United States District Court,
N.D. California.

Martin Gardner REIFFIN,
Plaintiff.
v.
MICROSOFT CORPORATION,
Defendant.

No. C-98-0266 VRW

April 30, 2002.

Martin Gardner Reiffin, Danville, CA, pro se.

Norman H. Beamer, Ropes & Gray, Palo Alto, CA, Jesse J. Jenner, Ropes & Gray LLP, New York, NY, Frank L. Bernstein, Sughrue Mion Zinn Macpeak & Seas, PLLC, Menlo Park, CA, for Plaintiff.

Eric L. Wesenberg, Erin Farrell, William L. Anthony, Orrick Herrington & Sutcliffe LLP, Menlo Park, CA, John D. Vandenberg, Todd M. Siegel, J. Christopher Carraway, Joseph T. Jakubek, Klarquist, Sparkman, Campbell, Leigh & Whinston, LLP, Portland, OR, Thomas W. Burt, Microsoft Corporation, Redmond, WA, Terrence Patrick McMahon, McDermott, Will & Emery, Palo Alto, CA, for Defendant.

ORDER

VAUGHN R. WALKER**, District Judge.**

On January 23, 2002, the court conducted a claim construction hearing. Based on the parties' arguments at this hearing and their submissions to the court, the court issues the following order.

A

In 1982, plaintiff filed a patent application entitled "Computer System with Real-Time Compilation." This application purported to disclose plaintiff's invention of "a novel computer architecture providing real-time compilation of a high-level language source program concurrently as the program is entered or edited at the console by the programmer." 1982 App (Doc # 311, Exh # 3) at 1. As explained in the application, plaintiff's invention was an attempt to solve problems related to "programming in high-level language." Specifically, plaintiff noted that even as the computer market was undergoing rapid expansion, programming in high-level language remained "slow, tedious and inefficient." *Id* at 2.

Computer programming in high-level language requires a program that analyzes and translates what the programmer enters because although humans write programs in a programming language (source code), computers execute only in machine language (object code). When programming is done in high-level

programming language, the original program must be analyzed by another program called a "compiler," which translates the programming language into object code. *Id* at 1. The reason for the inefficiency of high-level language computer programming at the time of plaintiff's 1982 application was that computers were generally only capable of handling one step of the programming process of at a time. As a result,

even under the optimum conditions of an interactive console, a compiled language requires a repeated sequence of steps comprising loading the editor, writing or editing the source code, loading the compiler, executing the compiler, loading the linker, executing the linker, running the program, and repeating the sequence when an error is indicated during compilation of the source code or execution of the object code. During much of the time the programmer is compelled to wait for completion of the loading or execution steps, and this waiting time is both wasteful and boring.

Id at 2.

Plaintiff's claimed invention was a method of combining writing or editing the source code with the execution of the compiler, so that compilation of the source code could be "performed in real-time as the source code is entered or edited by the programmer." *Id*. Plaintiff's system for accomplishing this goal, as described in his specification, utilized an interrupt mode of operation, whereby the computer's central processing unit (CPU) would execute the compiler and editor seamlessly as viewed by the computer user.

As described by plaintiff, in plaintiff's system the compiler executes continuously, until interrupted, performing analysis of the program source code stored in a source code buffer in the computer's memory and outputting compiled object code into an object buffer. Whenever the computer user strikes a key on the keyboard, the CPU executes an interrupt sequence, which causes the compiler's execution to pause while the editor processes the information required by the keystroke. After the editor finishes processing the keystroke, the interrupt service routine executes a return instruction and the compiler continues to execute. Because the compilation process is considerably faster than manual typing of source code, the user perceives the computer to be editing and compiling source code at the same time. *Id* at 18.

Plaintiff also noted in his discussion of his preferred embodiment that the interrupt sequence could be coordinated to a "timer or clock instead of a keyboard." *Id* at 20. In this modification, the compiler would be periodically interrupted and the input port polled to determine if a key has been struck. *Id*. If no key had been struck, interrupt would be terminated and control returned to the compiler. If a key had been struck, the editor would take control and execute the keystroke. Plaintiff noted that if this process occurred at intervals of 10 to 30 milliseconds, the user would perceive the editor and compiler to be working simultaneously. *Id* at 21.

Plaintiff filed a continuation of the 1982 application in 1985. No patents were issued on the basis of these applications, however, and they were eventually abandoned.

In 1990, plaintiff filed another application, which he deemed to be a continuation of his original 1982 application, also entitled "Computer System with Real-Time Code Processing." Doc # 311, Exh # 10. This application disclosed an invention very similar to the invention disclosed in plaintiff's 1982 application. In this application, however, plaintiff stated that his invention had applications beyond the context of programming in high-level source code. Plaintiff claimed as his invention a computer process that could provide real-time processing of alphanumeric code *and* natural languages. Besides its application to programming language, plaintiff's claimed invention could perform lexical and syntactic analysis (compile)

of "a natural language such as English" at the same time the natural language was being entered by the computer user. Plaintiff now claimed, as the scope of his invention, "[a] computer system for contemporaneous entry and analysis of character codes constituting a language having rules." *Id* at A0323.

The 1990 application issued as Patent No 5,694,603 on December 2, 1997. Patent No 5,694,604, filed in 1994 as a continuation of the 1990 application, also issued on December 2, 1997. The '603 and '604 patents were amended multiple times during their lengthy prosecution, which included several appeals to the Board of Patent Appeals and Interferences. As issued, the patents describe the invention of a "multithreaded computer system" to accomplish the concurrent execution of a plurality of instruction threads. The two patents include the identical illustrative embodiment originally described in plaintiff's 1982 patent application. The patents differ in their claims. The '603 patent claims a memory product storing multithreaded software and the '604 patent claims a method of multithreaded operation and a multithreaded system.

