 5,083,262 (1992) (claim 8), 5,280,613 (1994) (claim 13), 5,125,087 (1992) (claim 19), and 5,201,042 (1993) (claim 9).

101. *In re Alappat*, 33 F.3d 1526, 1539 (Fed. Cir. 1994) (en banc).

102. Means-plus-function claims are generally assumed to be subject to a two-tiered equivalents analysis. A first equivalents analysis is performed under the language of 35 U.S.C. § 112, para. 6, and a second equivalents analysis is performed under the doctrine of equivalents. For discussion on the doctrine of equivalents, see *Scripps Clinic & Research Found. v. Genentech, Inc.*, 927 F.2d 1565, 1580-81 (Fed. Cir. 1991); *Hilton Davis Chem. Co. v. Warner-Jenkinson Co.*, 62 F.3d 1512 (Fed. Cir. 1995) (en banc); *Atlas Powder Co. v. E.I. Du Pont De Nemours & Co.*, 750 F.2d 1569, 1578-81 (Fed. Cir. 1984); *National Presto Industries, Inc. v. The West Bend Co.*, 76 F.3d 1185, 1190-92 (Fed. Cir. 1996).

103. Given *Markman* and *Hilton Davis*, discussed *supra* note 67, a hardware or "new machine" construction/interpretation cannot be properly given to all pertinent elements of the claim if the hardware is not properly disclosed in the specification.

104. See *supra* part III.B.2.

105. As an aside, the business impact of this situation is also great. GM is probably not enjoying its participation (e.g., third party discovery) in a multimillion-dollar law suit, when GM's objective is to keep a good working relationship with all of its suppliers, vendors and customers. Therefore, contributory infringement claims may create unavoidable business strain which may be a disincentive for IBM to bring suit to protect its patent rights.

106. See 37 C.F.R. §§ 1.16(b)-(c) (1996) for additional filing fees due to additional independent claims in excess of three and additional total claims in excess of twenty.

107. See discussion *infra* parts V, VI, VII.

108. See *Total Containment, Inc. v. Environ Products, Inc.*, 921 F. Supp. 1355, 1401-02 (E.D. Pa. 1995); *Marsh-McBirney, Inc. v. Jennings*, 22 U.S.P.Q.2d 1621, 1624 (C.D. Cal. 1991); *C.R. Bard, Inc. v. Advanced Cardiovascular Systems, Inc.*, 911 F.2d 670, 673 (Fed. Cir. 1990).

109. See *Aro Manufacturing Co., Inc. v. Convertible Top Replacement Co., Inc.*, 84 S.Ct. 1526, 1538 (1964) (when the patentee has sold the patented article or authorized its sale and has thus granted to the purchaser an "implied license to use," he clearly cannot thereafter restrict that use; "so far as the use of it was concerned, the patentee had received his consideration, and it was no longer within the monopoly of the patent."); *Carborundum Co. v. Molten Metal Equip. Innovations, Inc.*, 72 F.3d 872, 878 (Fed. Cir. 1995) (license may be implied, and implied license, like express license, is defense to patent infringement); *Intel Corp. v. ULSI Sys. Technology, Inc.*, 995 F.2d 1566, 1568 (Fed. Cir. 1993) (authorized sale of a patented product places that product beyond the reach of the patent); *McCoy v. Mitsubishi Cutlery, Inc.*, 67 F.3d 917, 921 (Fed. Cir. 1995) (citing *Bloomer v. Millinger*, 68 U.S. (1 Wall.) 340, 350 (1863)) (patentees are entitled to but one royalty for the patented machine, and consequently when a patentee has himself constructed the machine and sold it, or authorized another to construct and sell it, or to construct, use and operate it, and the consideration has been paid to him for the right, he has then to that extent parted with his monopoly, and ceased to have any interest whatsoever in the machine so sold or so authorized to be constructed and operated).

110. *See* *Cyrix Corp. v. Intel Corp.*, 846 F. Supp. 522, 539 (licensees' sale or other transfer of microprocessors to manufacturer exhausted patent holder's rights); *PCI Parfums et Cosmetiques Int'l v. Perfumania, Inc.*, 35 U.S.P.Q.2d 1159 (S.D.N.Y. 1995) (patent rights are exhausted subsequent to an authorized sale of the patented item).

111. *See* U.C.C. § 2-315 (1996) (discussing implied warranties for sales of goods).

