MORRISON & FOERSTER LLP
MICHAEL A. JACOBS (Bar No. 111664)
mjacobs@mofo.com
MARC DAVID PETERS (Bar No. 211725)
mdpeters@mofo.com
DANIEL P. MUINO (Bar No. 209624)
dmuino@mofo.com
755 Page Mill Road, Palo Alto, CA 94304-1018
Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

BOIES, SCHILLER & FLEXNER LLP
DAVID BOIES (Admitted *Pro Hac Vice*)
dboies@bsfllp.com
333 Main Street, Armonk, NY 10504
Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
STEVEN C. HOLTZMAN (Bar No. 144177)
sholtzman@bsfllp.com
1999 Harrison St., Suite 900, Oakland, CA 94612
Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

ORACLE CORPORATION
DORIAN DALEY (Bar No. 129049)
dorian.daley@oracle.com
DEBORAH K. MILLER (Bar No. 95527)
deborah.miller@oracle.com
MATTHEW M. SARBORARIA (Bar No. 211600)
matthew.sarboraria@oracle.com
500 Oracle Parkway, Redwood City, CA 94065
Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

Attorneys for Plaintiff
ORACLE AMERICA, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.

Plaintiff,

v.

GOOGLE INC.

Defendant.

Case No. CV 10-03561 WHA

ORACLE AMERICA, INC.'S OPPOSITION TO GOOGLE INC.'S SECOND MOTION TO STRIKE PORTIONS OF THE MITCHELL PATENT REPORT

Date: October 13, 2011
Time: 8:00 a.m.
Dept.: Courtroom 8, 19th Floor
Judge: Honorable William H. Alsup

I. INTRODUCTION

Oracle opposes Google's second motion to strike portions of the Mitchell Opening Patent Infringement Report ("Report") and the Mitchell Reply Patent Infringement Report ("Reply Report").

II. ARGUMENT

In the Court's September 26, 2011 Order, the Court held: "Rule 3-1(c) required identification of the precise element of any accused product that was alleged to practice a particular claim limitation; it did *not* require identification of every evidentiary *item of proof* showing that the accused element did in fact practice the limitation. Google confuses this distinction." (ECF No. 464 at 4.) In Google's four new critiques, Google continues to confuse the distinction. As shown below, the precise element of Android that Prof. Mitchell identifies as practicing a given claim limitation is the same in Oracle's ICs.

A. Response to Critique D: The Report on Infringement of the '720 Patent Is Supported by the ICs

Oracle's '720 patent infringement theory focuses on a function present in Android called *do_fork()*. Page 19 of Oracle's claim chart for the '720 patent identifies Android's *do_fork()* function as practicing the "copy-on-write process cloning mechanism" element of Claim 1 and quotes from the relevant source code that implements it (fork.c). (Declaration of Brian C. Banner ("Banner Decl.") Ex. C, Oracle ICs Ex. G at 19.) Page 38 of Oracle's claim chart likewise identifies Android's *do_fork()* function as practicing the "process cloning mechanism" element of Claim 6, and includes the same source code quote. (*Id.* at 38.) Oracle ICs thus alleged that the *do_fork()* function in Android satisfies both the "copy-on-write process cloning mechanism" of Claim 1 and the "process cloning mechanism" of Claim 6.

Prof. Mitchell presents the same *do_fork()* infringement theory in his Report. Prof. Mitchell's analysis of how Android includes the copy-on-write process cloning mechanism of Claim 1 concludes with an identification of *do_fork()*:

> **The file linux-2.6\kernel\fork.c provides the *do_fork()* code,** that calls the *copy_process()* module, **to perform the copy-on-write process cloning** "to instantiate the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process, and to defer copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process."

(Declaration of Marc Peters ("Peters Decl.") Ex. A, Report ¶ 614 (emphasis added).) Similarly, Prof. Mitchell's infringement theory for Claim 6 rests on *do_fork()*. His analysis of how Android includes the process cloning mechanism of Claim 6 concludes with an identification of *do_fork()*:

> **The file linux-2.6\kernel\fork.c provides the fork code *do_fork()* to perform the process cloning** "to instantiate the child runtime system process by copying the memory space of the master runtime system process into a separate memory space for the child runtime system process." The unset CLONE_VM flag from the *clone()* system call can be passed as an argument for parameter *clone_flags* to *do_fork()*, which passes it to *copy_process()*, which performs the process cloning. As stated previously, because *clone()* does not set the CLONE_VM flag, ***do_fork()* provides the "process cloning mechanism" of claim 6** "to instantiate the child runtime system process by copying the memory space of the master runtime system process into a separate memory space for the child runtime system process."