II

Plaintiff sued Microsoft in 1998 claiming that Microsoft infringed the '603 and '604 patents. Plaintiff alleges that several of Microsoft's software applications infringe his patents, including word processing programs that check spelling and grammar as text is entered and operating systems, such as Windows 98, that control the switching of the program threads active during operation of a personal computer. Plaintiff essentially argues that he invented multithreading, which is, most generally, the execution of a plurality of threads within a single process.

Unsurprisingly, Microsoft provides a much different account of plaintiff's alleged inventions. Microsoft argues that plaintiff's 1982 application disclosed an invention, which Microsoft refers to as a "four-part-editor-compiler" or a "source code compiler," with a narrow application to the particular problem faced by programmers of high-level language code. Microsoft further notes that plaintiff's 1990 application did not mention multithreading but only broadened the approach of plaintiff's original four-part-editor-compiler to accommodate its application to natural languages. In the early 1990s, however, Microsoft alleges that plaintiff first learned about "multithreading," when he read of its use in the OS/2 operating system. Presumably noticing the similarities between the invention discussed in his patent applications and multithreading, Microsoft alleges that plaintiff began stealthily to insert the language of multithreading into his applications in order to "attempt to patent [a] technique developed by others." Def Stat (Doc # 279) at 4. In order to succeed in his attempt to patent technology that was not his own, plaintiff needed the benefit of his 1982 filing date and, therefore, Microsoft alleges, plaintiff included the identical drawings and descriptions of his four-part-editor-compiler from his 1982 application in all his subsequent patent applications. Although he kept the embodiment and description the same, plaintiff amended the summary sections of his patent, asserts Microsoft, in order to claim the invention of a form of multithreading. Moreover, in attempting to gain the benefit of the filing date of his 1982 application, Microsoft contends that plaintiff repeatedly told the patent office that the disclosures of all his patent applications were identical. Although plaintiff's patents were not granted the benefit of the filing date of his 1982 patent application, plaintiff successfully convinced the patent office to disregard prior art after 1982.

In light of this account, Microsoft urges two alternate theories of claim construction. First, which Microsoft states is its "preferred" position, Microsoft argues that the court should limit the claims of the disputed patents to the scope of the invention disclosed in plaintiff's 1982 application and disregard any "new matter" introduced into plaintiff's patents by amendment. By its "preferred" position, therefore, Microsoft asks the

court to interpret two patents, which both claim an invention of a form of multithreading, as having nothing to do with multithreading. Rather, all the patents' claims would be interpreted to claim the invention of the four-part-editor-compiler. Microsoft's proposed claim construction based on this theory is set forth in exhibit C of its claim construction statement. Doc # 279, Exh C. If the court does not choose to limit the claims in this broad manner, Microsoft alternately argues, in its "less preferred" position, that the court should limit plaintiff's claims in light of statements made during prosecution of the patents and in light of the patents' preferred embodiment. This proposed construction is set forth in exhibit B. Doc # 279, Exh B.

In support of its theory that the court should limit the patents' claims to the invention disclosed in plaintiff's 1982 application, Microsoft advances several proposed principles of claim construction. Microsoft contends, for example, that because plaintiff told the patent office that the disclosures of his later applications were identical to those in his 1982 application, the court should construe the claims in the issued patents to cover no more than the four-part-editor-compiler disclosed in the 1982 application. Essentially, by this argument Microsoft asks the court to apply the doctrine of judicial estoppel, limiting the claims because of statements made to the patent office. Microsoft also contends that the court should construe the claims to cover no more than the embodiment described in plaintiff's patents, which is identical to the four-part-editor-compiler described in the 1982 application. Microsoft further asserts that the court should disregard any "new matter" added to plaintiff's patents after his 1990 and 1994 applications for the purposes of claim construction. Alternately, Microsoft argues that the simultaneous existence of elements of multithreading and the four-part-editor-compiler in the same patents renders the patents invalid for inconsistency and/or vagueness.

Although Microsoft's preferred claim construction position has some force, due in large part to the unusual prosecution history of the patents in dispute, Microsoft is essentially asking the court to interpret claim terms by reference primarily to considerations that do not focus on the claim terms themselves. Indeed, in its preferred position, Microsoft's proposed theories of claim construction ask the court to construct the claim terms in a certain way, irrespective of what the claims themselves actually say. Before proceeding to consider the claim terms themselves, therefore, the court will review the law applicable to claim construction.

B

The goal of claim construction is "to interpret what the patentee meant by a particular term or phrase in a claim." Renishaw PLC v. Marposs Societa' per Azioni, 158 F.3d 1243, 1249 (Fed.Cir.1998). In determining what a patentee meant by a term or phrase, the court looks first to the claim itself.

The claims of the patent provide the concise formal definition of the invention. They are the numbered paragraphs which "particularly [point] out and distinctly [claim] the subject matter which the applicant regards as his invention." 35 USC s. 112. It is to these wordings that one must look to determine whether there has been infringement. Courts can neither broaden nor narrow the claims to give the patentee something different than what he has set forth. No matter how great the temptations of fairness or policy making, courts do not rework claims. They only interpret them.

EI Du Pont de Nemours & co. v. Phillips Petroleum Co., 849 F.2d 1430, 1433 (Fed.Cir.1988).

