The Question Left Unanswered In *WMS Gaming*: What Is the Algorithm?

David C. Bohrer and Michael I. Frankel

Building on recent precedent recognizing the patentability of software and software-related inventions, the Federal Circuit, in *WMS Gaming, Inc. v. International Game Technology*,[1] addressed the scope of such inventions drafted in means-plus-function format. The Federal Circuit held that a disclosed algorithm was properly part of the means for performing the identified function.[2]

In theory, restricting a software-related means-plus-function claim to a disclosed algorithm is both straightforward and well-founded. Decisions in subsequent cases reflect the accurate understanding that this is what was taught in *WMS Gaming*.[3] Moreover, within the software profession, an algorithm is well-understood to express in human language the essential steps that a computer is intended to take to achieve a particular result.

In practice, however, it is very difficult to identify the algorithm in the disclosures supporting a software-related claim. Neither *WMS Gaming* nor any of its progeny provide clear guidance as to how to determine the disclosed algorithm.[4] Further confusing matters is that *WMS Gaming* applied several different, if not outright inconsistent, methods of discerning the disclosed algorithm.

The potential consequences of not having clear standards for describing an algorithm are both drastic and pervasive. Patent attorneys and companies are denied the tools necessary to properly evaluate the scope and validity of their own and others' software-related patents. Worse yet, misconstruing the algorithm results in wrongly decided infringement rulings. A case in point is none other than *WMS Gaming*, in which the failure to properly discern the disclosed algorithm resulted in the erroneous ruling that the accused electronically controlled slot machine infringed the patent.

These circumstances present a compelling case for rejecting traditional methods for disclosing the algorithm in software-related patents. These traditional methods are unduly narrow, confusing, ill-suited to describing what the software does, and even out-dated. The better solution is for patent applicants, examiners, and courts to apply the Unified Modeling Language (UML), which was adopted by the software industry in 1997 as a standard method to describe software algorithms.

David C. Bohrer, a patent litigator in Dechert LLP's Silicon Valley Office, has a nationwide practice representing high-technology clients. Mr. Bohrer can be reached at *david.bohrer@dechert.com*.
Michael I. Frankel practices in the areas of intellectual property and antitrust in Dechert LLP's Philadelphia office. Mr. Frankel can be reached at *michael.frankel@dechert.com*.

Means-Plus-Function Claim Limitations

The discussion of algorithms and *WMS Gaming* is best understood after developing some common ground regarding claims drafted using means-plus-function language. A means-plus-function limitation allows a patent applicant to express an element in a patent claim as a "means for" performing a specified function without having to recite in the claim the structure necessary to perform that function.[5] However, to protect against overbreadth and ambiguity, the patent laws provide that such a claim does not cover every structure capable of performing the recited function; rather, it is limited to the particular structure disclosed in the specification.[6]

Construing means-plus-function claim limitations involves determining, as a matter of law, the claimed function and the structure corresponding to that function.[7] "Function," practically speaking, is the activity described in the language immediately following the "means for" phrase. The claim language itself defines the functional aspect of a means-plus-function claim limitation.[8]

"Structure," practically speaking, is what is referred to by the "means for" language in the claim. After the function is identified, the structure is identified by looking to the patent specification (the written description, the drawings, and other claims in the patent) to identify the structure corresponding to the claimed function.[9] Structure encompasses *only* that structure that is necessary to perform the recited function.[10]

A means-plus-function limitation is literally infringed by an accused device when the accused device performs the identical function specified in the claim and, in addition, the accused device employs a structure identical or equivalent to the structure described in the patent specification.[11] The test for determining whether the structure in an accused device is equivalent to the

structure in the means-plus-function limitation is whether the differences between the two structures are insubstantial.[12] The determination of what constitutes structural equivalents of the identified structure is a question of fact for the jury.[13] In comparison, an accused device can infringe under the doctrine of equivalents without infringing literally under 35 U.S.C. § 112, ¶ 6 because the doctrine requires only substantially the same function, not identicality of a function as in § 112, ¶ 6.[14]

Why Claim a Software-Related Invention in Means-Plus-Function Format?

Historically, computer programs were viewed as textual representations of mathematical algorithms and thus potentially appropriate subjects for copyright protection, but not for patents.[15] In 1994, the *en banc* Federal Circuit decided *In re Alapatt*,[16] clearing the way for much greater software patent protection. The decision established that claiming an inventive algorithm as part of some physical apparatus such as a general purpose computer or standard hardware or memory element satisfied the requirements for patentability. The *Alapatt* court reasoned that "a general purpose computer in effect becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from program software."[17] The insight that patent applicants derived from this analysis was that, if they wanted to patent software, they needed only to define their claims in terms of a computer program implemented in a machine.[18]

Because the *Alapatt* decision dealt with means-plus-function elements and spoke of a computer programmed to perform an inventive algorithm as a "means," many patent prosecutors took the case as a signal that this claiming format provided an easy method of claiming inventions featuring programmed devices.[19] A large number of software patents have in fact been claimed in means-plus-function language, and a correspondingly large percentage of these patents have matured to where the disputed terms are now being litigated.[20]

This is not to say that *Alapatt*'s legacy is confined to the use of the means-plus-function format to protect novel software. Other claiming techniques, such as claiming the steps performed by the programmed instructions as a process, have been found to satisfy the patentability requirements.[21] The nature and scope of such other claiming techniques, as well as the potentially broader patent coverage available under these techniques, is the subject of other articles. Suffice it to say that in the post-*Alapatt* era it is well-settled that software related inventions are *patentable* but that the exact *scope* to be given software claims is equally unsettled. The scope issue, at least insofar as means-plus-function claims were concerned, remained to be decided in *WMS Gaming*.

Scope of Software-Related Patents Drafted in Means-Plus-Function Format

WMS Gaming arose out of a cease and desist letter sent by International Game Technology (IGT) indicating that WMS' WMS 400 slot machine infringed IGT's patent (the Telnaes patent). WMS responded by filing a declaratory judgment action. Following a bench trial, the district court held that the Telnaes patent was valid and infringed. On appeal, the Federal Circuit affirmed the findings of validity and infringement[22] but also found significant error in the district court's construction of the software-related means-plus-function elements of the disputed claims.