(Banner Decl. Ex. A, Report ¶ 627 (emphasis added).) This is one of the paragraphs that Google moves to strike. But Prof. Mitchell's *do_fork()* infringement theory matches the one in the ICs exactly. In addition, his discussion of the `clone_flags` parameter of the *do_fork()* function, which Google also seeks to strike, is not new either. Oracle identified the `clone_flags` parameter in the ICs for both Claims 1 and 6. (Banner Decl. Ex. C, Oracle ICs Ex. G at 19, 38.)

Google's complaint about Prof. Mitchell's references to the *fork()* and other system calls mischaracterizes Prof. Mitchell's reliance on them. He invoked the *fork()* system call, for example, to explain how Android's Zygote functionality uses *fork()* when it receives a command to fork a new VM instance, and this in turn results in a call to *do_fork()*. (*See* Peters Decl. Ex. A, Report ¶¶ 598-614.) Similarly, he explained how the *do_fork()* routine provides the different types of process cloning in Claims 1 and 6 by referring to a Linux textbook, and explained how the *fork()*, *clone()*, and *vfork()* system calls invoke *do_fork()* with different `clone_flags` settings, including CLONE_VM (which is unset for both *fork()* and *clone()*). (*See id.* at ¶ 606, pages 302-305.) But this is not a new infringement theory—it is rather further flesh on the bones of the

original *do_fork()* theory. To borrow the Court's metaphor, the discussion of the *fork()*, *clone()*, and *vfork()* system calls is just to explain in which toolboxes the accused *do_fork()* hammer may be found. Prof. Mitchell is permitted to explain in his Report the infringement theories disclosed in Oracle's ICs. *Sicurelli v. Jeneric/Pentron Inc.*, No. 03-CV-4934 (SLT) (KAM), 2005 U.S. Dist. LEXIS 42233, at *33 (E.D.N.Y. May 3, 2005) ("[E]xpert reports are expected to provide additional information regarding the specific factual bases for plaintiffs' infringement contentions.").

The ICs gave Google sufficient notice of Oracle's *do_fork()* infringement theory. The Report adheres to that theory and provides an explanation of the theory, with citations to appropriate evidence. The Court should reject Google's Critique D.

B. Response to Critique E: There Is No New "mBS Mobile" Infringement Theory for the '476 Patent

Oracle's ICs allege that Android's implementation of the Java Security Framework, which includes such Java-API-specified classes as AccessController, ProtectionDomain, and SecurityManager, infringes the '476 patent. (In fact, these are some of the Java APIs that Google copied from Oracle.) Google moves to strike two identical paragraphs (70 and 77) of the Report that discuss an Android application called "mBS Mobile." But neither the ICs nor Prof. Mitchell accuse the "mBS Mobile" Android application of infringement. Instead, Prof. Mitchell discusses the mBS Mobile application to describe the benefits that application developers receive from Google having included the infringing code within Android. The paragraphs that Google moves to strike begin: "Some of the ways that application developers could benefit from the java.security framework are illustrated by descriptions of the mBS Mobile application." (Banner Decl. Ex. A, Report ¶¶ 70, 77.) These paragraphs, by their very terms, do not disclose an infringement theory; they describe a "benefit" of infringement.

The Report discusses the mBS Mobile application in connection with the benefit offered by Android's infringing functionality to rebut Google's contentions on two points. First, the mBS Mobile evidence establishes that the infringing code that Google made was used, which rebuts Google's contention that the code was never used. Second, it establishes that SecurityManager

was enabled in Android—in fact, it was functional in at least all versions of Android through version 2.2. The Court held that Patent Local Rule 3-1(c) does not require identification of every evidentiary item of proof of infringement. (ECF No. 464 at 4.) Prof. Mitchell's discussion of mBS Mobile falls into this category. The Court should reject Google's Critique E.

