

SNELL & WILMER LLP
Alan L. Sullivan (3152)
Todd M. Shaughnessy (6651)
15 West South Temple
Gateway Tower West
Salt Lake City, Utah 84101-1004
Telephone: (801) 257-1900
Facsimile: (801) 257-1800

FILED
CLERK, U.S. DISTRICT COURT
2004 AUG 23 P 5:02
DISTRICT OF UTAH
BY: 
DEPUTY CLERK

CRAVATH, SWAINE & MOORE LLP
Evan R. Chesler (admitted pro hac vice)
David R. Marriott (7572)
Worldwide Plaza
825 Eighth Avenue
New York, New York 10019
Telephone: (212) 474-1000
Facsimile: (212) 474-3700

*Attorneys for Defendant/Counterclaim-Plaintiff
International Business Machines Corporation*

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF UTAH**

THE SCO GROUP, INC.,

Plaintiff/Counterclaim-Defendant,

-against-

INTERNATIONAL BUSINESS
MACHINES CORPORATION,

Defendant/Counterclaim-Plaintiff.

**DECLARATION OF
RANDALL DAVIS**

Civil No. 2:03CV-0294 DAK

Honorable Dale A. Kimball

Magistrate Judge Brooke C. Wells

ORIGINAL

249

I. INTRODUCTION

1. I am a professor of Computer Science at the Massachusetts Institute of Technology in Cambridge, Massachusetts. Exhibit I provides more details of my technical background and experience, a list of publications, and a list of cases in which I have testified or been deposed. I received my undergraduate degree in Physics from Dartmouth College in 1970 and a Ph.D. in Computer Science from Stanford in 1976.
2. I have published some 50 articles on issues related to artificial intelligence and have served on several editorial boards, including *Artificial Intelligence*, *AI in Engineering*, and the MIT Press series in AI. I am a co-author of *Knowledge-Based Systems in AI*.
3. In recognition of my research in artificial intelligence, I was selected in 1984 as one of America's top 100 scientists under the age of 40 by *Science Digest*. In 1986 I received the *AI Award* from the Boston Computer Society for contributions to the field. In 1990 I was named a Founding Fellow of the American Association for AI and in 1995 was elected to a two-year term as President of the Association. From 1995–1998 I served on the Scientific Advisory Board of the U. S. Air Force.
4. In addition to my work with artificial intelligence, I have also been active in the area of intellectual property and software. Among other things, I have served as a member of the Advisory Board to the US Congressional Office of Technology Assessment study on software and intellectual property, published in 1992 as *Finding a Balance: Computer Software, Intellectual Property, and the Challenge of Technological Change*. I have published a number of articles on the topic, including co-authoring an article in the *Columbia Law Review* in 1994 entitled "A Manifesto Concerning Legal Protection of Computer Programs" and an article in the

Software Law Journal in 1992 entitled “The Nature of Software and its Consequences for Establishing and Evaluating Similarity.”

5. From 1998-2000 I served as the chairman of the National Academy of Sciences study on intellectual property rights and the emerging information infrastructure entitled *The Digital Dilemma: Intellectual Property in the Information Age*, published by the National Academy Press in February, 2000.

6. I have been retained as an expert in over thirty cases dealing with alleged misappropriation of intellectual property, such as the allegations raised in this case, and have done numerous comparisons of code. I have been retained by plaintiffs who have asked me to investigate violations of intellectual property, by defendants who have asked me to investigate allegations made against them, and by both sides to serve as the sole arbiter of a binding arbitration.

7. In 1990 I served as expert to the Court (Eastern District of NY) in *Computer Associates v. Altai*, a software copyright infringement case that articulated the abstraction, filtration, comparison test for software. I have also been retained by the Department of Justice on its investigation of the INSLAW matter. In 1992 (and later in 1995) my task in that engagement was to investigate alleged copyright theft and subsequent cover-up by the Federal Bureau of Investigation, the National Security Agency, the Drug Enforcement Agency, the United States Customs Service, and the Defense Intelligence Agency.

8. I have been retained by counsel for IBM in this lawsuit and am being compensated at a rate of \$550 per hour.

II. SUMMARY OF FINDINGS

9. I have been asked by counsel for IBM to evaluate the opinions set out in the declaration of Chris Sontag submitted in opposition to IBM's motion for partial summary judgment of non-infringement of copyright with respect to its Linux activities. Specifically, I have been asked to address Mr. Sontag's conclusions concerning what information is required in order to determine whether there is substantial similarity between Linux and SCO's allegedly copyrighted works.