The dispute centered on a slot machine design that electronically manipulates the odds of winning. To maintain and increase the appeal to players, the market demanded slot machines with higher payoffs. To generate higher payoffs without decreasing the slot machine's profitability, the probability of winning any one play had to be decreased. However, to decrease the payout odds for a mechanical slot machine, either more reels or larger reels with more stop positions must be added. This approach reduced the machine's appeal since players perceive physically larger machines or machines with more reels as being "less good" in terms of winning and payout chances. So the challenge faced by the inventors of the Telnaes patent was to increase payouts and decrease the probability of winning without increasing the physical size of the machine.

The Telnaes Patent

The Telnaes patent discloses an electronically controlled slot machine that decreases the probability of winning while maintaining the external appearance of a standard slot machine. To decrease the probability of winning, the control circuitry extends the reel "virtually" to include a range of numbers greater than the number of actual stop positions and then maps these numbers non-uniformly to the actual stop positions. When in play, the control circuitry randomly determines the stop position of each reel and then stops the reels at the randomly determined positions. The reels' only function is to display the randomly chosen result.

For example, if a reel with 22 stop positions contains a cherry symbol in two positions, the probability of stopping at a cherry is two out of 22 or 0.0909. If the reels were "virtually" extended to include 44 virtual

stop positions with the cherry symbol mapped to numbers seven, 22, and 37, then the probability of stopping at the cherry is three out of 44 or 0.0681.[23] Thus, the probability of stopping on the cherry symbol is reduced without altering the physical appearance of the slot machine.

The WMS 400

Unlike the apparatus disclosed in the Telnaes patent, which determines the stop positions first and then determines the payoff based on those stop positions, the accused product, the WMS 400, calculates the payoff first and then chooses stop positions that represent that payoff. The WMS 400 randomly selects a number from a range of 1 to 632. Once selected, that number is mapped to a first multiplier stored in memory. Next, a second number is randomly selected from a second range of 1 to 632, and once selected, it is mapped to a second multiplier stored in memory. The two multipliers are then multiplied together to determine the payout value. If there is only one way to display the payout, then the reels are stopped on those symbols. If, however, there is more than one arrangement of symbols that can indicate the payout, then a third random number is necessary. This third random number determines which one of the possible reel arrangements will be displayed.

Claim Construction

The construction of Claim 1 of the Telnaes controlled the determination of infringement. Claim 1 reads as follows:

> A game apparatus, comprising:
>
> [1] a reel mounted for rotation about an axis through a predetermined number of radial positions;
>
> [2] means to start rotation of said reel about said axis;
>
> [3] indicia fixed to said reel to indicate the angular rotational position of said reel;
>
> [4] means for assigning a plurality of numbers representing said angular positions of said reel, said plurality of numbers exceeding said predetermined number of radial positions such that some rotational positions are represented by a plurality of numbers;
>
> [5] means for randomly selecting one of said plurality of assigned numbers; and
>
> [6] means for stopping said reel at the angular position represented by said selected number.

The parties agreed that the accused device contained the first three claim elements. The parties' dispute centered on the last three means-plus-function elements of claim 1.

Of particular significance was the construction given by both the district court and the Federal Circuit to the "means-for-assigning" element. The district court, relying on the parties' stipulation that the Telnaes patent disclosed "a microprocessor, or computer, to control the operation of the slot machine," held that the "means-for-assigning" element was satisfied by "an algorithm executed by a computer." Under the district court's construction, the "means-for-assigning" limitation covered "any table, formula, or algorithm" that performed the claimed function.

On appeal, the Federal Circuit held that the district court had erred by failing to limit the "means-for-assigning" element to the exact algorithm disclosed in the Telnaes patent specification. The Federal Circuit stated that, "[i]n a means-plus-function claim in which the disclosed structure is a computer or microprocessor, programmed to carry out an algorithm, the disclosed structure is not the general purpose computer, but rather the special purpose computer programmed to perform the disclosed algorithm."[24]

WMS Gaming effectively limits the software-related means-plus-function limitation to the disclosed algorithm. In theory, this seems straightforward; indeed, subsequent cases reflect an accurate understanding that this was the holding of *WMS Gaming*.[25]

The theoretical emphasis on algorithms is further supported by the fundamental nature and purposes of computer hardware and software. Although computer hardware is a real machine, when turned on, it does not perform any of its operations but rather waits for some signal from the outside to set the machine in motion.[26] The computer hardware needs the software or program to tell it what behavior is desired.[27] The creative and inventive action taken by the computer comes in with the software.

Accordingly, software's *raison-dêtre* is causing the computer to behave in a certain fashion.[28] The question then is how to describe the creative and novel behavior that the software causes the computer to perform.

There are different levels of abstraction that are used to describe what the software is telling the computer to do. The lowest level of abstraction is machine language, also known as object code, which only the computer understands (as opposed to people) and which is what the computer is actually being told to do. A somewhat higher level of abstraction is source code (*e.g.,* Ada,

Pascal, C++, JAVA, COBOL, Fortran, etc.), which is not directly understood by the computer until it is compiled into machine code but which also is difficult for all but a small number of persons skilled in the art to understand. At a still higher level of abstraction are algorithms, which are easier for professional persons to understand and convey the action that the computer is intended to perform (as compared to what the computer is actually told to do).

Algorithms are analogous to engineering blueprints; they provide a medium through which to communicate a particular software design and to confirm agreement among the interested parties as to the intended action to be taken by the computer. In short, it is entirely appropriate for the Federal Circuit to choose to focus on software algorithms when comparing accused and claimed software.

In practice, however, the determination of the precise algorithm disclosed in the patent can be problematic and complex. These practical issues are manifest in none other than the decision that focused software comparisons on algorithms—*WMS Gaming*.

How Should the Disclosed Algorithm Be Described?

Having announced the rule that the disclosed algorithm is a limitation on the claim, the *WMS Gaming* decision proceeds to apply the rule without first discussing what an algorithm is, how it is used, and how it is usually (or should be) expressed. There is no discussion in *WMS Gaming* about how one skilled in the art of software engineering would be expected to review a patent to determine what the algorithm consists of, nor is there any guidance provided to the inventor as to how to disclose the software algorithm in the patent. The absence of this discussion, which characterizes not just *WMS Gaming* but all subsequent Federal Circuit and lower court decisions applying *WMS Gaming*, is of tremendous significance.

As discussed above and again in later sections, software algorithms are a well-recognized means of expressing the logic of computer programs. However, there are many different techniques that can be used to disclose an algorithm, and each such technique presents an equal if not greater array of choices as to the level of detail provided. Absent guidance from the courts, the algorithm disclosed in a patent becomes a subjective reflection of the interests of the inventor as opposed to a legitimate standard that has application across case lines. *WMS Gaming* essentially teaches that the "disclosed algorithm" is the standard by which to compare different software programs, but the decision provides no guidance as to how such a comparison should proceed.