C. Response to Critique F: The Reply Report on Infringement of Claim 14 of the '476 Patent Is Supported by the ICs

Oracle objects that Google did not obtain leave to move to strike any part of the Reply Report, as this is beyond the scope of its August 30, 2011 précis and the Court's September 26, 2011 Order, which concerned only the Report. For this reason alone, the Court should deny Google's Critique F.

Oracle's ICs identify Android's AccessController class as containing instructions that perform Claim 14's "determining whether said action is authorized based on an association between permissions and a plurality of routines in a calling hierarchy associated with said principal." (Banner Decl. Ex. D, Oracle ICs Ex. E at 22-23.) The ICs disclose that AccessController does the determining by calling the checkPermission method of Android's AccessControlContext class. (*Id.* at 23-24.) The ICs quote from Android's AccessController source code to specifically identify that its checkPermission method is declared to be a "public" and "static" method. (*Id.* at 23.)

Google's complaint is that Prof. Mitchell's Reply Report addresses infringement when SecurityManager is disabled. But there is nothing in the infringement theory in Oracle's ICs that *requires* Android's SecurityManager class for Android to infringe. It is true that use of Android's SecurityManager results in infringement, but that is because SecurityManager calls AccessController's checkPermission method. This fact was disclosed in the ICs (*id.* at 21) and was discussed in Prof. Mitchell's Report. (Banner Decl. Ex. A, Report ¶ 736.) But there is nothing in either Oracle's ICs or the Report that limits the infringement theory to SecurityManager alone.

The paragraph of Prof. Mitchell's Reply Report that Google moves to strike rests squarely on the identification of AccessController in Oracle's ICs. Here it is in full:

> Even if the SecurityManager is disabled, the Java security framework may still be used. For example, "the static methods in AccessController are always available to be called." (L. Gong et al., Inside Java 2 Platform Security 109 (2nd Ed. 2003).) Even if no SecurityManager is instantiated, Android code includes AccessController.java and the static methods associated with the class may be called. Therefore, the functionality of the Java security framework is accessible via the AccessController.

(Banner Decl. Ex. B, Reply Report ¶ 115.) In this paragraph, Prof. Mitchell explains, with reference to a textbook written by the '476 inventor, that Android includes AccessController and its static methods whether or not a SecurityManager object was ever created (instantiated). This is a reference to the accused AccessController and its static method checkPermission that was identified on page 23 of Oracle's ICs. It is not a new infringement theory because it is not pointing to any new accused functionality; it is merely pointing out that infringement occurs even if SecurityManager is disabled.

Google's Critique F is thus directly analogous to the critique regarding DvmDex.h that the Court rejected in Google's first motion, and should be rejected for the same reason. (ECF No. 464 at 4 ("[T]he Mitchell report identified specific *functions* located within that file as satisfying the storing limitation. Those functions also were identified in the relevant portion[s] of Oracle's infringement contentions.").) *DataTreasury Corp. v. Wells Fargo & Co.*, No. 2:06-CV-72 DF, 2010 U.S. Dist. LEXIS 110658, at *23 (E.D. Tex. Sept. 13, 2010) (denying defendant's motion to strike portions of plaintiff's infringement expert report because report did "not substantially deviate from [the] infringement contentions"). The Court should reject Google's Critique F.

D. Response to Critique G: The Reply Report on Infringement of the '520 Patent Is Supported by the ICs

Oracle again objects that Google did not obtain leave to move to strike any part of the Reply Report, as this is beyond the scope of its August 30, 2011 précis and the Court's September 26, 2011 Order, which concerned only the Report. For this reason alone, the Court should deny Google's Critique G.

Oracle's ICs identify the Simulator class (and, in particular, its "simulate" methods) as performing the step of "simulating execution of the byte codes of the clinit method against a

memory without executing the byte codes to identify the static initialization of the array by the preloader" that is recited in Claim 1 of the '520 patent. (Banner Decl. Ex. E, Oracle ICs Ex. F at 11-15.) Google described Simulator as the "Class which knows how to simulate the effects of executing bytecode." (*Id.* at 11.) As the ICs disclose, Simulator employs several helper methods to perform its simulating functions, including the parseInstruction and parseNewarray methods. (*Id.* at 13, 14, 15) For the "storing . . . an instruction" step of Claim 1 of the '520 patent, Oracle's ICs identify the "writeDex" method of the Main.java file as performing the step. (*Id.* at 21-22.)