"Absent a special and particular definition created by the patent applicant, terms in a claim are to be given their ordinary and accustomed meaning." York Prods., Inc. v. Central Tractor Farm & Family Ctr., 99 F.3d 1568, 1572 (Fed.Cir.1996). In order to determine whether a term has been used with a special or particular

meaning, the court should review the specification. See Vitronics Corp. v. Conceptronic, Inc., 90 F.3d 1576, 1582 (Fed.Cir.1996) (noting that "it is always necessary to review the specification to determine whether the inventor has used any terms in a manner inconsistent with their ordinary meaning.").

Besides the claims themselves, the specification includes a written description of the invention, which must be sufficiently clear and complete to enable those of ordinary skill in the art to make and use the invention. See *id*. As part of this description, the inventor must set forth the "best mode contemplated by the inventor of carrying out his invention." 35 USC s. 112.

In construing claim terms, the court must consider whether the description and the best mode, or the embodiment, of the invention may assist in determining what the patentee meant by a claim term. Using the specification, of which the claims are one part, in order to interpret what the patentee meant by a word or phrase in a claim, however, "is not to be confused with adding an extraneous limitation appearing in the specification, which is improper." EI Du Pont, 849 F.2d at 1433. By "extraneous," the Federal Circuit means "a limitation read into a claim from the specification wholly apart from any need to interpret what the patentee meant by particular words or phrases in the claim." *Id*.

The court may also consider the prosecution history of the patent, if in evidence, in order to determine the meaning of claim terms. Vitronics, 90 F.3d at 1582. Any interpretation of a claim term that is provided or disavowed during the prosecution of the patent informs the scope of the claim. See Renishaw, 158 F.3d at 1249 n3. The rule that the court does not impose extraneous limitations on claim terms from language in the specifications, however, is also "applicable to the impropriety of injecting into claims limitations from the prosecution history." Intervet America, Inc. v. Kee-Vet Laboratories, Inc., 887 F.2d 1050, 1053 (Fed.Cir.1989). "Ambiguity, undue breadth, vagueness, and triviality are matters which go to claim *validity* for failure to comply with 35 USC s. 112-para. 2, not to interpretation or construction." *Id,* emphasis in original.

These principles of claim construction diverge in certain key respects from several of Microsoft's proposed methods of claim construction, as outlined above. Microsoft, for example, contends that the court should "limit the patents" to the four-part-compiler-editor invention that plaintiff disclosed in his original 1982 patent application. See Def Br (Doc # 310) at 2. Microsoft argues that the court should limit the claims of the issued patents in this manner because the '603 and '604 patents contain the same description of the illustrative embodiment that plaintiff claimed was the subject of his 1982 patent application. Microsoft, therefore, asks the court to adopt the rule that the claims of a patent cannot exceed or, indeed, deviate from the embodiment disclosed therein. In this respect, Microsoft is confusing claim construction with enablement analysis. Absent indication that the patentee so intended, the court should not limit the meaning of claim terms to what is enabled by the specification. See Affymetrix, Inc. v. Hyseq, Inc., 132 F Supp 2d 1212, 1219 (N.D.Cal.2001).

In support of its claim that the court should limit the claims in the issued patents to the invention disclosed in plaintiff's 1982 application, Microsoft attempts to convert a truism, that a patent defines the scope of the invention, into a novel principle of claim construction, that claims must be limited to the scope of the embodiment. This position is made plausible by plaintiff's transformation of the invention disclosed in his 1982 application, the four-part-editor-compiler, into the preferred embodiment of the invention, preemptive multithreading, claimed in his patents. Nevertheless, the court must reject Microsoft's effort. As the Federal Circuit has already held in this case, "the relevant specifications are those of the '603 and '604 patents; earlier specifications are relevant only when the benefit of an earlier filing date is sought under 35 USC s.

120." Reiffin v. Microsoft Corp., 214 F.3d 1342, 1345 (Fed.Cir.2000). In the '603 and the '604 patents, the four-part-editor-compiler is but an illustrative embodiment of plaintiff's alleged invention. And as stated in Netword, LLC v. Centrall Corp., 242 F.3d 1347 (Fed.Cir.2001), a case cited by Microsoft for the proposition that the court should limit the patents' claims to their embodiment, "the specification need not present every embodiment or permutation of the invention and the claims are not limited to the preferred embodiment of the invention." Id at 1352.

Microsoft also argues that plaintiff's statements to the patent office in support of his opinion that he is entitled to the benefit of the filing date of his 1982 application should function to limit the patents' claims to the scope of the invention claimed in plaintiff's 1982 application. Microsoft, therefore, seeks to limit the claims of the issued patents to the scope of an abandoned patent application, based on plaintiff's statement that the disclosures in his later applications are identical to the disclosures in his 1982 application. Again, Microsoft seeks to blend fact-laden infringement issues, here those relating to priority date under 35 USC s. 120, with the construction of claim terms, which is a matter of law. Moreover, in nearly all of the statements to the patent office cited by Microsoft, plaintiff explicitly stated that his applications disclose not the identical *invention,* but the identical "detailed description and drawings constituting the disclosure." Pet (Doc # 311, Ex # 21) at A0563. Such statements were completely accurate.

Finally, the court rejects Microsoft's assertion that the court should disregard "new matter" for the purposes of claim construction. Section 132 of the Patent Act provides: "No amendment shall introduce new matter into the disclosure of the invention." 35 USC s. 132. In order to determine if new matter has been introduced, courts consider "whether the material added by amendment was inherently contained in the original application." Schering Corp. v. Amgen Inc., 222 F.3d 1347, 1352 (Fed.Cir.2000). Courts do not make a determination whether a patent contains new matter at the claim construction stage. "[T]he new matter prohibition is closely related to the adequate disclosure requirements of 35 USC s. 112." *Id*.