The Federal Circuit's determination in *WMS Gaming* of the disclosed algorithm in the Telnaes patent serves only to confuse an already uncertain situation. The Federal Circuit's decision did not comply with its own rule that the disclosed algorithm must be found in the the written description part of the specification as distinguished from the language of the claims.[29]

The Federal Circuit said that the structure corresponding to the "means-for-assigning" element was "a microprocessor programmed to perform the algorithm illustrated in Figure 6" of the Telnaes patent.[30] The court goes on to describe the algorithm allegedly disclosed in Figure 6 as (1) the number of single numbers exceeds the number of stop positions; (2) each single number is assigned to only one stop position; (3) each stop position is assigned at least one single number; and (4) at least one stop position is assigned more than one single number.

However, items 1 and 4 of the Federal Circuit's algorithm are virtually the same as the function expressed in the claim language. The insight is that the algorithm is properly derived from functional language notwithstanding directions to the contrary. At least one lower court decision, *Faroudja Laboratories, Inc. v. Dwin Electronics, Inc.*,[31] has interpreted *WMS Gaming* in this fashion. In *Faroudja,* the court said that *WMS Gaming* presents a special case in which the structure is altered by virtue of its programmable nature, such that a court "must . . . limit the structural element to its functional purpose by importing functional language into the structural specification."[32]

Moreover, there are aspects of the structure of Figure 6 of the Telnaes patent that were not read into the claim. Beginning at column 4, line 34 of the patent, Figure 6 is described as a diagrammatic form of representation of one standard type of reel where circle 46 illustrates 22 positions of numbers on the reel. The disclosure then states that "the table entry in the random number generator for the machine is illustrated by the circle 50." Such "table entry" of numbers assigned in the random number generator was not read into the claim. The Federal Circuit did not say that the algorithm included the step of providing a single number for each location on the reel (as shown by circle 46) or that additional single numbers were provided for each location on a table entry in the random number generator. The Federal Circuit merely indicated that such additional numbers were assigned but not how. The attempted but ultimately incomplete application of Figure 6 presents one more question regarding the appropriate way to find the disclosed algorithm. There exists a compelling basis for revisiting the nature and

purposes of software algorithms from the perspective of persons skilled in the art of writing software.

Misunderstanding of the Software Art

Courts will have to learn a better way to evaluate software patent specifications in order to describe and compare algorithms effectively. The disclosure of algorithms in a patent specification also must better serve the needs of those who mean to evaluate the specification for these purposes. Patentees, potential infringers, patent examiners, attorneys, judges, and even jurors need a method for communicating the algorithms that define the scope and detail of software patent claims that is simultaneously accurate, complete, and digestible. Unfortunately, the most common methods employed by patent applicants to document software algorithms satisfy little or none of these criteria. Fortunately, the software industry itself has already adopted a satisfactory method for specifying software algorithms in the form of its standardized UML.

The Inherent Structure of Software Algorithms

Persons of ordinary skill in the art of writing software recognize that all software algorithms have an inherent structure that, within reason, can be unambiguously specified.[33] Although software algorithms can be extremely complex, the nature of software has not changed since it was first invented more than 50 years ago. All software algorithms consist of (1) data that (2) under certain state conditions (3) are manipulated by functional processing.[34] Understanding each of these perspectives of an algorithm, first separately and then operating together, is necessary to complete one's understanding of its scope.

The data perspective discloses only the data elements manipulated by the algorithm and the navigational paths between sets of data elements necessary for the algorithm to traverse that data during particular functional processing. This perspective discloses only data at rest and is akin to viewing the data in a spreadsheet devoid of any underlying formulas that would modify those data values.

The state conditions of an algorithm disclose its most critical decision-making logic. This logic acts as an administrator for the algorithm and reacts to events by triggering one or more functional calculations or by modifying the internal state of the system to recognize either that additional functional processing is now permitted to occur if appropriately triggered or that additional functional processing is no longer permitted to occur while the system remains in its new state. For example, in an audit state, a spreadsheet may permit final column totals to be tallied, but manipulations of the original data transactions may not be permitted. Later, when a fiscal year has expired, a spreadsheet may not permit any changes at all to its content.

The functional perspective consists of the processing steps that access data and execute formulas, a more visible component of algorithms and too often the sole focus of software patent specifications.[35] The individual formulas attached to distinct cells in a spreadsheet are the functions that it executes when appropriately requested by the user. Some software systems impose additional requirements on the timing of the execution of functional processing, but the same three essential elements are always present.

Software patent litigation usually concerns the logical structure of an algorithm (*i.e.,* what it does), as opposed to its physical structure (*i.e.,* how it does it). For example, a logical structure may specify that 10 functional steps must be performed to manipulate three tables of data elements. By contrast, the physical structure results from modifying the logical structure to accommodate engineering constraints on memory space or processor speed. The physical structure may specify that the 10 processing steps are divided into two programming subroutines, each with five processing steps and executing in parallel, and that two of the three data tables have been combined into one larger table to improve data access time for the user. Although details of physical structure can help those skilled in the art understand how the patentee implemented the algorithm, such techniques are familiar to the average software engineer and therefore usually unnecessary to enable implementation. Further, instead of facilitating an understanding of what the algorithm is ultimately accomplishing, such physical details distract from this goal.

Another inherent characteristic of software algorithms is that all software functions are event driven.[36] That means that all functional processing within a software algorithm executes upon the occurrence of a defined event—the input into a computer of a user request for information, an internally recognized data condition (*e.g.,* exceeding a safety threshold value), or the expiration of a specified time period. Some events directly trigger functional processing to occur, while other events merely enable particular functional processing to occur when other events are later detected. Both types of events cause algorithms to transition through a series of distinct qualitative states. These state conditions are a critical component in the logical structure of complex software algorithms, and they provide two important benefits. First, complex algorithms are more easily digestible when its functions are evaluated

in the context of the state in which they execute. Second, events that trigger software execution are a convenient mechanism by which to associate a claimed function with its supporting structural components in the algorithm.

The Goals of Algorithmic Disclosure in Software Patent Specifications

To facilitate evaluating the scope of a software patent under *WMS Gaming*, a specification disclosing the algorithm must permit a thorough human understanding of the relevant logical structure of the software function being claimed. Therefore, to determine whether two algorithms are structurally identical or structurally equivalent, the algorithms must be disclosed in sufficient detail and from multiple perspectives.