112. *Carborundum*, 72 F.3d at 876 (absent direct infringement of patent claims, there can be neither contributory infringement nor inducement of infringement under 35 U.S.C. § 271).

113. For some case law on staple goods, see *Oak Indus., Inc. v. Zenith Elec. Corp.*, 726 F. Supp. 1525 (N.D. Ill. 1989) (if practice of patented method is incidental and necessary to practice of unpatented methods, device is "staple," and there can be no contributory infringement; however, if practice of patented method is not necessary or incidental to practice of unpatented methods, jury can find that device as whole is not staple and that seller is liable for contributory infringement); *Dawson Chem. Co. v. Rohm and Haas Co.*, 448 U.S. 176 (1980) (making or selling nonstaples especially made or adapted for use in practicing a patent is contributory infringement, but making or selling staples is not, however, useful in practicing a patent); *RCA/Ariola Int'l, Inc. v. Thomas & Grayston Co.*, 845 F.2d 773 (8th Cir. 1988) (finding that staple goods were common consumable type of goods such as office supplies); *Preemption Devices, Inc. v. Minnesota Mining & Mfg. Co.*, 630 F. Supp. 463, 471 n.10 (E.D. Pa 1985) (products clearly designed to be used in a system specified in the claims of the patent do not rise to the level of a staple article or commodity of commerce suitable for substantial non-infringing use). Typically, only consumables or inexpensive material that have wide application are found to be staple. *See* *Hodosh & Richardson-Vicks, Inc. v. Block Drug Co.*, 833 F.2d 1575, 1578 (Fed. Cir. 1987) (to determine if defendant's goods are staple articles, the Court must look at the entire device, not just the part capable of practicing the claims); *Haworth, Inc. v. Herman Miller, Inc.*, 37 U.S.P.Q.2d 1080 (W.D. Mich. 1994) (in assessing whether a product is a staple article of commerce, the quality, quantity and efficiency of the suggested alternate uses are to be considered).

114. *See supra* note 102.

115. *See In re Alappat*, 33 F.3d 1526, 1583 (Fed. Cir. 1994) (en banc) (a software process is often interchangeable with a hardware circuit). This statement may be interpreted to mean that hardware and software are equivalents under the doctrine of equivalents.

116. *See* *Valmont Indus., Inc. v. Reinke Mfg. Co.*, 983 F.2d 1039, 1043 (Fed. Cir. 1993) ("equivalent" under doctrine of equivalents results from insubstantial change which, from the perspective of one of ordinary skill in the art, adds nothing of significance to the claimed invention; equivalent under doctrine of equivalents, though not literally meeting claims, still infringes patent).

117. *Hilton Davis Chem. Co. v. Warner-Jenkinson Co., Inc.*, 62 F.3d 1512, 1517 (Fed. Cir. 1995) (en banc) (discussing the scope of doctrine of equivalents and its applications where insubstantial differences exist between the claims and the infringing product); *see also In re Donaldson Co., Inc.*, 16 F.3d 1189 (Fed. Cir. 1994) (means-plus-function claims should be interpreted in light of the disclosure in the specification).

118. *See* *Graver Tank and Mfg. Co. v. Linde Air Prods. Co.*, 339 U.S. 605, 608-09 (1950) (equivalents determined using the function-way-result test under which if the accused infringing product performs a similar function, in a similar way, with a similar result, then equivalents will hold); *see also* *Pennwalt Corp. v. Durand-Wayland, Inc.*, 833 F.2d 931, 925 (Fed. Cir. 1987) (equivalents must be found on an element-by-element basis or a limitation-by-limitation basis within the claims).

119. What is even more troubling is that the IBMs of the world can no longer turn to U.S. copyright law for protection as a secondary or primary means of protection. Not only is independent creation a defense to copyright infringement where there is no access and substantial similarity, the courts are now weakening copyright protection for higher levels of hierarchical abstraction within software. *See* *Lotus v. Borland*, 49 F.3d 807 (1st Cir. 1995) (finding that a software user interface was not patentable since the software is a method of operation barred from protection by the Copyright Act). Ideas and algorithms of software patents cannot safely turn to copyright law for protection in the near future. This author feels that software protection must come from somewhere within U.S. intellectual property law. The best choice within U.S. intellectual property law is the patent law because copyright law provides too long a period of protection (75-100 years), requires no governmental examination whatsoever, requires no publication notice as do patents, and will almost certainly require a multimillion-dollar law suit to determine the scope of protection before someone will license the technology for a substantial fee.