In support of its claim that the court should make a new matter determination and disregard anything determined to be new matter at the claim construction stage, Microsoft relies heavily on a Court of Claims case from 1970, Dresser Indus., Inc. v. United States, 193 Ct.Cl. 140, 432 F.2d 787 (Ct.Cl.1970), decided long before the Supreme Court made clear that claim construction is a matter of law, to be decided exclusively by the court. See Markman v. Westview Instruments, Inc., 517 U.S. 370, 387, 116 S.Ct. 1384, 134 L.Ed.2d 577 (1996). As its only post- *Markman* support, Microsoft relies on the Federal Circuit's decision in *Schering Corp*. Contrary to Microsoft's assertion, however, in *Schering Corp* the Federal Circuit limited the meaning of the term in dispute because the patentee had expressly limited its meaning during prosecution, not because the court determined it was new matter. See Affymetrix, 132 F Supp 2d at 1219.

Considering the practical effect of Microsoft's claim that the court should disregard new matter reinforces the impropriety of adopting it. Microsoft contends that as plaintiff's 1990 application did not mention multithreading, the court should disregard its discussion of multithreading as new matter; yet the claims of the patent that issued from the 1990 application, the '603 patent, are full of references to multithreading. In fact, Microsoft argues that the claims of the '603 patent themselves are new matter. See Supp App (Doc # X) at A2023. If the court adopted Microsoft's preferred position, therefore, for the purposes of claim construction, the court would disregard the language of the claims as issued. This the court cannot do.

Absent Microsoft's contention, which the court rejects, that the claims of the issued patents should be limited to the invention disclosed in plaintiff's 1982 application, the parties' divergence on the meaning of the claim terms is less dramatic. Plaintiff repeatedly asserts that all claim terms are meant in their ordinary

sense and that he is willing to have them defined with recourse to Microsoft's own technical dictionaries. Microsoft contends that plaintiff's "ordinary" definitions are not sufficiently "all-inclusive." Microsoft also argues that plaintiff's claim terms must be defined in a way that takes into account plaintiff's statements to the patent office distinguishing a different form of multithreading in an attempt to avoid prior art.

Specifically, Microsoft advances an alternative to its broad theory discussed above. Based on the patents' specification and prosecution history, this argument contends that the multithreading described in plaintiff's patents involves a "periodic preemption 'traffic light' mode of operation," as opposed to a "priority-based preemption 'traffic cop' type of multithreading." Def Br (Doc # 310) at 10. In a traffic light mode of multithreading, threads are interrupted periodically, without account for the priority level of the thread. In a traffic cop mode of multithreading, "[t]he relative priority levels of waiting and executing threads determines the allocation of CPU time to the individual threads." Id at 12. Microsoft argues that in distinguishing prior art, plaintiff limited his form of multithreading to a method based on periodic preemption. For example, in distinguishing the prior art Cheriton reference, plaintiff stated that:

Preemption of the active process in Cheriton is priority driven * * * preemption of the active thread in appellant's recited claims is driven only by a clock-activated interrupt.

Doc # 311, Exh # 22 at A586.

In construing the claims, the court will take into account plaintiff's statements to the patent office during the patents' prosecution. The court will also look to elements of the specification, besides the claims themselves, in order to determine if plaintiff has used terms or phrases in the claims in a particular way.

Finally, before turning to the individual claims, the court notes that the patents in suit involve a large number of claims. The <603 patent has 41 claims and the <604 patent has 36. Despite years of litigation the parties have not identified, or even narrowed, the particular claims that are in dispute. The terms in dispute, however, are common for most claims and, therefore, the court will concentrate on the central terms in dispute, as opposed to considering each claim on an element-by-element basis. As a result, the court may not fully resolve the construction of all claims relevant to infringement in this order. In fact, the court will exercise caution in the number of claims construed in this order, as construction issues may crystallize as infringement issues develop. Plaintiff has noted that he expects to narrow the number of asserted claims after infringement discovery. Pl Rep Br (Doc # 331) at 4 *. The court, therefore, may need to revisit claim construction as the litigation progresses.

III

A

MULTITHREADING

Plaintiff contends that "multithreading" should be given its common and ordinary meaning, within the field of computer programming. Citing a Microsoft publication, plaintiff asserts that "multithreading" means: "Supporting more than one thread within a single process." Pl Br (Doc # 331) at 18.

Plaintiff explicitly defined "multithreading" within both patents, although somewhat differently in each. The

'603 patent reads:

The term "multithreading" is used in this application in its ordinary generally understood sense to mean the concurrent time-sliced preemptive execution of a plurality of threads of instructions located within the same single operator-selected application program, whereby during execution of the program each thread may have at various times direct access to the same program address space, and with at least one thread invoked by a periodic clock-activated interrupt service routine which upon each activation asynchronously and preemptively takes control of the central processing means away from an executing thread at a repetition rate sufficiently fast so that even where the system contains only a single central processor the concurrent threads appear to execute effectively simultaneously and are so perceived by the user.

'603:1:24-37.

The '604 patent reads:

The term "multithreading" is used in this specification in its ordinary generally understood sense to mean the concurrent asynchronous preemptive time-sliced execution of a plurality of threads of instructions located within the same software program, controlled by a clock or timer which periodically activates the interrupt operation of the central processor.

'604:1:27-32.