Detail means precision in disclosure and facilitates an accurate comparison of algorithms where the issue resides. For example, if the distinction at issue between two algorithms concerns the scope of data they can process, the algorithmic disclosure must list and define the data elements processed by the algorithm and the relevant navigable paths for traversal among those data elements. Without such detail appropriately specifying the structure of the data upon which the algorithm executes, the resolution of an issue that turns on data scope becomes extremely difficult.

An algorithm must also be disclosed from all relevant data, state, and functional perspectives for a proper evaluation of its structure to proceed. A software patent claim dispute that concerns the scope of functional processing performed by the algorithm cannot be easily resolved if the patent specification discloses only a set of data tables purportedly manipulated by the claimed function. Similarly, a purely functional specification devoid of any detail concerning the data structure of the algorithm produces a necessarily skewed understanding of its scope. Like any complex engineered invention, software algorithms have multiple interconnected layers that must be understood first separately and then in unison to fully comprehend the scope and effect of their structure.

Finally, the human factor here is paramount. Algorithms are creatures of human thought, intended to represent what humans desire a computer processor to do. Algorithms exist distinct from the computer instructions—programming code—which represent what a computer is actually instructed to do using a machine-interpretable language. The difference is not merely one of form or level of detail. The critical difference is that an algorithm is a vehicle for human beings to communicate to one another the structure of a software function. Therefore, complexity must be managed for human consumption, algorithmic detail must be appropriately organized by its nature, symbology must be standardized, and a common grammar must be in operation. Just as structural complexity is universally communicated among architects and contractors via blueprints, software complexity requires its own appropriate communication vehicle if courts are to understand what software engineers intended their software programs to do.

Inadequate Methods of Disclosing Algorithms

The methods commonly used by patent applicants to describe algorithms in patent specifications can provide some context for the algorithm being evaluated. However, these disclosures do not provide the minimum necessary accurate, complete, and digestible disclosure of that algorithm needed by those who must consider the functional claim in dispute. Patent applicants use these methods separately and sometimes in concert. However, using them in concert does not resolve the inadequacies that each type possesses individually. The following examination of the most common methods demonstrates these inadequacies.

Source Code

Some software patent specifications disclose the programming source code used to implement the function being claimed as a method of disclosing the algorithmic structure. This method frustrates human understanding of the algorithm for a number of reasons. First, even the most "readable" source code is organized for the purpose of instructing a machine and does not lend itself to immediate comprehension even by those skilled in the art. Software engineers would sooner review a graphic model or textual summary of the algorithm embedded in source code than read the source code itself, unless the code needs to be debugged or modified. Moreover, because software engineers are usually conversant in only a few of the multitude of existing programming languages, disclosure of source code in any particular programming language is not necessarily understood by a majority of those skilled in the software art.

Second, source code by definition is the merger of both the logical and physical structure of the algorithm, which, as discussed above, impedes comprehension of the logical scope at issue in the algorithm. Third, source code merges the data, state condition, and functional perspectives of algorithms into one continuous form. Understanding complex algorithms begins with separating these views, a goal that source code by its nature prevents. A related practical consideration is that source code typically qualifies for protection against unauthorized copying or other use under the trade secret doctrine. However, trade secret status is contingent on

maintaining the confidentiality of the source code, which is obviously lost if the source code is disclosed in an issued patent.

Flow Charts

Flow charts as a method of algorithmic disclosure are an improvement over source code but still often frustrate human understanding. As with source code, flow charts in practice sometimes mix physical and logical structure together, but they are capable of representing only logical structure. The graphic symbology used on flow charts is generally straightforward to follow even by laypersons, and the symbols themselves have been mostly standardized throughout the software industry through decades of practice. Flow charts, however, present two problems. First, they disclose only the functional processing perspective of an algorithm, leaving the data and state condition perspectives to be inferred after frequent readings. Second, flow charts do not scale up easily, so once an algorithm starts to require multiple pages of flow charts to be comprehensive, one gets easily lost following the logic as they jump from page to page. Therefore, while flow charts can be accurate disclosures of functional processing, they are incomplete and lack digestibility.

Benghiat v. Itron[37] provides a compelling example of how the reliance on flow charts (both by the patent applicant and the court) may result in an erroneous interpretation of the software algorithm. The patent at issue in *Benghiat* disclosed more than 10 pages of flow charts encompassing not just the algorithm necessary to perform the function called out in the claim but also additional processing steps, such as the manner in which data was stored in memory following the execution of the algorithm. The flow charts also were complex and detailed, which, particularly as the reader moved from page to page, made it extremely difficult to follow the different algorithmic functions. The court in *Benghiat* nonetheless determined as part of the claim construction that the flow charts, *in their entirety*, disclosed the algorithm. The court effectively read all of the flow charts into the claim and thereby gave the claims an unduly narrow interpretation.[38] The court grounded this extreme position on little more than passing reference to a computer hornbook stating that algorithms "may involve the use of a flow chart."[39]

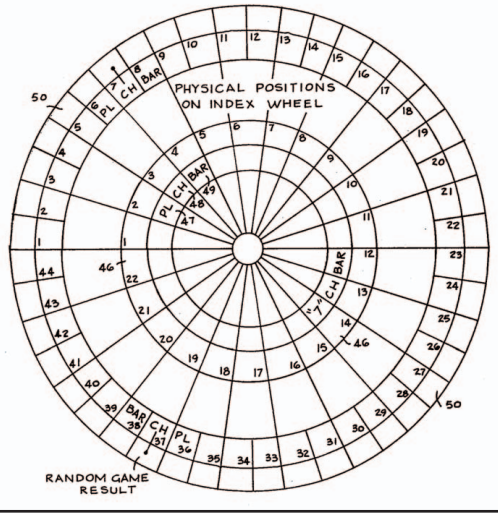
Hardware Schematics

Another type of diagram frequently included in software patent specifications is the hardware schematic, which graphically describes the hardware components that comprise the physical machine that is the subject of the patent. These diagrams include representations of the memory chips on which software and data will be stored, the microprocessors that will execute the software instructions, and the physical communication links over which data will travel into or out of the system. Although a hardware schematic can help understand the patentee's preferred embodiment of the machine and may also directly bear on patent claims beyond the means-plus-function claims at issue in regard to the algorithm, such a diagram does not disclose the algorithm's logical structure. A hardware schematic generally discloses only where an algorithm executes, not what it does upon execution.