120. 53 F.3d 1583, 1584 (Fed. Cir. 1995)

121. *Guidelines, supra* note 57.

122. Until a precedential opinion is rendered by the CAFC, expect trouble regarding article of manufacture claims in the USPTO. This author has had encounters with Examiners subsequent to *Beauregard* where *Beauregard*-type claims were singled out and scrutinized in excessive detail under 35 U.S.C. § 101. This attitude may continue in the USPTO even after a precedential opinion is issued by the CAFC. It is not uncommon for art groups in the USPTO to adopt an attitude that looks like nonacquiescence with respect to the CAFC. Nonacquiescence to the CAFC on patent matters seems ridiculous and totally unjustified. Nonetheless, the reality is that it is common to receive vigorous 35 U.S.C. § 101 rejections from the USPTO in 1996 that completely ignore all of the pro-software-patent opinions from the 1990s, and instead rely totally on the anti-software patent opinions from the 1970s.

123. *See, e.g., In re Alappat*, 33 F.3d 1526 (Fed. Cir. 1994) (en banc), discussed *supra* part III.A.

124. 33 F.3d 1354 (Fed. Cir. 1994).

125. *Id.* at 1357. Claim 1 recites: "A method for generating a data structure which represents the shape of physical object in a position and/or motion control machine as a hierarchy of bubbles, comprising the steps of: first locating the medial axis of the object and then creating a hierarchy of bubbles on the medial axis."

126. *Id.* at 1358. *Warmerdam* is an important case to keep in mind since it can plausibly be interpreted as holding that adding the limitation "memory" to a nonstatutory claim under 35 U.S.C. § 101 will render this claim statutory under 35 U.S.C. § 101. This technique could work wonders for mathematical algorithms, data structures, and other per se nonstatutory subject matter, although the USPTO will put up a hard fight in many of these circumstances.

127. *See In re Beauregard*, 53 F.3d 1583, 1584 (Fed. Cir. 1995).

128. *Guidelines, supra* note 57, at 429.

129. *Id.* at 429-30.

130. *Id.* at 429.

131. *Cf. Parker v. Flook*, 437 U.S. 584, 593 (1978) (claims must be examined as a whole under 35 U.S.C. § 101 and not in selected pieces); *Jones v. Hardy*, 727 F.2d 1524, 1527 (Fed. Cir. 1984) (reducing the claim to its general idea and rendering the claim not patentable on this basis is improper); *In re Gulack*, 703 F.2d 1381, 1384 (Fed. Cir. 1983) (simply because a portion of a claim is printed matter does not mean that this material should be discounted from the claim under 35 U.S.C. §§ 101, 102, or 103). Therefore, selectively ignoring any portion of a claim to render a 35 U.S.C. § 101 decision is improper.

132. *Guidelines, supra* note 57, at 429.

133. Article of manufacture claims which read on computer readable media have caused the USPTO some anxiety for a significant period of time. For example, in June 1995, the USPTO provided *Proposed Guidelines for Computer-Implemented Inventions*, 60 Fed. Reg. 28,778 (1995). In this document, the USPTO seemed to suggest that article of manufacture claims for software stored on a disk would be allowed only if "the specific physical configuration of the substrate of the computer-readable storage medium that represents the data" is specifically laid forth in the claim. *Id.* at 28,780. This author and other practitioners have interpreted this language to mean that if a CD was claimed, the interrelation of the trenches (or like structure) on the CD substrate must be claimed to render the claim patentable under 35 U.S.C. § 101. If a memory IC was claimed, specific limitations of floating gate or ROM transistor placement and interconnection must be claimed. In many cases, residual anxiety lived on in the USPTO for many years, often even contrary to sweeping new federal court decisions such as *Alappat* and *Iwahashi*. Therefore, the USPTO anxiety over article of manufacture claims must be investigated and understood so that when residual anxiety about these new software claims surfaces in some art groups and examiners in the USPTO, the software patent practitioners can better avoid costly appeals and abandoned cases by providing a more intelligently crafted specification which leaves the attorney with an arsenal of possible responses to the Examiner's 35 U.S.C. § 101 rejection.

134. "Plurality of binary values" can be replaced with "computer instructions," "opcodes," "subroutines," "executable code," or "software," depending upon the mood of the Examiner and the tastes of the inventor. The "storage media" in the preamble may be

replaced with "storage medium," "computer-readable medium," "magnetic medium," etc., as is deemed appropriate for the particular invention.