An explicit definition of a claim term within the specification is given considerable weight in claim construction. See e g, EI Du Pont, 849 F.2d at 1433. The difference in the phrasing between plaintiff's definitions of multithreading in the two patents creates an ambiguity. Specifically, although plaintiff has provided an explicit definition of a claim term in each patent, it is not immediately apparent whether or not plaintiff intended to work with different definitions of multithreading in the two patents. It would be permissible for plaintiff to employ two different definitions of multithreading, but his statement in each patent that he is using multithreading "in its ordinary generally understood sense" suggests that plaintiff intended multithreading to have but a single meaning. In the '603 patent, multithreading is defined to include a periodic clock-activated interrupt service routine, but is apparently not limited to periodic preemption, as this definition implies that not all threads must be controlled by a periodic clock interrupt service routine. The definition in the '604 patent, however, states that preemption is "controlled" by a clock or timer, which activates the interrupt operation "periodically."

Looking at plaintiff's description of the illustrative embodiment, which is the same for both patents, does not resolve this ambiguity. Plaintiff's embodiment, as discussed above, is the invention disclosed in his 1982 application: the four-part-editor-compiler. As described in the patents, the compiler executes continuously in the background and is interrupted, not by a clock, but when the "programmer strikes a key on the console keyboard." ' 603:4:14-15. Plaintiff's detailed description of his preferred embodiment is nearly entirely devoted to a description of a system that does not rely on a clock activated interrupt. Indeed, in the patents' detailed description only a single brief paragraph, the same paragraph in each patent, mentions a clock or timer:

Furthermore, the interrupt which causes control of the CPU to pass from the compiler to the editor may be activated by a timer or clock instead of by a keyboard * * *. That is, the compiler may be periodically interrupted and the input port polled to test if a key has been struck. If not, the interrupt is terminated and

control returns to the compiler. If polling the port reveals that a key has been struck then the interrupt service routine editor takes control and is executed in the manner described above. For most applications clock interrupts at intervals of about every 10 to 30 milliseconds should be frequent enough to keep up with the keys stroked at the keyboard.

'603:13: 8-20; '604:10:3-14.

Aside from summary sections, this paragraph is the only mention of a clock activated interrupt form of multithreading in the detailed description section of the patents in dispute. The claims of both patents, however, discuss plaintiff's invention only in terms of periodic preemption based on timeslices assigned to the threads. This apparent disconnect between the detailed description of plaintiff's preferred embodiment and the patents' claims may well raise serious 35 USC ' 112 validity questions. For the purposes of claim construction, however, the court is obliged to interpret "what the patentee has chosen to claim as his invention." SSIH Equipment SA v. U.S. Int'l Trade Comm., 718 F.2d 365, 378 (Fed.Cir.1983). Valid or not, plaintiff's clear intent is to claim the invention of a form of multithreading that operates pursuant to a periodic clock activated preemption.

The parties disagree whether the patents' version of multithreading excludes multithreading wherein the assignment of which thread may execute after the executing thread is interrupted is governed by the relative priority of threads. The patents do not discuss any method of assigning priority to threads; nor do the patents indicate that so assigning priority would be a possibility or, in fact, desirable. Plaintiff argues that the source code compiler embodiment is priority driven because the editor operates in the foreground and the compiler operates in the background. In this embodiment, however, only two threads operate and, therefore, it is never necessary to choose which thread operates next when one is interrupted, accepting for the moment plaintiff's contention that the editor may be interrupted. Plaintiff does not discuss an embodiment which operates with more than two threads; nor does his specification specifically discuss how the chain of execution will be determined.

Clearly plaintiff's claimed invention does not require a feature that assigns priority to threads. Nevertheless, the court thinks it premature to determine whether a form of multithreading that operates pursuant to the patents' description, but *includes* a method for assigning priority amongst threads would infringe plaintiff's patents. If as the litigation proceeds it becomes necessary further to clarify the court's construction of multithreading, the court will undertake that task. It is likely, however, that this determination will take place as part of a fact-laden infringement analysis and not claim construction, which is a matter of law.

If there is any material difference between the explicit definitions of multithreading given in the patents, it relates to whether the '604 definition, which provides that multithreading is "controlled" by a clock or a timer, is limited to preemptive multithreading. As the court has decided that it is premature to make this determination, there is currently no need to choose between the explicit definitions of multithreading given in each patent. Accordingly, the court determines that multithreading shall be construed in each patent as explicitly defined in that patent. To the extent there is any material difference between these definitions, the patents shall be considered to be working with different definitions of multithreading.

THREAD

Microsoft argues that plaintiff's proposed construction of thread as an "executable entity within a process," Pl Br (Doc # 331) at 19, fails to be sufficiently specific. Microsoft cites plaintiff's discussion of the

definition of thread in a reply to the patent office concerning newly cited references. In this discussion, plaintiff stated:

The phrase "sequence of instructions executed by a processor" could not possibly have constituted an all-inclusive "definition" of the term "thread" because the phrase is so broad as to refer to many distinct and different things, such as (1) a program, (2) a routine, (3) a subroutine, (4) a function, (5) a job, (6) a task, (7) a thread, (8) an algorithm, (9) a process, or (10) a procedure, each of which consists of or comprises or refers to a "sequence of instructions executed by a processor.

11/18/93 Affidavit at para. 11, cited in Def Exh I (Doc # 311, Exh # 16)

Microsoft, however, fails to note plaintiff's definition of thread stated within the same discussion.

"To simplify in order to provide some sorely needed clarity, a "process" is the instance of running a program, whereas in current terminology a "thread" is the execution of a sequence of instructions constituting one of the possibly many procedures, functions or subroutines within the program. That is, in current terminology, a "thread" is merely one of the many resources of a "process."

Id at para. 14.