Concept Diagrams

Concept diagrams are sometimes included to ease the reader into the patent subject matter or to convey a novel idea graphically. For example, Figure 1 is a reproduction of the concept diagram included in the Telnaes patent at issue in *WMS Gaming*.[40] It shows a wheel-like graphic used to convey the concept of virtual reel stop positions within the slot machine. This figure is not altogether unhelpful in understanding the algorithm; in fact, after some brief study, it does generally convey the novelty claimed in the patent. However, this kind of diagram is not used by those skilled in the art of software engineering, and it contains no formality in symbology or grammar that can ensure that it is interpreted as intended by the inventor. An example of this frailty is that this diagram is also inaccurate on its face. As a wheel with radially extending spokes from the center (physical reel stop positions) to the outer circle (vir-

Figure 1: Concept Diagram from Telnaes Patent



tual reel stop positions), the graphic implies that that the assignment of virtual reel stop positions to physical ones is limited by the particular increase of space available as the spoke moves farther from the center. The actual algorithm claimed in the specification was more flexible than that because it permitted anywhere from one to an unlimited number of virtual stop positions to be assigned to any one physical position.[41]

The Unified Modeling Language as the Industry Standard

The software industry has already resolved for itself how to represent algorithms accurately, completely, and in a digestible form. After decades of experimentation with various representational methods, UML was adopted in 1997 by the software industry as its standard method for communicating software algorithms among those skilled in the art.[42] UML has also served as a basis for non-software professionals to communicate with software engineers regarding the completeness and accuracy of algorithms under evaluation.

UML consists of a series of diagrams variously targeted toward each inherent perspective of algorithmic structure: data, state conditions, and functional processing. UML is based on the theory of object-oriented software development, which manages the inherent complexity of software systems by purposefully organizing and distributing algorithmic structures around the objects naturally found in the subject matter domain being automated by the system.[43] This type of algorithmic organization facilitates understanding because the most stable, broadest sweeping rules about the patent subject matter are examined first. State logic and functional processing are then more easily digested because they have been partitioned into smaller pieces and can be evaluated in context of the broader goals of the algorithm. Furthermore, because these data, state and functional views are partitioned, each one can be precisely specified without duplicating the information on another view. Complex systems may require more than one subject matter domain to be modeled, and each domain often has multiple objects, each with as much associated algorithmic structure necessary to fully describe its behavior.

UML diagrams are subject to both internal and external norms that prevent gross distortions of the algorithm from being asserted. Internal norms are derived from the definition of UML itself and include rules for proper use of graphic symbols, completeness of information, and consistency among diagrams that purport to represent interconnected perspectives of the same algorithm.[44] External norms include rules for determining whether data, state conditions, or functional processing are missing or are being incorrectly distributed among the objects on the diagram. Internal norms are analogous to English grammar rules, and external norms are analogous to guidelines for writing an effective journal article.

Figures 2 through 7 illustrate how UML would be used to communicate the slot machine algorithm disclosed by the Telnaes patent specification in *WMS Gaming*. Figure 2 shows the data perspective, revealing that the objects Lever, Reel, Bet, Player, Slot Machine, etc. are the basic objects about which the algorithm is concerned. Figure 2 also indicates, by their allocation, which object each data element best describes. Certain objects on this diagram have state conditions that trigger or enable functions within the algorithm. These state conditions and triggering events are distributed to the Reel, Bet, and Lever objects as shown on Figures 3, 4, and 5, respectively. Each state in these diagrams is further allocated a small portion of the functional processing steps of the gaming algorithm, as shown in Figure 6. Then, as shown in Figure 7, UML can also tie the algorithmic pieces together so that a start-to-finish execution path can be followed.

Of particular importance for means-plus-function claims in software patents is that UML can be used to disclose the algorithm of a claimed function regardless

Figure 2: Data Model for Telnaes Patent

(shaded box encloses algorithmic structure at issue)

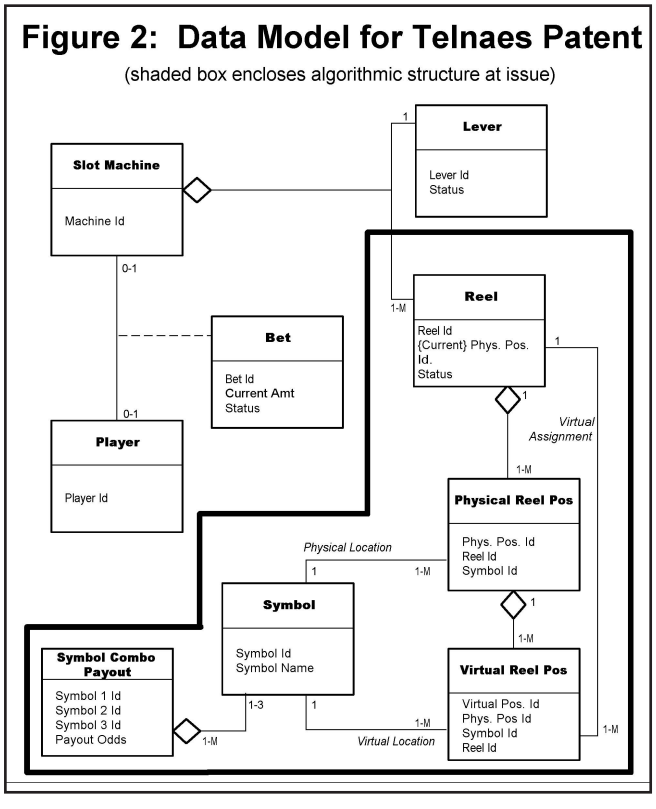


Figure 3: State Models for Reel Object (Telnaes Patent)