135. *Guidelines, supra* note 57, at 429.

136. *See id.*

137. 37 C.F.R. § 1.83(a) (1996).

138. 37 C.F.R. § 1.71 (1996).

139. Anything in the original specification can be selectively added to the claims to overcome 35 U.S.C. § 101 hurdles. *See supra* text accompanying notes 69-71.

140. *See Guidelines, supra* note 57, at 429.

141. CD, magnetic disk, tape, and integrated circuit storage media.

142. *See* 37 C.F.R. §§1.16(b)-(c) (1996).

143. The term of patent is no longer seventeen years from issuance but twenty years from filing. 35 U.S.C. § 154 (1994 & Supp. 1996).

144. *Guidelines, supra* note 57, at 429.

145. *Id.*

146. *See* 37 C.F.R. § 1.83 (1996); 35 U.S.C. § 112, para. 1 (1994).

147. A disclosure in a patent application may be deliberately supplemented or completed by reference to the disclosure set forth in other patents or publicly available documents. *See In re Lund*, 376 F.2d 982, 989 (C.C.P.A. 1967). However, caution should be used because for enablement, any public material may be disclosed, but the material needs to support the claim or "essential material" needs to either be fully disclosed in the text of the specification or by a referenced U.S. patent. *Quaker City Gear Works, Inc. v. Skil Corp.*, 747 F.2d 1446, 1455 (Fed. Cir. 1984); *see also* MPEP § 608.01(p) (1994). Also, if the specification fails to disclose specific details, one may argue that this material is: (1) well known to one of ordinary skill in the computer art; or (2) properly claimed using the doctrine of inherency under the new matter case law discussed previously. *See supra* text accompanying notes 69-71.

148. *See Kayton, Nonobviousness of the Novel Invention-35 U.S.C. § 103*, in PATENT PRACTICE (5th ed., vol. 1), at 5-1 (discussing 35 U.S.C. § 103 and ways in which 35 U.S.C. § 103 can be avoided using CAFC and CCPA case law).

149. If elements are combined to obtain a new result, then the item or process should be patentable and not rejected under 35 U.S.C. § 103, regardless of the fact that old elements or knowledge are embodied in the patented structure. *See Seymour v. Osborne*, 78 U.S. (11 Wall.) 516, 541 (1871) (U.S. historical adoption of the new result test); *Sakraida v. Ag Pro, Inc.*, 425 U.S. 273, 281-82 (1976) (combination of known elements are patentable if they produce a "synergistic result" or a new or different function); *Ryko Mfg. Co. v. Nu-Star Inc.*, 950 F.2d 714, 716 (Fed. Cir. 1991); *Alcon Lab. Inc. v. Allergan Inc.*, 17 U.S.P.Q.2d 1365, 1373 (N.D. Tex. 1990) (even without "synergy" a combination of old elements may be patentable under 35 U.S.C. § 103); *American Hoist & Derrick Co. v. Sowa & Sons, Inc.*, 725 F.2d 1350, 1360 (Fed. Cir. 1984) (claims of combination of old elements may be allowed even without a new result or function); *In re Raynes*, 7 F.3d 1037, 1039 (Fed. Cir. 1993) (USPTO bears burden of proving that combination of old elements is obvious).

150. If the software operational limitations of the article of manufacture claims are not taught in the prior art, then the claim, when taken as a whole as required under 35 U.S.C. § 103, is patentable under 35 U.S.C. § 103. *See In re Fine*, 837 F.2d 1071, 1075 (Fed. Cir. 1988); *In re Gulack*, 703 F.2d 1381, 1384 (Fed. Cir. 1983) (the claim must be read as a whole and software limitations cannot be dissected from the prior art to support a rejection under 35 U.S.C. § 103); *Jones v. Hardy*, 727 F.2d 1524, 1528 (Fed. Cir. 1984) (the invention cannot be reduced to an idea (e.g. putting software on a disk) with the goal of using the obviousness of this idea to render the

total claim unpatentable).

