**UNITED STATES DISTRICT COURT**
**DISTRICT OF MASSACHUSETTS**
**EASTERN DIVISION**

RED BEND LTD., and
RED BEND SOFTWARE INC.,

Plaintiffs,

v.

GOOGLE INC.,

Defendant.

Civil Action No. 09-cv-11813-DPW

# DECLARATION OF STEPHEN A. EDWARDS IN SUPPORT OF PLAINTIFFS' REPLY CLAIM CONSTRUCTION BRIEF

I, Stephen A. Edwards, declare as follows:

## I. Introduction

1. I make this declaration in support of Red Bend Ltd.'s and Red Bend Software Inc.'s (collectively "Red Bend") reply claim construction brief.

2. The statements made herein supplement my three prior declarations dated November 17, 2009, March 24, 2010, and July 15, 2010 and address issues raised by Google's Claim Construction brief and the July 15, 2010 expert declaration of Martin Walker.

## II. Comparing the Quality of Diff Utilities

3. Diff utilities may perform differently depending on their design. For example, developers may make trade offs between matching-accuracy, computational time, and the size of the difference results when creating a diff utility, which would cause one utility to identify fewer matches or more matches than another. For instance, the technique described in "A Cross-Platform Binary Diff" ignores matches of less than six bytes because: "Due to the file format used, matches of less than six bytes increase the size of the diff file instead of reducing it." (Exh A at 1, attached hereto). Accordingly,

Dockets.Justia.com

use of that diff utility may fail to detect as a match corresponding parts of two different versions of the same program, because the size of the matching entries was less than the threshold match size used by the utility.

4. The invention of the ‘552 patent is not dependent on the diff utility used, and is intended to improve the performance of any binary difference utility. (*See* Google Exh. E, S. Peleg Dep. at 35:25-36:15, 104:14-18).

5. The inventor made it clear (Red Bend Exh.[1] 1, ‘552 patent at 2:17-20) that “there [was] a need in the art to provide for an efficient tool which will result in significantly smaller volumes of difference results.” Thus, a central goal, reflected in his choice of the term “compact difference result,” was to make the difference result *smaller*, not necessarily that it *exclude* certain data. (*See,* Google Exh. C at RedBend150 (“To simply reflect all the changed references when computing a difference, one must include them all. In accordance with the present application such a need is reduced or eliminated.”)).

6. It follows that the construction of the term “compact difference result” should be based on the reduction of size of the difference result, according to Red Bend’s construction. Google’s construction, which requires that “references that have changed due to delete/insert modifications do not appear,” does not make sense at least because (1) it permits something else to take their place in the difference result; and (2) consequently, this does not necessarily reduce the size of the difference result. In other words, simply not appearing is not helpful unless the amount of data (in the difference result) is also reduced. (Google Br. at 14). For instance, simply adding the number one

[1] Citations to “Red Bend Exh. __” refer to Exhibits attached to the July 15, 2010 Declaration of Jennifer C. Tempesta in Support of Plaintiffs’ Opening Claim Construction Brief.

to the value of each byte of the new program (whether that byte includes an instruction or a reference) prior to generating the difference result would ensure that virtually no reference in the new program would appear in the difference result (because they all changed due to the modification made by adding the number one to each byte), but it would do nothing to reduce the size of the difference result (because under Google's construction all the changed bytes appear instead). No one of ordinary skill in the art would consider this a "compact difference result," yet it would be under Google's construction.