10. My analysis and conclusions are based upon the principles described in *Gates Rubber v. Bando*, 9 F.3d 823 (10th Cir. 1993), which I understand to describe the appropriate methodology for determining substantial similarity and therefore to suggest what information is required to determine substantial similarity. My analysis and conclusions are also based upon my experience and expertise in the field of computer science.

11. In summary, I find fundamental errors in Mr. Sontag's conclusions. He grossly exaggerates what is required to determine whether there is substantial similarity between Linux and SCO's allegedly copyrighted works. In fact, the materials necessary to determine substantial similarity have been available to SCO for years (at least since it acquired access to the allegedly copyrighted works in 2001). Tools capable of efficiently evaluating that material have also been publicly available to SCO for years. The task Mr. Sontag says could take 25,000 man-years to complete should take capable programmers no more than several months.

12. The first section of this report describes my background and qualifications to address the issues addressed herein, while the second analyzes Mr. Sontag's declaration and

demonstrates the errors in his conclusions concerning the magnitude of the effort required to determine substantial similarity.

III. ANALYSIS OF THE SONTAG DECLARATION

13. Mr. Sontag correctly concludes that determining substantial similarity requires a comparison of the Unix and Linux kernels. (¶¶ 6-8.) He is incorrect, however, in concluding that the comparison requires a Herculean effort and that the only way SCO can feasibly conduct the comparison is if it is provided large quantities of additional information from IBM and from third parties.

14. As Mr. Sontag states in paragraph 8 of his Declaration: “To show that Linux code is substantially similar to Unix code requires a comparison of that code” Common sense supports that statement. Moreover, the *Gates Rubber* case is clear that determining substantial similarity requires a comparison of the works under consideration. 9 F.3d at 838-39.

15. Mr. Sontag acknowledges that SCO has access to both Linux and the allegedly copyrighted works. (¶¶ 19-23.) This is of course no surprise, as Linux is publicly available to everyone, and at least until recently, SCO was itself a distributor of Linux. As the alleged copyright holder, SCO must have copies of the allegedly copyrighted works.

16. Mr. Sontag goes on, however, to characterize as essentially impossible the task of comparing Linux to the allegedly copyrighted works, at one point suggesting that comparing just one version of the Linux kernel to one version of the Unix kernel “could take on the order of 25,000 man-years.” (¶ 14.) He then proposes a “shortcut” (¶ 15): “comparing similar directory structures of the Unix and Linux operating systems,” and immediately

indicates that this “shortcut” would still take “about 35 man-years,” and then only if the file names and the organization of all of the Linux kernel code were identical to the file names and organization of the Unix kernel code.

17. As I explain below, (1) the task of comparing Unix to Linux (for purposes of determining substantial similarity) is a manageable undertaking that could be accomplished by capable programmers, with the materials that have been available to SCO for years, in no more than several months; and (2) the additional materials that Mr. Sontag says are required so that the comparison will not endure for 25,000 more years are in fact absolutely unnecessary. The reasons offered by Mr. Sontag to explain why SCO has not been able to complete the task of determining substantial similarity are untenable.

III.1 Feasibility of the Task

18. As stated, SCO has had, since before the initiation of this case, all the raw material it needs to find any alleged substantial similarity between Linux and Unix. It of course has all relevant versions of Unix; it can get any version of the Linux kernel from publicly available web sites. As one example, www.linuxhq.com contains *every* version of the Linux kernel since the original 1.0.0 and a complete history of *every change* made to *every kernel file* over its *entire development history*. Exhibit II contains a sample listing from that site, showing the entire development history for the file `/fs/inode.c` through 1081 kernel versions.¹

¹ In the on-line listing, each version number is a link to a file showing exactly what changes were made to go from one version to the next.