151. See *In re Spinnoble*, 405 F.2d 578 (C.C.P.A. 1969); *In re Peehs*, 612 F.2d 1287 (C.C.P.A. 1980).

152. For a detailed discussion of other 35 U.S.C. § 103 rebuttal strategies, see Kayton, *supra* note 148.

153. See *Kori Corp. v. Wilco Marsh Buggies & Draglines, Inc.*, 761 F.2d 649, 656 (Fed. Cir. 1985) (court probably awarded damages on the entire market value theory of infringing machines, rather than on the portion of the machine that included the patented product); *TWM Mfg. Co., Inc. v. Dura Corp.*, 789 F.2d 895, 900 (Fed. Cir. 1986) (providing another example of the entire market value theory); *State Indus., Inc. v. Mor-Flo Indus., Inc.*, 883 F.2d 1573, 1580 (Fed. Cir. 1989) (larger damages are acceptable when the patented feature is the basis for customer demand).

154. 35 U.S.C. § 284 (1994); see also *SmithKline Diagnostics, Inc. v. Helena Laboratories Corp.*, 926 F.2d 1161, 1163 (Fed. Cir. 1991) (discussing reasonable royalty, established royalty, and lost profits damages).

155. A complete discussion on the doctrine of established royalty as a measurement of infringement damage is a paper in and of itself. For case law in this area, see *Seymour v. McCormick*, 57 U.S. (16 How.) 480, 481 (1853); *Rude v. Westcott*, 130 U.S. 152, 154 (1889); *Faulkner v. Gibbs*, 199 F.2d 635, 638 (9th Cir. 1952) (established royalty must be set prior to the infringement currently at issue, paid by an adequate number of people to imply reasonableness, uniform at the place where the licenses are issued, not paid under threat of litigation, for similar rights/activities under the patent); *Deere & Co. v. Int'l Harvester Co.*, 710 F.2d 1551, 1557 (Fed. Cir. 1983) (one licensee is not enough to create an established royalty); *Allen Archery, Inc. v. Browning Mfg. Co.*, 898 F.2d 787, 788 (Fed. Cir. 1990).

156. Royalty payments and patent awards in the federal courts for tens to hundreds of millions or even billions of dollars are not that unusual. Given rough cost-benefit analysis, a few million dollars spent on a settlement will be well spent if it avoids the risk of having to pay hundreds of millions of dollars, even if the risk is small (especially since the defense itself, even if successful, is likely to cost the defendant on the order of millions of dollars). From the defendant's perspective, the desirability of defending this law suit may be diminished by providing method of manufacture claims incorporating the software innovation.

157. For case law in this area, see *Suffolk v. Hayden*, 70 U.S. (3 Wall.) 315 (1865); *Coupe v. Royer*, 155 U.S. 565 (1895); *Georgia-Pacific Corp. v. United States Plywood Corp.*, 318 F. Supp 1116 (S.D.N.Y. 1970); *Panduit Corp. v. Stahlin Bros. Fibre Works*, 575 F.2d 1152 (6th Cir. 1978); *Bio-Rad Lab., Inc. v. Nicolet Instrument Corp.*, 739 F.2d 604 (Fed. Cir. 1984); *Del Mar Avionics, Inc. v. Quinton Instrument Co.*, 836 F.2d 1320 (Fed. Cir. 1987); *Fromson v. Western Litho Plate & Supply Co.*, 853 F.2d 1568 (Fed. Cir. 1988); *Polaroid Corp. v. Eastman Kodak Co.*, 16 U.S.P.Q.2d 1481 (D. Mass. 1990).

158. See 37 C.F.R. §§ 1.16, 1.17 (1996) for fee amounts.

159. This technique may have the additional advantage of avoiding the additional cost and lost prosecution time associated with a restriction requirement.

160. For case law on the issue of damages based on lost profits, see *Yale Lock Co. v. Sargent*, 117 U.S. 536 (1886); *Story Parchment Co. v. Paterson Parchment Paper Co.*, 282 U.S. 555 (1931) (even if lost profit damages cannot be proven exactly, the proof of some damage is enough for jury to set an award as best as it can); *Datascope Corp. v. SMEC Inc.*, 879 F.2d 820 (Fed. Cir. 1989); *Paper Converting Machine Co. v. Magna-Graphics Corp.*, 745 F.2d 11, 21 (Fed. Cir. 1984) (causation requirement satisfied for lost profit when reasonable probability of causation exists); *Kaufman Co. v. Lantech, Inc.*, 926 F.2d 1136 (Fed. Cir. 1991); *Gyromat Corp. v. Champion Spark Plug Co.*, 735 F.2d 549 (Fed. Cir. 1984); *Lam, Inc. v. Johns-Manville Corp.*, 735 F.2d 604 (Fed. Cir. 1983).