## <u>III. Google's Use of "Reference" and "Reference Entry"</u>

7. Google continues to be sloppy in their use of the terms "reference" and "reference entry." In explaining the example Google relies upon in its brief, it states "in the instruction 'move 11' at address 5 in the 'old program,' *11 is a reference entry*. After new instructions (shown in green) are inserted, this becomes 'move 13 in the 'new program' below, with *13 being the new reference entry*." (Google Br. at 5 (emphasis supplied)). The italicized statements are technically incorrect. The '552 patent glossary clearly defines "reference" and "reference entry," and the parties have agreed upon the meaning of those terms that closely tracks the language of the glossary. (Red Bend Exh. 1, '552 patent 2:42-47). Based on the agreed definitions of "reference" and "reference entry," in Google's example, "11" and "13" are *references*, because each is an address within an entry in a data table that is used to refer to another entry in the same data table. In contrast, entry 5 in the "old program" (*i.e.*, "move 11") and entry 7 in "new program" (*i.e.*, "move 13"), respectively, are *reference entries*, because both are addressable units of a data table or program that *include a reference*. In particular, in the entry "move 11,"

"11" is a *reference* to the entry stored at address 11 of the old program (*i.e.,* "data" in Google's example). Similarly, in the entry "move 13," "13" is a *reference* to the entry stored at address 13 of the new program (*i.e.,* "data").

8. Google's misunderstanding of the distinction between a "reference" and a "reference entry" causes it to come to a wrong understanding of the descriptions in the '552 specification in support of its proposed claim constructions. In particular, Google misinterprets the specification's statement that "*[t]he net effect is that the invariant reference entries . . . will not appear in the difference result*" as requiring the "exclu[sion] [of] all references that change only due to the ripple effect." (Google Br. at 6 *citing* '552 Patent 3:36-46 (emphasis in Google's quote)). However, with a proper understanding of the applicant's distinction between *references* and *reference entries*, it is clear that the statement quoted by Google is directed at invariant *reference entries*, and does not suggest exclusion of "all references that change." (*Id.*).

9. Merely "excluding" references (*i.e.*, so that they do not appear) from the difference result would not make it compact: the space that they would have otherwise occupied would have to be eliminated as well. In other words, you cannot just exclude references and replace them with other data -- you need to reduce the size of the difference result. Furthermore, if all information about the references that changed was eliminated from the difference result, it would be impossible to reconstitute the new program or data table on the client side, which must include the new reference values. (*See* Red Bend Exh. 1, '552 Patent 11:35-12:10, 13:21-32). What the inventor explains in the specification sections cited by Google (*see also*, Red Bend Exh. 1, '552 Patent 10:3-15) is that rendering references that changed due to delete/insert modifications invariant effectively hides the changes from the difference utility, making the output more compact.

## IV. The Okuzumi Reference

10. The Okuzumi reference is directed at recognizing differences between two source programs. (*See* Google Exh.[2] G at RedBend5434). A source program (or source code) is commonly stored as a text file.

11. There are important differences between algorithms that perform diffs on text files and those that work on "binary" data. For example, the former usually takes advantage of line delimiters that are commonly found in textual data that is intended to be human-readable, such as source code. Binary comparison tools cannot assume the presence of and therefore do not take advantage of such delimiters, and therefore usually search for matches on a byte-by-byte basis. This distinction is explained further in "A Cross-Platform Binary Diff," one of the prior art references considered by the patent office during prosecution of the '552 Patent. (See Exh. A). As explained therein:

> Many diff utilities only work with pure text files because they depend heavily on the concept of a 'line' denoted by some kind of delimiter. Dividing files into lines lets you index the lines in one file, then use the index to find matches with lines in another file. A binary file, however, has no such delimiter, making it harder to index.

(Id. at 1).

12. It is worth mentioning that the distinction between a "text file" and a "binary file" in this context refers to how the contents of the file are meant to be interpreted -- not the physical format in which the file is stored or transmitted. Nearly all computing devices encode and transmit data (whether the data represents "text" or "binary" information) in binary format due to the physical nature of computing devices, which consist of storage and processing circuits that are designed to store and process

[2] Citations to "Google Exh. __" refer to the Exhibits attached to the July 15, 2010 Declaration of Susan Baker Manning in Support of Google Inc.'s Claim Construction Brief.

only digitized, binary data. When the logical content of a computer file is textual, the text is typically converted to a binary format. There are several well-known ways to represent text in binary format for this purpose, including ASCII and Unicode. (*See, e.g.,* Exh. B, attached hereto, *Microsoft Press Computer Dictionary* (2nd ed. 1994) at 27 ("ASCII")).