The parties dispute whether a thread, or an execution of a sequence of instructions within a program, is required to have its own private memory stack. Because a thread is interrupted during multithreading, the thread's context must be saved and retrievable when the thread is ready to resume execution. The question is whether a thread, by definition, saves its context in a private memory stack or whether multiple threads may share a stack in which their contexts is saved. Microsoft, citing a variety of technical treatises and plaintiff's own reference to these treatises in communications with the patent office, argues that a thread is required to have its own private memory stack. Plaintiff disputes this requirement.

Plaintiffs' patents explicitly refer to the compiler and editor in the preferred embodiment as threads: "In this illustrative example, one of the program threads is an editor and another thread is a code processing routing in the form of a compiler. '603:2:3-5. The editor and compiler are not, however, assigned their own private memory stack in which to save their context.

The court is, therefore, faced with a conflict between two sources of evidence of the meaning of claim terms. Dictionaries and technical treatises are generally considered sources of extrinsic evidence. See e g, Vitronics Corp. v. Conceptronic, Inc., 90 F.3d 1576, 1583 (Fed.Cir.1996). Plaintiff, however, repeatedly quoted dictionary and treatise definitions of thread to the patent office in an attempt to distinguish his invention from prior art, bringing these definitions within the realm of intrinsic evidence.

Plaintiff engaged in a lengthy discussion of the use of "thread" in his patent applications in a 1993 response to the patent office regarding newly cited references. 11/18/93 Affidavit, cited in Def Exh I (Doc # 311, Exh # 16). In particular, plaintiff attempted to distinguish the use of the term "process" in the Dufond reference from plaintiff's use of the term "thread." As noted above, plaintiff argued that while a process was, essentially, a program, a thread was a sequence of instructions within that program. See *id* at para.para. 12-15. In support of this distinction, plaintiff cited several industry definitions, some of which included the requirement that a thread have a private memory stack. Plaintiff, for example, cited:

To work successfully with multitasking, you need to understand clearly the difference between a process and a thread. A process is simply the code, data, and other resources of a program memory, such as the open files, allocated memory, and so on. MS OS/2 considers every program that it loads to be a process. A thread, which is everything else required to execute the program, consists of a stack, the state of CPU registers, and an entry in the execution list of the system scheduler.

Id at para. 18, quoting *Microsoft Operating System/2 Programmer's Reference,* Vol 1, Microsoft Corp, 1989, Pp 587, 593.

It is important to note, however, that plaintiff cited such definitions in an attempt to distinguish between a process and a thread. Microsoft cites other technical treatises in support of its position that a thread must have its own private memory stack. Microsoft cites, for example, one of its programmer reference guides, which provides:

Threads are the basic entity to which the operating system allocates CPU time. Each thread maintains a set of structures for saving its context while waiting to be scheduled for processing time. The context includes the thread's set of machine registers, the kernel stack, a thread environment block, and a user stack in the address space of the thread's process.

Supp App (Doc # X, Exh 5) at A2008, quoting *Microsoft Win 32 Programmer's Reference,* Vol 2, 1993.

Although the patents provide that when the compiler of the preferred embodiment is interrupted the stack pointer and other CPU registers are saved by the interrupt service routine, the compiler is not allotted a private memory stack. See '603:4:25-30. Microsoft's proposed construction, therefore, runs up against another principle of claim construction, which, in Microsoft's preferred position, was argued with vigor, namely: "[A] claim interpretation that would exclude the inventor's device is rarely the correct interpretation." Modine Mfg. Co. v. United States Int'l Trade Comm'n, 75 F.3d 1545, 1550 (Fed.Cir.1996). Further:

A patent claim should be construed to encompass at least one disclosed embodiment in the written description portion of the patent specification. * * * A claim construction that does not encompass a disclosed embodiment is thus rarely, if ever, correct and would require highly persuasive evidentiary support.

Johns Hopkins Univ. v. Cellpro, Inc., 152 F.3d 1342, 1355 (Fed.Cir.1998), citations and internal quotations omitted.

It is also worthy of note that at the time of plaintiff's patent applications, the idea of a "thread" was a new one:

Until the latter half of the 1980s, most operating systems allowed a process to have only one thread of execution. (In fact, most operating systems used the term *process* to refer to an executable entity. *Thread* is a relatively new term.).

Helen Custer, *Inside Windows NT,* Microsoft Press, 1993 at 93.

Clearly, plaintiff intended to distinguish a thread from a process. Further, when a thread is interrupted, its

context must be saved in some manner and be retrievable when it begins to operate. Yet in light of plaintiff's general definition given to the patent office and in light of the patents' preferred embodiment, the court cannot conclude that a thread must have its own private memory stack. As a result, the court determines that the proper construction of thread is that which plaintiff provided to the patent office in November of 1993. Accordingly, a thread is the execution of a sequence of instructions constituting one of the possibly many procedures, functions or subroutines within the program. Further, when interrupted, a thread's context must be saved and retrievable when a thread is reassigned control of the CPU and resumes execution. If a determination of whether a thread is required to have a private memory stack becomes central to determining infringement, the court may be open to revisiting its construction of this claim term.

TIMESLICE

Microsoft argues that timeslice has a different meaning in the context of periodic multithreading, which is the subject of plaintiff's patents, and priority-based multithreading, which is the method utilized by other forms of multithreading, including some prior art that plaintiff distinguished during prosecution. Microsoft argues that in priority-based multithreading, a timeslice is the *maximum* amount of time a thread may run without being interrupted and preempted. If a higher priority thread is ready to run during the timeslice, the lower priority thread will be interrupted. See Def Br (Doc # 310) at 17. In periodic multithreading, however, Microsoft argues that a timeslice is a fixed length of time during which a thread operates before it is interrupted.