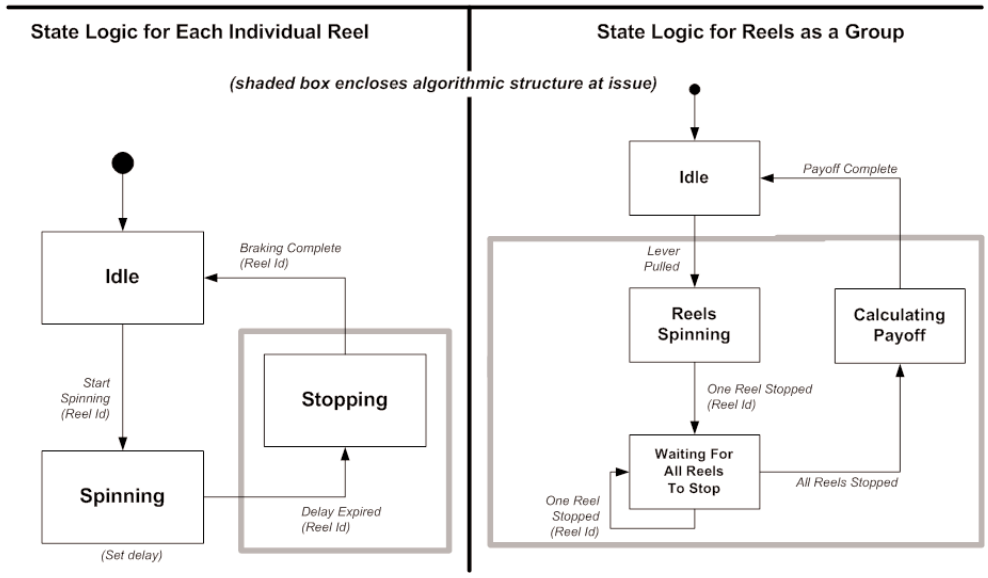


Figure 4: State Model for Bet Object (Telnaes Patent)

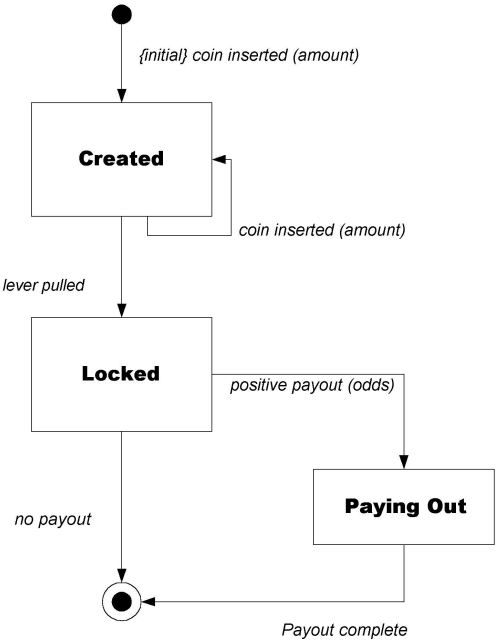


Figure 5: State Model for Lever Object (Telnaes Patent)

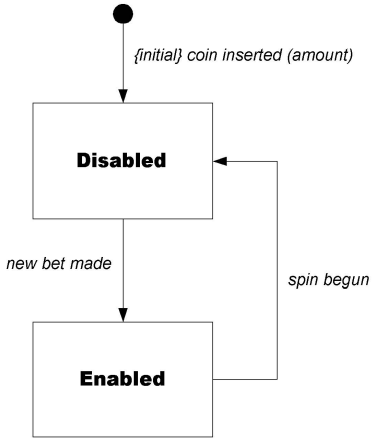


Figure 6: Reel Object Functional Processing:

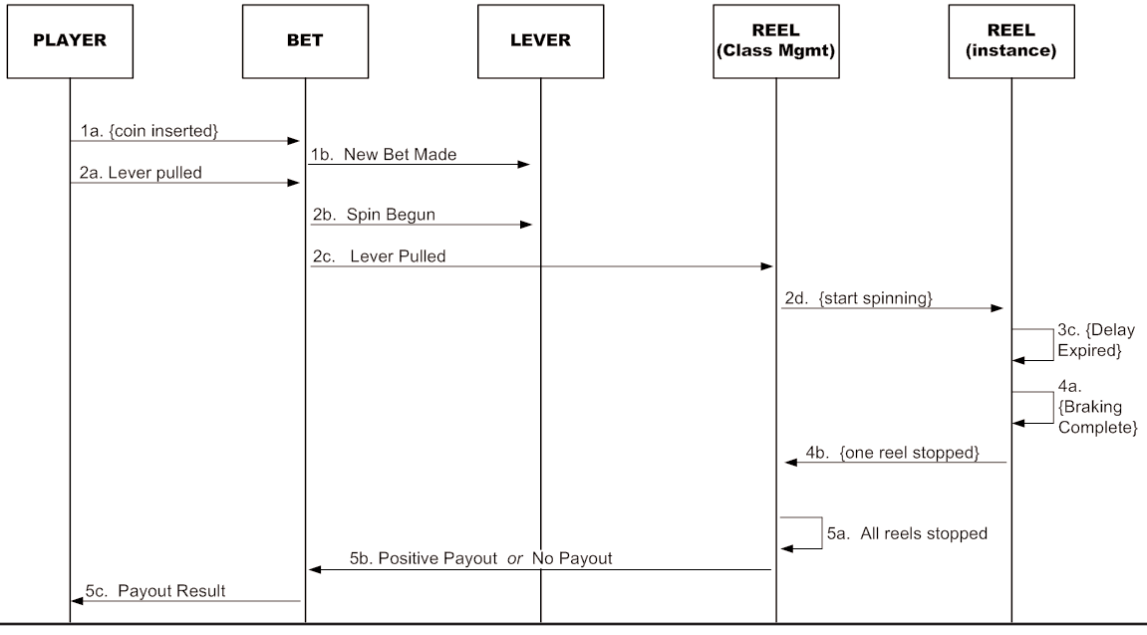
***Stopping* state**

1. Generate random # for Virtual Stop Position
2. Determine Physical Stop Position thatmaps to Virtual Stop Position
3. Stop Reel at Physical Stop Position
4. Trigger event One Reel Stopped (Reel Id = this Reel)

***Reels Spinning* state**

1. For each Reel instance, perform:
   - Start Spinning (Reel Id)

Figure 7: Execution Path Across Entire Algorithm (Telnaes Patent)



of whether the patentee originally used UML to develop the software and regardless of whether the actual source code is based on any object-oriented software development principles. Similarly, UML can be used in litigation forums to persuade fact-finders of an algorithm's limitations, even if the patent author did not use it. The scope of an algorithm, that is, what it does, can be represented in UML at any stage because an algorithm's inherent logical structure never changes. Of course, each patent owner must gauge how accurately and completely they wish to disclose their algorithm to maximize the protection they seek.