19. There are a number of tools that are publicly available (in both executable and source code form) to compare large, complex programs for the purpose of determining substantial similarity. Three common varieties of tools used successfully in a number of cases like this one include:

- Comparing an exhaustive list of identifiers from both programs. It is a simple matter technically to assemble a complete list of every “word” in a body of source code (i.e., every variable name, function name, data structure name, field name, etc.), even one as large as those in use here. A list is created for each program and then compared (by the computer) to find words that show up in both lists. That list of terms in common can then be scanned to find “unusual” words (i.e., words not routinely found in code, or in the application in question), and the places where those unusual words are found in the text then become places to look for possibly copied code. As this process typically includes the (English) text contained in comments in the code, it at times even finds places where code has been changed, but comments left unchanged. As a result, it uncovers instances of copying that infringers may have thought they had obscured.
- Using tools that do literal or near-literal comparisons (e.g., COMPARATOR [1]). There are a variety of tools available to do this; they can be “tuned” to, for instance, ignore differences in spacing and layout (an issue raised by Mr. Sontag (¶ 11), ignore comments in the code, etc. SCO acknowledges using such tools, but misrepresents their utility. They are in my opinion quite effective.
- Using tools that do syntactic comparisons (e.g., SIM [2]). By syntax I mean the structure of the code, analogous to notions of noun, verb, adjective, etc., in English. Such tools find code with the same structure, ignoring entirely the actual names that have been used, and hence can be very effective even where code has been modified.

20. In an attempt to show that SCO could not possibly compare the works at issue without more time and information, Mr. Sontag states that existing tools may not detect minor changes in the code (¶ 11), are subject to false positives (¶ 13) and will require years to implement unless SCO is afforded more information (¶¶ 14-15).

21. First, the existing tools are entirely adequate, even accepting the observation (¶ 11) that minor changes can prevent an absolutely literal matching process from being effective.

There are several reasons why the existing tools will do the job. Despite SCO's implication, one cannot casually change punctuation, rename variables, change spelling, alter the text (whatever that means), or insert, delete, or reorder lines of code (§ 11). Code is extremely brittle and thus is in some respects quite similar to a complex and intricately designed mechanical device, like a finely made wristwatch. One can no more casually change punctuation, insert, delete, or reorder lines of code than one could casually insert, delete, or reorder the parts in the watch and still expect it to work.

22. Software development is difficult in large part exactly because the code has to be just right. In the C language (in which both Unix and Linux are written), for example, a semicolon means something very different from a comma: substituting one for the other changes the modified code completely (and very likely breaks it). Similarly, a single equal sign '=' means one thing, but two of them in a row '==' mean something entirely different. Hence even a minor typo can go unnoticed (because it can produce a syntactically valid program), yet wreak havoc on program behavior. Programmers routinely have the experience of serious and obscure malfunctions arising out of the simplest typographical mistake, or out of the well-intentioned act of making a small change to code. Code is sensitive to even slight alterations; changes are not easy to make.

23. This is one reason why, while it is in principle possible to copy code and then purposely obscure its origin, that practice is generally carried out on programs at the scale of freshman homework assignments (where it is more easily detectable than freshmen think), not sections of multi-million line operating systems. Especially where large, complex programs are concerned, changes are that much more difficult to make, and purposeful obfuscation on a large scale is nearly impossible.

24. Second, while all of these tools are subject to the problem of false positives, i.e., suggesting copying where none exists, with appropriate tuning and use the percentage of false positives can be kept well within reasonable bounds. Despite SCO's claims, checking those false positives need not be labor intensive: programs are easily created to present the alleged matches in a side-by side fashion on the screen, allowing an experienced programmer to determine with little more than a glance whether each match is worth further study. An experienced programmer can quickly scroll through a substantial amount of information to find any plausible matches.

25. Third, the existing tools could have been employed here in a reasonably short time period without any additional information. In fact, based on my experience, I estimate that the task would take experienced programmers a matter of months.

26. Fourth, the existing tools have been designed by experienced programmers who are aware of the kinds of modifications that can be made to code that may make its origin less obvious. The tools are capable of dealing with the sorts of things that concern Mr. Sontag (§ 11), such as differences in spacing and layout, variations in uppercase vs. lowercase, and comments in the code. The tools thus do both literal and non-literal matching, and are not misled by a variety of changes. There are also tools like SIM, noted above, that match based on syntax and hence are not misled by such things as renaming a variable, changing spelling, etc. (§ 11). The existing tools are thus fully capable of doing the job. Given the

claimed volume of the alleged copying, finding any copying that exists should not be a difficult task.²

27. Mr. Sontag poses the problem as if no results will be known until the entire comparison task is complete. Even if the entire task were daunting (which it is not), if “much” of SCO’s 3.5 million lines of code were copied (§ 47), this would imply that there must be thousands of examples waiting to be found, and hundreds able to be found after a modest amount of effort. Mr. Sontag’s own declaration acknowledges that SCO has used one or more of the existing tools to do the requisite comparisons (§§ 18, 21-23), but SCO has yet to present any credible examples of substantial similarity.