Microsoft's construction is supported by the language of the patents and their prosecution history. Towards the end of his patent prosecution, in order to overcome prior art (the "Cheriton" reference), plaintiff vigorously argued to the patent office that his form of multithreading was novel and distinct, because in plaintiff's invention the timeslice during which a thread may operate was a fixed length of time.

In Cheriton the timing of preemption of the active process occurs at variable non-predetermined times whenever another process, having a higher priority than the active process, happens to become "ready" (for execution), and this event is neither predetermined nor predictable

In applicant's invention preemption of a currently executing thread occurs at fixed predetermined time intervals at the expiration of its current execution timeslice of fixed predetermined duration.

3/15/95 Decl at para. 36(a) and (b), cited in Def Exh I (Doc # 311, Exh # 44).

This interpretation of timeslicing is buttressed by plaintiff's specification. Plaintiff argues that the editor in his preferred embodiment may be interrupted before its timeslice has run and, therefore, that "timeslice," as used in the claims, refers to a maximum period, not a necessary period, during which a thread may run. Besides conflicting with the prosecution history, however, this interpretation conflicts with plaintiff's description of his embodiment. The (single) paragraph discussing timesliced preemption in the patents' written description refers only to "the interrupt which causes control of the CPU to pass from the compiler to the editor." '603:13:8-9; '604:10:3-4. The patents' description does not even mention the possibility that the editor may be interrupted before it has processed the keystroke.

At the claim construction hearing, when asked for support in the patents for the claim that the editor could be interrupted before completing its task, plaintiff's counsel cited the last sentence of the patents' timesliced preemption paragraph, which states: "For most applications clock interrupts at intervals of about ever 10 to

30 milliseconds should be frequent enough to keep up with the keys stroked at the keyboard." '603:13:17-20; '604:10:11-14. Plaintiff argues that this sentence indicates that the editor too may be interrupted at intervals of 10 to 30 milliseconds. This interpretation, however, is contradicted by the preceding language of this paragraph. After noting that the compiler may be interrupted and the input port polled to determine if a key has been struck, the patents state that if a key has not been struck,

the interrupt is terminated and control returns to the compiler. If polling the port reveals that a key has been struck then the interrupt service routine editor takes control and is executed in the manner described in the above embodiment.

'603:13:12-15; '604:10:8-11.

The description of the editor's execution in the above embodiment does not provide a mechanism by which the editor may be interrupted. As a result, the court concludes that the proper construction of timeslice is the fixed, predetermined length of time during which each thread is given uninterrupted control of the CPU, at the expiration of which the executing thread is preempted in favor of another thread.

PREEMPTION

Plaintiff argues that preemption refers to the taking of control away from an executing thread. Microsoft argues that preemption requires more than interruption and includes the requirement that control must be not only taken away from the executing thread, but control must be given to another thread. This construction would make plaintiff's claims read against his preferred embodiment, as although control is periodically taken away from the compiler, it is only given to the editor when a keystroke has been entered. The court is aware of no reason to impose an additional requirement upon the term preemption.

As a result, the court determines that a thread is "preempted" when control is taken away from it.

CONCURRENT

Microsoft contends that the concurrent limitation should include the restriction that each thread must be interrupted after a predetermined timeslice and that no thread may execute with interrupts disabled. This condition would, however, mean that the preferred embodiment disclosed by plaintiff's patents is not concurrent, because in this embodiment the editor is not interrupted after a predetermined timeslice when processing a keystroke, but is allowed to run until the keystroke is processed. To adopt the interpretation urged by Microsoft, therefore, would be to determine that plaintiff's patents are invalid by introducing a fatal disconnect between the claims and other elements of the specification.

Clearly, plaintiff intended the condition that multiple threads operate "concurrently" to mean that the user perceive them to be operating at, essentially, the same time. The question is whether this condition goes beyond a perception by the user to a requirement that no thread may control the CPU for as long as required to finish its task. The court concludes that plaintiff did not intend "concurrently" to mean that no thread could control the CPU, with interrupts disabled, while executing its task. Rather, plaintiff intended "concurrently" to be a measure of the user's perception. Consider claim two of the '603 patent, which states in part:

whereby said one thread executes interactively with the user in the foreground while another thread of the same program executes in the background *concurrently* with said one thread with control of the central

microprocessor repeatedly switching back and forth between threads, and wherein said fast efficient preemption provided by said direct access by the threads to said memory address space enables control of the central microprocessor to be switched between the threads so rapidly that an interactive user perceives the foreground and background threads to be executing simultaneously and without any perceptible interference by the background thread with the user's interaction with the foreground thread.

'603:14:47-53, emphasis added.

The patents clearly contemplate that the modifier "concurrently" focuses on the user's perception. "Concurrently" thus refers to: A mode of operation wherein control of the CPU is alternated among executing threads of the same program at so rapid a rate that the multiple threads of execution appear to be executing simultaneously to the computer user.

B

Microsoft contends that several of plaintiff's claims include means-plus-function claim limitations, in accordance with 35 USC s. 112, para. 6, which states:

An element in a claim for a combination may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.

35 USC s. 112, para. 6.

Claims written in means-plus-function form are construed to cover the structure performing that function disclosed in the specification. The first thing for the court to consider is whether a claim phrase invokes the means-plus-function analysis. "If a claim element contains the word 'means' and recites a function, this court presumes that element is a means-plus-function element under s. 112, para. 6." Envirico Corp. v. Clestra Cleanroom, Inc., 209 F.3d 1360, 1364 (Fed.Cir.2000). "That presumption fails, however, if the claim itself recites sufficient structure to perform the claimed function." *Id*.