Application of UML Dictates a Different Result in *WMS Gaming*

The application of UML demonstrates significant differences between the disclosed algorithm in the Telnaes patent and the algorithm used in WMS 400, the accused product in *WMS Gaming*. These differences cannot be characterized as "insubstantial." There

was clear error in the trial court's finding that the claimed and accused algorithms were structural equivalents. The Federal Circuit in *WMS Gaming* therefore should not have affirmed the finding that WMS 400 infringed the Telnaes patent. This, in turn, suggests a compelling need for the courts to reevaluate how they determine and compare the algorithms in accused and claimed software or software-related products.

In Figure 8, UML is used to describe the data perspective of the accused device in *WMS Gaming*. The objects shown enclosed in the shaded box represent the portion of the accused device's algorithm at issue. In Figures 2 and 3, the Telnaes patent's disclosed algorithm is similarly highlighted. On first examination, the algorithms for the claimed and accused devices contain some similar basic elements, such as a Reel, its Physical Stop Positions, and that each combination of three symbols (*e.g.*, Cherry, Bar, Double-Bar) are associated with a Symbol Combo Payout for the player. Because these are the essential elements of all slot machine gambling, any related algorithm, no matter how diverse, would necessarily contain these objects. In addition, both algorithms are targeted at manipulating the Payout Odds without changing the physical Reels on the machine.

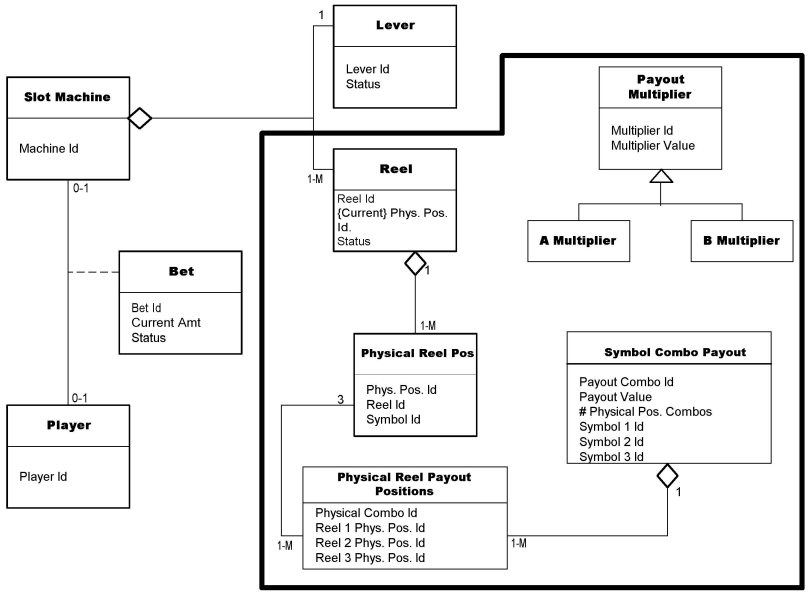
What is different about these algorithms is both the mechanism by which the Payout Odds are manipulated and whether the Reel positions drive the Payout Odds or vice versa. As shown in Figure 2, in the Telnaes patent, one or more Virtual Stop Positions are assigned to each Physical Stop Position before the slot machine is operated. Then, as shown on Figure 3, when the Player pulls the Lever, the Reels as a group are set in motion, but each reel *independently* chooses its own random Virtual Stop Position and then brakes at the corresponding Physical Stop Position. Once every Reel has stopped, the algorithm looks up the resulting symbol combination to determine the Payout Odds.

By contrast, a Virtual Stop Position is not employed by the algorithm in the accused device. Instead, as shown in Figure 8, the accused device starts by using two randomly selected Multipliers to determine the Payout Odds and the corresponding symbol combination to show the Player.[45] Once the desired symbol combination is determined, one of the potentially many combinations of Physical Stop Positions that would display that symbol combination is chosen. Finally, the Reels are commanded *as a group* to brake at the chosen set of Physical Stop Positions. The UML representations of these algorithms demonstrate that:

1. Where the Telnaes patent employed a virtual expansion of the stop positions on a slot machine reel, the accused device contained no such algorithmic structure.

2. Where the Telnaes patent randomly selected Virtual Stop Positions, the accused device randomly selected the Payout Odds (via dual multipliers) and then randomly selected (if more than one possible) the Physical Stop Positions on the Reel.

Figure 8: Data Model for Accused Device (WMS 400)
(shaded box encloses algorithmic structure at issue)



3. Where the Telnaes patent let the Reels spin independently and then derived the Payout Odds from the result, the accused device determined the Payout Odds first and then stopped the Reels accordingly.

In *WMS Gaming*, the district court found that the algorithm implemented in the accused device was the equivalent of the algorithm disclosed in the Telnaes patent.[46] The district court adopted expert testimony that the accused device's assignment of a "combination of numbers" (*i.e.*, the two random multiplier selections and the one random choice of Physical Stop Position combinations) to a Physical Stop Position was not substantially different than the Telnaes patent's assignment of a single number (each Virtual Stop Position) to a Physical Stop Position.[47] Therefore, the court determined, the algorithmic structures in the two devices were equivalent.[48]

The district court's finding, however, was based on the erroneous assumption that the accused device was performing assignments of "combinations of numbers." As the UML data model for the accused device shows, there was no pre-assignment of the *individual* Multipliers to any other data element. Furthermore, the final set of Physical Stop Positions, if more than one set was possible to display the desired symbol combination, was *randomly selected*, not pre-assigned. If a pre-assignment had existed, a random selection would not be necessary.

The district court also erred when it accepted the expert's characterization of the issue as concerning *how many* numbers were being assigned. The more integral question should have been *what* numbers, if any, were being assigned, *when* were those assignments occurring, and *how* did those assignments affect the course of each algorithm's execution. Most arguably, the accused device had a substantially different algorithmic structure and therefore did not infringe the Telnaes patent under the doctrine of equivalents. The district court may have done the best it could under the circumstances. Unfortunately, without a method for evaluating the algorithms at issue that was accurate, complete, and digestible, the district court's ability to discern the critical differences was severely handicapped. The end result was clear error in the district court's finding of structural equivalents. The Federal Circuit's decision to affirm the district court's finding of infringement cannot be reconciled with this conclusion.

Conclusion

Existing software patents continue to mature and new software patents continue to multiply. Absent modification of rules announced in *WMS Gaming* regarding application of the disclosed algorithm, the mistakes of algorithmic interpretation that have occurred in *WMS Gaming* as well as in other decisions interpreting computer implemented means-plus-function claim limitations are likely to re-occur. Courts should require the application of UML to determine the disclosed algorithm.