161. *Panduit Corp. v. Stahlin Bros. Fibre Works*, 575 F.2d 1152, 1156 (6th Cir. 1978); see also *Bio-Rad Lab. v. Nicolet Corp.*, 739 F.2d 604, 616 (Fed. Cir. 1984).

162. It must also be remembered that many companies create software solutions in an attempt to further goals other than the mere sale of the software itself. For example, Intel may create a software CAD tool useful for designing floating gate memory products. Since Intel does not plan on selling the software and since Intel has no interest in suing various software vendors due to the fact that these software companies provide Intel with other software tools, pure software claims may have little value to Intel. However, method of manufacture claims which prevent a competitor from either buying or developing knock-off CAD tools and prevent competitors from using these knock-off tools to manufacture competing floating gate memory products are advantageous for companies like Intel.

163. A controversial issue with respect to lost profits damages and method of manufacture claims lies in the fact that software usually improves upon a prior art manufacturing process which was previously used by the infringer. The defendant will argue that noninfringing or prior art manufacturing substitutes limit the damages for infringement. However, many courts have determined that if the patented process and the prior art noninfringing substitutes vary significantly in terms of cost, functionality, performance, advantages, disadvantages, time to market, profit potential, and feasibility, then alternatives will not reduce the lost profit compensation. *See, e.g.,* Radio Steel & Mfg. Co. v. MTD Prods., Inc., 788 F.2d 1554, 1555 (Fed. Cir. 1986); TWM Mfg. Co., Inc. v. Dura Corp., 789 F.2d 895, 900 (Fed. Cir. 1986); Smithkline Diagnostics, Inc. v. Helena Lab. Corp., 926 F.2d 1161, 1163 (Fed. Cir. 1991); Kaufman Co. v. Lantech, Inc., 926 F.2d 1136, 1140 (Fed. Cir. 1991); Uniroyal, Inc. v. Rudkin-Wiley Corp., 939 F.2d 1540, 1544 (Fed. Cir. 1991). Therefore, if the patent holder can show that the method of manufacture as a whole is superior over prior art substitutes, then the defendant will not be allowed to become unjustly enriched from the use of the patent holder's manufacturing benefits.

164. This action is authorized by the Omnibus Trade and Competitiveness Act of 1988, codified at 35 U.S.C. §§ 271(g), 287(b), 295 (1994); 19 U.S.C. § 1337 (1994).

165. *See Guidelines, supra* note 57, at 429-30; Massachusetts Inst. of Technology v. AB Fortia, 774 F.2d 1104 (Fed. Cir. 1985) (invalid claims result in no violation of the Tariff Act); Texas Instruments Inc. v. U.S. I.T.C., 854 F.2d 1327 (Fed. Cir. 1988) (claims were held invalid and not infringed in ITC, resulting in lost time, greater cost, and review in the CAFC; eventually reversed in part on validity holding); Levi Strauss & Co. v. Golden Trade S.r.L., Nos. 92 Civ. 1667 (RPP), 90 Civ. 6291 (RPP), 90 Civ. 6292 (RPP), 1995 WL 710822 (S.D.N.Y. 1995) (claims attacked under 35 U.S.C. § 101).

166. *Guidelines, supra* note 57, at 427 n.27 (citation omitted).

167. *See In re Warmerdam*, 33 F.3d 1354, 1358, 1361-62 (Fed. Cir. 1994) (the limitations of "memory" and "machine" rendered claim 5 patentable, whereas "data structure" did not render claim 6 patentable). The court stated that "data structure" is not one of the categories enumerated in 35 U.S.C. § 101 and that a "data structure" may be either logical or physical in nature, where the logical instantiation is definitely an "abstract idea" and not patentable under 35 U.S.C. § 101. The court seems to suggest that if one focuses on the physical limitations and physical interconnectivity of a data structure (not just the logical relationship), this will render the claim patentable. *Id.* at 1362. *See also In re Bradley*, 600 F.2d 807, 811 (C.C.P.A. 1979) (claims to data structure held valid under 35 U.S.C. § 101 due to physical structural limitations in the claims and specification); *In re Lowry*, 32 F.3d 1579, 1583 (Fed. Cir. 1994) (data structure held patentable since the structure recited was specific electrical or magnetic structural elements in a memory).