28. It would appear, in fact, that SCO completed months ago some of the very comparisons that Mr. Sontag says might take 25,000 man years. For example, on June 10, 2003, a SCO representative stated that it “was able to uncover the alleged violations by hiring three teams of experts, including a group from the MIT math department, to analyze the Linux and Unix source code for similarities” and that “[a]ll three found several instances where our Unix source code had been found in Linux”. (Robert McMillan (quoting SCO), “SCO shows Linux code to analysts,” IDG News Service, June 10, 2003.)

III.2 No Additional Information Is Required

29. Having concluded (incorrectly, I believe) that it is impossible as a practical matter to determine whether there is substantial similarity in the Unix and Linux kernels,

² For example, Mr. Sontag claims that “SCO believes that much of its copyrighted code was copied from AIX and Dynix into Linux.” (§ 47.)

Mr. Sontag states that the only way for SCO to determine substantial similarity is to get a vast amount of additional materials from IBM and a number of other individuals or entities. In fact, none of this additional material identified by Mr. Sontag is necessary to the substantial similarity task.

III.2.1 No Additional Information from IBM Is Needed

30. Mr. Sontag states (§§ 43, 50) that SCO requires the following additional materials from IBM:

- all version control system and bug tracking information (including documents, data, logs, files and so forth) for AIX, Dynix/ptx, ptx, and Dynix from 1984 to the present,
- source code and log information for all interim and released versions of AIX, Dynix, ptx and Dynix/ptx from 1984 to the present,
- depositions as appropriate for programmers identified from the foregoing,
- all design documents, white papers, and programming notes, created from 1984 to the present.

31. This information is irrelevant to SCO's task in the current context, which is showing that IBM's *Linux* activities infringe SCO's alleged copyrights in Unix software. The similarity to be demonstrated is between the claimed Unix software and Linux; the history of AIX and Dynix development plays no part in this judgment.

32. Having access to all of the materials concerning AIX and Dynix to which Mr. Sontag refers in his declaration (which appears to be a huge amount of information) would not, in my opinion, be of any assistance in determining whether *Linux* is substantially similar to *Unix*. Those materials are not useful for the task at hand.

33. As I understand the concept, and as is only logical, substantial similarity must be a determination about two bodies of code as they are, not a question of their heritage. Any given segment of Linux code either is or is not substantially similar to a given body of Unix code; it is irrelevant to the determination of similarity how the Linux code came to look the way it does. So even if it is true that some code that came to be included in Linux originated in AIX or Dynix, that is simply not important to the analysis of whether that Linux code is in fact similar to any Unix code.

34. To suggest otherwise leads to the absurd notion that one work can be considered similar to another even if the two are currently completely different, if only one can show a (perhaps very long) sequence of small changes that lead from one to the other. This would be like playing the game of “telephone,” in which a sentence is successively whispered from one person to the next in a long line, and claiming that, even though the sentence that emerged was totally different from the one that started the process, they were “substantially similar” because the last was the result of many small changes to the first. Similarity means just that — similarity. And the determination of similarity is made on the code as it is, independent of how it got that way.

35. Yet this is not what Mr. Sontag apparently has in mind, as he claims that “By viewing each version of the Dynix/AIX code, SCO will be better able to determine if the structure, sequence, and organization of the corresponding Linux code matches that of Unix.” (¶ 35.) The structure, sequence and organization of some Linux code either does or does not match that of some part of Unix code; that judgment is surely not dependent on the derivation history of the code. Again, information relating to AIX and Dynix, let alone extremely

detailed information about AIX and Dynix, is not useful to the analysis of whether portions of Linux are substantially similar to portions of Unix.

36. Consider also Mr. Sontag's indication (§ 30) that "Because of changes made to source code over time, the current code version may 'look' different than the initial code version." He slides too quickly here past the possibility that, because of changes to source code over time, the current code version may in fact *be* different, i.e., no longer substantially similar. The changes made to correct bugs, improve features, add new features, or as a consequence of re-thinking the design of a section of the code, may over time simply produce code that is completely different from the original.