Notes

1. WMS Gaming, Inc. v. International Game Technology, 184 F.3d 1339 (Fed. Cir. 1999).
2. *Id*. at 1348-1349.
3. *See* Itron v. Benghiat, 169 F. Supp. 2d 1073 (D. Minn. 2001); GTE Wireless, Inc. v. Qualcomm, Inc., 188 F. Supp. 2d 1201 (S.D. Cal. 2002); ABB Automation, Inc. v. Schlumberger Resource, 2003 WL 1700013 (D. Del. 2003).
4. *See* William M. Atkinson and John A. Wasleff, "The Devil Is In The Details: Patent Protection For Software Driven Inventions After *WMS Gaming*," 17 No. 9 *CILW* 6, 8 (Sept. 2000) (Atkinson, Devil is in the Details) ("What the *WMS Gaming* decision doesn't do, however, is provide guidance concerning the degree of difference between two software programs necessary to avoid infringement. This decision . . . is left for future cases.").
5. 35 U.S.C. § 112, ¶ 6.
6. *Id*. *See* Valmont Indus, Inc. v. Reinke Mfg. Co., 938 F.2d 1039, 1042 (Fed. Cir. 1993).
7. Chiuminatta Concrete Concepts, Inc. v. Cardinal Indus. Inc., 145 F.3d 1303, 1308 (Fed. Cir. 1998).
8. *See id*.
9. *See id*.
10. *See id,* quoting 35 U.S.C. § 112, ¶ 6.
11. *Valmont Indus. Inc.,* 983 F.2d at 1042.
12. *Chiuminatta*, 145 F.3d at 1309.
13. *See* Odetics, Inc. v. Storage Tech Corp., 185 F.3d 1259, 1268 (Fed. Cir. 1999); *Chiuminatta,* 145 F.3d at 1309.
14. *See* Al-Site v. VSI Int'l, Inc., 174 F.3d 1308, 1320-1321 (Fed. Cir. 1999).
15. *See* Gottschalk v. Benson, 409 U.S. 63 (1972) (The US Supreme Court held that *mathematical* algorithms (not just formulae) were non-patentable subject matter, effectively blocking the patenting of pure software and forcing patent applicants to shift their focus to patenting mechanical devices and processes that happened to include computer programs.). *See also*, Julie E. Cohen and Mark A. Lemley, "Patent Scope and Innovation in the Software Industry," 89 *Cal. L. Rev.* 1, 8 (Jan. 2001) (Cohen, *Patent Scope*); Bruce Abrahamson, "Promoting Innovation in the Software Industry: A First Principles Approach to Intellectual Property Reform," 8 *B.U.J. Sci.&Tech. L.* 75, 81 (Winter 2002).
16. In re Alapatt, 33 F.3d 1526, 1545 (Fed. Cir. 1994) (*en banc*).
17. *Id*. at 1545.
18. Cohen, *Patent Scope*, at 10.

19. Atkinson, *The Devil is in the Details*, at 7.
20. Bradley D. Baugh, "WMS Gaming, Inc. v. International Game Technology" 15 *Berkeley Tech. L.J.* 109, 114 (2000), quoting Mark D. Janis, "Who's Afraid of Functional Claims? Reforming the Patent Laws § 112, ¶ 6 Jurisprudence," 15 *Computer & High Tech. L.J.* 231, 235 (1999) ("Claims drafted in means-plus-function [language] are especially prevalent in patents on software-related inventions, where the format has been thought useful for complying with the subject matter eligibility requirement.").
21. *See, e.g.*, State Street Bank & Trust v. Signature Financial Group, 149 F.3d 1368, 1374 (Fed. Cir. 1988), *cert. denied*, 525 U.S. 1093 (1999) (jettisoned the physicality requirement in favor of rule that business methods are patentable so long as they produce a "useful, concrete and tangible result").
22. More specifically, as to infringement, the Federal Circuit reversed the holding of literal infringement because the function of the accused device was not the same as the function of the claimed device in all respects, *see* 184 F.3d at 1352, but nonetheless affirmed the finding of infringement under the doctrine of equivalents based on the further finding that the differences between accused and claimed devices were insubstantial, *see id.* at 1353-1354.
23. Baugh, *supra* n.20, at 117, which contains an excellent presentation on the facts and technology at issue in *WMS Gaming*.
24. *WMS Gaming*, 184 F.3d at 1349.
25. *See supra*, n.3.
26. Allen Newell, "Response: The Models are Broken," 47 *U. Pitt. L. Rev.* 1023, 1028 (Summer 1986).
27. *Id.*
28. Bruce Abrahamson, "Promoting Innovation in the Software Industry: A First Principles Approach to Intellectual Property Reform," 8 *B.U.J. Sci.&Tech. L.* 75, 87 (Winter 2002).
29. *See* GTE Wireless, Inc. v. Qualcomm, Inc., 188 F. Supp. 2d 1201, 1210 (S.D. Cal. 2002) ("The Federal Circuit [in *WMS Gaming*] concluded the corresponding structure was the algorithm disclosed in the written description portion of the specification, not the claims.").
30. *WMS Gaming*, 184 F.3d at 1349.
31. Faroudja Laboratories, Inc. v. Dwin Electronics, Inc., 76 F. Supp. 2d 999, 1010 (N.D. Cal. 1999).
32. *Id.*
33. *See generally* Sally Shlaer & Stephen J. Mellor, "Object Lifecycles: Modeling the World in States" (1991).
34. *See generally id.*
35. *See, e.g.*, US Patent No. 5,193,056, figs. 5-11 & cols. 7-13 (issued Mar. 9, 1993) (at issue in *State Street*, 149 F.3d 1368); US Patent No. 4,448,419, cols. 3-4. (issued May 15, 1984) (Telnaes patent) (at issue in *WMS Gaming*, 184 F.3d 1339).
36. *See generally* Shlaer & Mellor, *supra* n.33.
37. Benghiat v. Itron, 169 F. Supp. 2d 1073. *See supra*, n.3.
38. *Id.*
39. *Id.*
40. *See* Telnaes patent, fig. 6.
41. *See WMS Gaming*, 184 F.3d at 1349.
42. See Cris Kobryn, "UML 2001: A Standardization Odyssey," Communications of the ACM at 31 (Oct. 1999).
43. *See id.* at 37.
44. *See id.* at 33-34.
45. The order of functional execution in the accused device is described by the court in *WMS Gaming*. *See WMS Gaming*, 184 F.3d at 1344.
46. *See id.* at 1354.
47. *See id.* at 1351.
48. *See id.* at 1354.