168. 32 F.3d 1579, 1580-81 (Fed. Cir. 1994).

169. *Id.* at 1581.

170. *Id.* at 1582.

171. *Id.*

172. *Id.* at 1583.

173. *Id.* at 1583-84.

174. *Guidelines, supra* note 57, at 427.

175. *Id.* The theory behind this rule derives from the printed matter doctrine, which stands for the theory that if an invention is primarily embodied in legible form on printed paper, then copyright should protect the work and patents should not be used. *See In re Miller*, 418 F.2d 1392 (C.C.P.A. 1969); *In re Jones*, 373 F.2d 1007 (C.C.P.A. 1967).

176. *Guidelines, supra* note 57, at 427.

177. *Id.*

178. *Id.*
179. *See supra* text accompanying note 166.
180. *Guidelines, supra* note 57, at 427.
181. This position directly contradicts the definition of data structures contained in the Examination Guidelines. *See supra* text accompanying note 166.
182. *Lowry*, 32 F.3d 1579, 1583 (Fed. Cir. 1994) (quoting *In re Bernhart*, 417 F.2d 1395, 1399 (C.C.P.A. 1969)).
183. *See supra* note 175.
184. *Guidelines, supra* note 57, at 429.
185. In fact, the very software heart of Wintel-based computer systems is stored as a Read Only Memory Basic Input/Output System (ROM BIOS) that is comprised of all read only memory (ROM) cells. *See* PETER NORTON, *INSIDE THE PC* (1995). Furthermore, many microcontrollers, such as the HC11, HC16, HC05, and like microcontrollers available from Motorola Inc., are provided with ROM sections solely to store software executable code. These ROM areas are inadequate to store most databases in existence today since databases usually must be written to or changed over the course of time.
186. For examples of the dynamic nature of data structures, see THOMAS L. NAPS & BHAGAT SINGH, *INTRODUCTION TO DATA STRUCTURES WITH PASCAL* (1986).
187. *See In re Beauregard*, 53 F.3d 1583, 1584 (Fed. Cir. 1995); *In re Alappat*, 33 F.3d 1526, 1545 (Fed. Cir. 1994) (en banc).
188. 37 C.F.R. § 1.16(a) (1996).
189. *See* 35 U.S.C. § 121 (1994); MPEP § 806 (1994). The correctness and wisdom of the USPTO restricting inventions in N-way restrictions is a subject of a paper in and of itself. The facts are that the restriction requirement: (1) exists and will be used; (2) is expensive to the client in terms of maintenance fees, attorney fees, and filing fees; (3) is difficult to overcome in the USPTO; (4) is more detrimental to client's patent rights than ever before due to the new twenty-year patent term; (5) is not wisely overcome by stating that all of the restricted groups will stand or fall together due to obviousness over one another; (6) raises prior art inconsistency problems between divisional applications which create inequitable conduct problems; and (7) raises grave file wrapper estoppel claims between sibling and parent applications. The client needs to be aware of the possibility of a restriction requirement so that claims can be prioritized from a financial and business perspective and a cost/benefit analysis can be performed. A typical response is either to: (1) file divisional applications and ramp the claims in each divisional back up to twenty total claims so that the client obtains maximum protection for his money, and so that the USPTO is doing double or triple the work for the same invention; or (2) abandon certain claims due to financial limitations and obtain a patent that is less valuable than deserved.
190. *See, e.g.*, U.S. Patent No. 5,274,815 (1993) (Fig. 1).
191. *See, e.g.*, U.S. Patent No. 5,386,375 (1995) (Fig. 6).
192. *See* 37 C.F.R. § 1.96 (1996); *see, e.g.*, U.S. Patent Nos. 5,088,034 (1992) (Figs. 14-18) and 5,375,299 (1994) (Appendix).
193. *See, e.g.*, U.S. Patent No. 5,386,375 (1995) (Figs. 1, 3-5).
194. *See, e.g.*, U.S. Patent No. 5,274,815 (1993) (Figs. 3-4).
195. 37 C.F.R. § 1.71 (1996); MPEP § 608.01 (1994).
196. *See supra* text accompanying notes 69-71.

197. It is ironic that few believed software was even patentable at all in the 1970s, while today software is the only technology (or at the very least one of the few technologies) that is patentable in view of 35 U.S.C. § 101 under three of the four enumerated categories (a process, a manufacture, and a machine).