37. Note also that the claim of the utility of intermediate versions and change logs skips blithely past a central point: in order to find any places where Unix code might have been copied into AIX or Dynix, SCO will have to compare all of its Unix code against all of the AIX or Dynix code. How else will it find all the similarities? But it is worse than that: SCO will have to compare all of its Unix code against *every version* of the AIX or Dynix code; after all, Mr. Sontag has noted (§ 48) that "IBM could have copied System V code into any number of the multiple versions of AIX and Dynix."

38. Here the basic contradiction in SCO's request becomes particularly clear. Mr. Sontag spends substantial space in his declaration suggesting that the task of comparing *one* version of Linux against *one* version of Unix is impossibly large. Given the IBM materials I understand SCO already has in hand – notably the many versions of Dynix and AIX that I understand IBM has produced – the task SCO proposes is many, many times larger than the one it claims is far too difficult. And yet SCO requests, among other things, "source

code and log information for all interim and released versions of AIX, Dynix, ptx and Dynix/ptx from 1984 to the present,” and “[a]ll design documents, white papers, and programming notes, created from 1984 to the present.” (§ 50.) If the original task of finding substantial similarity in one version is pragmatically impossible, what are we to make of a task many times larger, and one that is perhaps many, many times larger again than that? SCO seems in one breath to claim the task is too large and with the next claim that the task would become feasible if only it had a volume of information perhaps many, many times larger than what it already possesses.

39. It is estimated that the additional AIX and Dynix source code that SCO seeks exceeds 2 billion lines of code. Based upon the estimates Mr. Sontag used to arrive at his 25,000 man-years calculation, it would take SCO more than 14 million man-years to review just the additional AIX and Dynix code that SCO says it needs, putting aside how many more man-years it would require SCO to review the other materials it says it needs.³

³ Mr. Sontag (§ 14) states that comparing the 4 million lines of code in the Linux 2.4 kernel with the 3.4 million lines of code in SCO's Unix System V 4.2 MP kernel would require the comparison of 66,000 x 58,000 pages (assuming about 60 lines of code per page), and that a “initial” review of the code, assuming that each page comparison takes one minute, could take 25,000 man-years to complete. (66,000 x 58,000 equals 3,828,000,000 minutes, or 63,800,000 hours—SCO's calculation of 25,000 man-years thus implies a man-year of approximately 2,552 hours.) The approximately 2 billion lines of additional AIX and Dynix code would result in approximately 33,333,333 (again, using SCO's assumption of 60 lines of code per page) pages of code. SCO's suggested method of comparing code would thus require 66,000 x 33,333,333 = 2,199,999,978,000 minutes, or 36,666,666,300 hours, or 14,367,815 man-years to complete.

III.2.2.No Additional Information From Others Is Needed

40. As voluminous as the list of materials SCO seeks from IBM is, it is dwarfed by the request for additional information from other parties (§ 57):

- “Determine what third parties IBM has partnered with to develop Linux and what work those groups have done . . . particularly as to the details of the partnering, such as which party makes what contribution, the motivation for the contribution, and the starting and ending code versions that resulted from the partnership. . . .”
- “Take discovery on Linus Torvalds, the purported creator of Linux, about the contributors and contributions to Linux since its inception, and the maintenance of any records about the development history of Linux. Mr. Torvalds is expected to have detailed records of these contributors and their contributions, material that is not publicly available. Further, Mr. Torvalds can answer specific questions as to what each contributor intended, and where and how the contributor acquired or developed the derived code.”
- “Take discovery on the maintainers of the kernels. . . .”
- “There are many contributors to the kernels, some of who [sic] have significant contributions to Linux code over the years. Some of these individuals, whose names are publicly available, should be deposed to find out their sources for their contributed code.”
- “Many corporations have made contributions to Linux, and SCO needs to take their discovery on certain of these companies to determine the sources of their contributions. Also, SCO needs to depose the programmers who work for these companies and made the contributions to determine the sources of those programmers’ code contributions....”
- “SCO has identified some, but not all, independent authors of various portions of the Linux code. ... Those authors should know the sources of their code and should be able to provide information as to whether the code they contributed to Linux was obtained from SCO copyrighted code.”
- “Several private groups also made major contributions to Linux, so SCO should also be permitted adequate time to identify and take discovery from these entities.”
- “Many organizations exist whose purpose is to track and report on changes to Linux... SCO needs access to the more detailed information these organizations maintain. . . .”