Claim 14 of the '603 patent, a dependent claim, claims:

A disk means * * * wherein said means for changing said stored body of data comprises editor instructions executable by said computer for editing said body of data while said body of data remains stored in memory means, said body of data comprising words of a language, said first thread comprising means for checking the spelling of said words stored within said memory means.

'603:17:22-29.

The phrase in dispute, "means for checking the spelling of said stored words," reappears in claims 30-33 of the '603 patent. Plaintiff contends that this claim element discloses sufficient structure to remove it from the ambit of s. 112, para. 6. As claim 14 is dependent on claim 12, the "said first thread of instructions" referred to in claim 14, refers to the "first thread of instructions executable by the microcomputer," which in turn is one element of:

A computer-readable disk means encoded with a plurality of concurrently executable threads of instructions constituting a multithreaded computer application program to control the execution of a desktop microcomputer having an interrupt operation, a clock timer for periodically activating said interrupt operation, and memory means for storing a body of data * * *.

'603 patent, 16 :60-64.

The court, therefore, must determine whether the phrase "thread of instructions executable by the microcomputer" describes structure sufficient to remove the phrase "means for checking the spelling of said stored words" from the ambit of s. 112, para. 6. In making this determination, the court finds it helpful to note the history of the means-plus-function element of s. 112, para. 6.

Before 1952, means-plus-function claim language was prohibited and rendered claims of which it was a part invalid, as the Supreme Court feared that such language was overbroad and ambiguous. See Halliburton Oil Well Cementing Co. v. Walker, 329 U.S. 1, 9, 67 S.Ct. 6, 91 L.Ed. 3 (1946). In the 1952 Patent Act, however, Congress decided to permit broad means-plus-function language, but provided a standard, expressed in the current s. 112, para. 6, to make this broad language more definite. See Valmont Industries, Inc. v. Reinke MFG. Co., Inc., 983 F.2d 1039, 1042 (Fed.Cir.1993). The reason for the limiting condition of para. 6, which provides that a means-plus-function claim limitation refers back to structures disclosed in the specification, is that without this limitation a means-plus-function clause could encompass *any* conceivable means capable of performing the described function. See *id*.

If stated without limitation, the function "checking the spelling of said stored words" could be accomplished by any variety of "means." Plaintiff's claims, however, expressly designate that this function is to be accomplished through a particular structure, namely "a thread of instructions executable by the microcomputer." This microcomputer, moreover, is described as being "a desktop microcomputer having an interrupt operation, a clock timer for periodically activating said interrupt operation, and memory means for storing a body of data." In this light, the executable thread 20 of instructions clearly refers to a programming algorithm. In the court's view, in the context of plaintiff's patents, this designates sufficient structure to remove the phrase "means for checking the spelling of said stored words" from the ambit of s. 112, 16.

The '603 patent does not disclose any particular algorithm that can check the spelling of stored words, but the patent does sufficiently specify the structure intended to perform that function. In certain infringement contexts, the failure to disclose the algorithm itself *may* raise validity questions. When discussing software programs, however, disclosing the software structure and the function that software is expected to execute may be enough to satisfy the disclosure requirements of s. 112. See e g, Fonar Corp. v. General Elec. Co., 107 F.3d 1543 (Fed.Cir.1997). In discussing whether the failure to disclose the substance of the algorithm rendered the patent invalid for failing to satisfying s. 112 disclosure requirements, the Federal Circuit in *Fonar Corp* noted:

As a general rule, where software constitutes part of a best mode of carrying out an invention, description of such a best mode is satisfied by a disclosure of the functions of the software. This is because, normally, writing code for such software is within the skill of the art, not requiring undue experimentation, once its functions have been disclosed.

Id at 1549.

Based on the foregoing, the court concludes that the patents' designation that the spell check function is to be accomplished by a thread of instructions executable in the described manner on the specified microcomputer sufficiently discloses structure so as to clarify the means by which the function is to be performed and remove the phrase from the meansplus-function ambit. A variety of algorithms, all within the skill of the art, could be used to check the spelling of words of a natural language. Plaintiff is not attempting to patent a particular spell check program or a spell check program that is executed *after* the text is entered. Rather plaintiff claims a system for running a spell check program at the same time text is entered. Because plaintiff's claimed invention is not a particular spell check algorithm itself, but a system for processing that algorithm, plaintiff did not need to disclose the contents of a spell check algorithm for the purposes of claim construction, but merely needed to specify the type of structure by which spell checking was accomplished within plaintiff's claimed invention.

The court expects that this discussion will apply to other claim limitations that Microsoft alleges require means-plus-function analysis. It applies in full, for example, to the phrase from claim 6 of the '603 patent, "said means for analyzing said language code." '603:15:41-42. This dependent claim refers back to the "first set of instructions," which is part of the disk memory set forth in claim 1. '603:15:35. Again, plaintiff is not attempting to patent a particular algorithm for analyzing language code, which would not be novel. Rather, the patents claim the invention of a system for integrating the execution of a set of instructions capable of analyzing language code into the execution of the processing of the entry of that language code. As a result, the specification that the analyzing process will occur through a thread, or a set of instructions, sufficiently discloses structure to remove the phrase, "said means for analyzing said language code," from the ambit of s. 112, para. 6.

As noted, this discussion will likely apply to other phrases that Microsoft contends require means-plus-function analysis. The court declines to go through this analysis for all the claim phrases identified by Microsoft at this time. If particular phrases require further clarification as litigation proceeds, the court will revisit its construction of claims allegedly employing means-plus-function language.

IT IS SO ORDERED.

N.D.Cal.,2002.
Reiffin v. Microsoft Corp.

Produced by Sans Paper, LLC.