The quotation in Google's brief of Prof. Mitchell's in paragraph 92 of the Reply Report is incomplete. (Mot. at 5.) The full paragraph that Google moves to excise portions of is:

> parseNewarray is part of the functionality and operation of Simulator.java, the "class which knows how to simulate the effects of executing bytecode," and which Prof. Parr concedes simulates bytecodes. It is not uncommon that functionality be carried out or span multiple classes of files. The dx tool was designed to receive class files that had been created by a Java compiler, and parseNewarray interprets the array initialization bytecodes that result from compilation to determine the static values, which are then used to create (and store) one or more instructions requesting the static initialization of the array. The claims do not require the simulating step to succeed in identification of the static initialization of the array in all cases, which is why Prof. Parr's example in his Appendix A, Section II is not evidence of noninfringement. **Actually, it is evidence of infringement, because the operation of the Simulator on his modified T.class file produced instructions requesting the static initialization of the array, which were stored in the output .dex file.** I note that those instructions were fewer than the original number of bytecodes from the .class file, which is the purpose of the '520 patent.

(Banner Decl. Ex. B, Reply Report ¶ 92 (emphasis added).) Note that Prof. Mitchell did not, as argued by Google, identify the number of instructions stored in the .dex file as the reason for infringement. By use of an ellipsis to merge two different sentences, Google in its brief removed the text in which Prof. Mitchell identified "the operation of the Simulator" as his reason for finding infringement.

There is no infringement contention in paragraph 92 of the Reply Report that is not in Oracle's ICs. The analysis rests on Simulator, which was disclosed in the ICs. The purpose of this paragraph is to *respond* to an argument made by Google's noninfringement expert Prof. Parr, who created a class file (the "modified T.class file" referred to above) by manually altering a real class file with an editing tool and then processing the altered file with the Android dx tool. (It is worth noting here that the non-real-world example created by Prof. Parr was not disclosed in

Google's noninfringement interrogatories. (*See* Peters Decl. Ex. B.) When Prof. Mitchell opines that Prof. Parr's example is evidence of infringement rather than noninfringement, it is not a new infringement theory. The "modified T.class" file is not one that is actually produced when compiling Java source code using the real-world Android SDK, it is something Google's expert created. (*See* Peters Decl. Ex. C ("After compiling this class using the standard Java compiler (i.e., into a file named 'T.class'), I was able to use a library called ASM to alter the bytecode within the clinit method of the T.class file.").) Whether or not processing it satisfies one of the '520 patent claim limitations is not something that needed to be disclosed in Oracle's ICs.

It would be unfair to permit Google's expert to hypothesize a factual scenario different from real facts and opine that this scenario reveals something about the operation of the Android SDK that is evidence of noninfringement, and yet forbid Oracle's expert from responding to that opinion. All that Prof. Mitchell has done in paragraph 92 of the Reply Report is apply the Simulator-based infringement theory that was disclosed in the ICs to the new, hypothetical factual scenario created by Prof. Parr. Because that is permissible for experts to do, Google's attempt to take a scalpel to paragraph 92 must fail.

Another reason that the Court should reject Google's Critique G is because it is directly contrary to the position Google took in its noninfringement interrogatory response. On August 1, 2011, Google argued that Android does not infringe any asserted claim because "it would not employ a method that creates or stores a single instruction to perform each of the respective accused functions **in that there are multiple instructions identified in Exhibit F** [Oracle's '520 infringement chart] . . . ." (Peters Decl. Ex. B at 27 (emphasis added).) As even Google's own interrogatory response acknowledged, the ICs support a more-than-one-instruction infringement theory.

III. CONCLUSION

The Report and the Reply Report are fully supported by the ICs. Prof. Mitchell has adhered to the infringement theories that Oracle disclosed. For the foregoing reasons, Oracle asks this Court to deny Google's second motion to strike.

Dated: October 4, 2011

MICHAEL A. JACOBS
MARC DAVID PETERS
DANIEL P. MUINO
MORRISON & FOERSTER LLP

By: /s/ Michael A. Jacobs

Attorneys for Plaintiff
ORACLE AMERICA, INC.