- “Licensees and former licensees of Unix source code to see if these entities, their employees, or former employees are contributing Unix code to Linux.”

41. This list, too, is striking, for a number of reasons. First, while the information requested from IBM would make SCO’s task many times larger than it is, the request for this third-party information would surely magnify SCO’s task still more. Consider, for example, just the last item, requesting materials from “Licensees and former licensees of Unix source code to see if these entities, their employees, or former employees are contributing Unix code to Linux.” (¶ 57.)

42. Second, Mr. Sontag provides vanishingly little rationale for this voluminous request, which is not surprising, as the requested information is irrelevant to the task at hand. Once again, the task at hand is finding substantial similarity between Unix and Linux as it is now. Gathering information regarding the entire development history of Linux, including from potentially hundreds or even thousands of individuals, would not merely require a considerable amount of time, it would be of little or no meaningful assistance. The notion, for example, that “Mr. Torvalds can answer specific questions as to what each contributor intended, and where and how the contributor acquired or developed the derived code,” suggests a wholly unrealistic picture of any mortal and of the code development process. The task would be done far faster, and the time better spent, if SCO were simply to put even part of the effort imagined by Mr. Sontag to the task of comparing the Unix and Linux source code SCO already has.

43. Finally, the vast bulk of the information Mr. Sontag lists – and far more information than is necessary to determine substantial similarity – has long been accessible to SCO. Table 1 below lists a collection of web sites with archives of Linux code, mailing lists

maintained by kernel authors (indexed by contributor), etc., that covers much of what has been requested and certainly more than SCO could possible need.

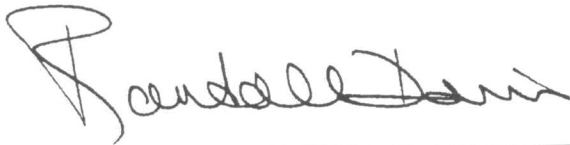
- www.linuxhq.org is an enormous collection of historical material, including a list of anyone who contributed code to the Linux project back to February 1996, indicating the general area they were involved in.
- www.linuxhq.com has *every patch* or other contribution to the code over the entire development history. See Exhibit II for one example.
- <http://www.tux.org/lkml/>, the Linux kernel mailing list. which includes pointers to a variety of other useful sources
- <http://www.uwsg.indiana.edu/hypermail/linux/kernel/>, which has a search by word/subject capability.
- <http://marc.theaimsgroup.com/?l=linux-kernel>, which keeps a collection of Linux-related list archives.
- <http://groups.google.com/groups?hl=en&q=fa.linux.kernel&meta=>, which is the Google interface to the fa.linux.kernel newsgroup
- <http://gossamer-threads.com/lists/linux/kernel/>, which has an easy-to-use interface for searching the large accumulation of messages
- <http://www.kerneltraffic.org/> provides a weekly summary of the discussions about the Linux kernel and archives previous summaries.

Table 1: List of Readily Available Sources of Information on Linux

V. SUMMARY

44. Mr. Sontag grossly exaggerates what is required to determine whether there is substantial similarity between Linux and SCO's allegedly copyrighted works. The materials necessary to the task have been available to SCO for years and tools capable of evaluating that material in a matter of months have also been available to SCO for years.

45. I declare under penalty of perjury that the foregoing is true and correct.

A handwritten signature in black ink, appearing to read "Randall Davis", written over a horizontal line.

Randall Davis

Date: 23 August 2004

Place: Weston, Massachusetts

VI. REFERENCES AND MATERIALS CONSIDERED

References

- [1] Eric S. Raymond, Resource page for COMPARATOR 2.0.
<http://www.catb.org/~esr/comparator/>
- [2] Dick Grune, The software and text similarity tester SIM.
<http://www.cs.vu.nl/~dick/sim.html>

Materials Considered

Computer Associates v. Altai, 982 F.2d 693 (2d Cir. 1992)

Gates Rubber, Inc., v. Bando American, Inc., 9 F.3d 823 (10th Cir. 1993)

Mitel, Inc. v. Iqtel, Inc. 124 F.3d 1366 (10th Cir. 1997)

References listed above

Web sites listed in Table 1 above

Declaration of Chris Sontag in Support of SCO's Opposition to IBM's Motion for Partial Summary Judgment, undated