13. During prosecution, the applicant distinguished Okuzumi from the executable program claims of the '552 patent, the only claims that require an executable program as input, on the basis that Okuzumi operates on source programs. (Google Exh. C at RedBend149-152).

## V. Claim Construction

14. Dr. Walker asserts that the "data table" of the claims cannot include source or other symbolic code, citing *part of* a statement made by the applicant to distinguish the executable program of claim 1 from the source programs processed by Okuzumi. (*See,* Docket No. 95, Walker Decl ¶¶22-24 (citing RedBend151 "In extracting a diff between 2 versions of executable files as defined in Claim 1, there is no source involved, and neither statements, nor any textual or other symbolic representation of the program even exist"). This statement is clearly limited to the executable programs of claim 1. Even if the statement applied to all claims, which it clearly does not, it still would not support Dr. Walker's position. The applicant stated that no "textual or symbolic representation *of the program* even exist." (emphasis added). Thus, the applicant was pointing out that the claimed techniques work without requiring access to a symbolic representation of the *entire* program, which clearly distinguished the invention over Okuzumi, which works only when *the entire program* is represented symbolically.

Therefore the excerpt of the prosecution history cited by Dr. Walker would suggest to one skilled in the art that the applicant's claims would permit representations of a program or data table that included *some* source or symbolic code.

15. "Data table" by definition is broader than "executable program" because an executable program is only one example of a data table. (Red Bend Exh. 1, '552 patent 2:33-34; 2:61-63).

16. Google and Dr. Walker assert that the modified old/new data tables/programs must be in executable format and cite, as support, the applicant's statement discussed above in paragraph 14. (Google Br. at 19; Docket No. 95, Walker decl. ¶¶25-27). Their reasoning and interpretation contradicts the explicit definition of "data table" and is further incorrect for the same reasons I state above.

17. In the generation of the modified old/new data tables/programs, the asserted independent generation-claims recite that "substantially each reference in an entry . . . *are reflected as invariant references*." (*See, e.g.,* Red Bend Exh. 1, '552 patent claim 8). The unasserted independent generation-claims recite "replacing the reference of said entry by a distinct label mark." (*See, e.g.,* Red Bend Exh. 1, '552 patent claim 1). One of ordinary skill in the art would recognize that "reflected as invariant" is different from "replacing the reference of said entry by a distinct label mark," the former being much broader than the latter. In particular, "reflected as invariant" just means that other parts of the claimed technique, such as the difference utility, would interpret the reference in the entry as being invariant. There is no requirement to make modifications to a program in place to practice the asserted claims (8-12, 21-25, 28-34, 42-46, 55-60, 62-67).

18. Google and Dr. Walker, relying on the prosecution history, attempt to limit "modifications to replacement of 'the reference of said entry by distinct label mark' and not 'the generation of auxiliary index data structure.'" (Google Br. at 20; Docket No. 95, Walker decl. ¶26 (both citing RedBend173)). However, if considered in context, it is clear that the applicant was distinguishing the *optional* auxiliary index of Miller, from the *obligatory* creation of modified old programs/data tables in the '552 patent. (*See* Google Exh. C at RedBend172-3). Miller's optional index is used to decrease search time for text strings in the old file. (*See* Google Exh. C at RedBend173). Performing Miller's process without creating and using the optional index would simply cause the process to take longer. (*Id.*). In contrast, without generating the modified old program/data table of the '552 patent, the compact difference result could not be generated. Moreover, the Examiner ultimately disagreed with the applicant's characterization of Miller, further confirming that a person of ordinary skill in the art would not rely on this section of the prosecution history to support Google's attempt to narrow the claims.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct and that this declaration was executed on July 29, 2010 at Cambridge, UK.

By: [illegible]

Stephen A. Edwards