

EXHIBIT 1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Reexamination of)	MAIL STOP: <i>Ex Parte Reexamination</i>
U.S. Patent No.: 6,546,552)	Group Art Unit: 3992
Sharon Peleg)	Examiner: Andrew L. Nalven
Issued: April 8, 2003)	Confirmation No.: 4316
Reexam Control No.: 90/009,670)	
For: DIFFERENCE EXTRACTION BETWEEN)	
TWO VERSIONS OF DATA-TABLES)	
CONTAINING INTRA-REFERENCES)	

SUPPLEMENTAL RESPONSE

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to a telephone call from the Reexamination Unit at the U.S. Patent and Trademark Office, the Patent Owner ("Owner") files a Supplemental Response relating to the Response filed in the U.S. Patent and Trademark Office on June 21, 2010, in the above-identified reexamination proceeding. Specifically, after filing the initial Response, three oversights were noted in a Letter Regarding Response filed on June 22, 2010. However, the U.S. Patent and Trademark Office Reexamination Unit has requested that this Supplemental Response be filed incorporating the changes mentioned in the Letter Regarding Response. Other than the changes noted in the Letter Regarding Response, no other substantive changes have been made to the Response of June 21, 2010, reproduced below.

This document is in response to the Office Action dated May 28, 2010.
Reconsideration is respectfully requested.

The Owner thanks the Examiner for courtesies extended to the undersigned during the telephone call of June 17, 2010. It is the undersigned's understanding from this telephone call that if the Examiner is not fully persuaded by this paper to confirm all the remaining rejected claims of the above-captioned patent, he will contact the undersigned within a few weeks of this submission to arrange an interview. On this basis, the Owner will forego the interview previously scheduled for June 25, 2010.

The Owner also thanks the Examiner for the indication that claims 5-7, 12-13, 18-20, 25-26, 29-34, 39-41, 46-47, 52-54, 59-60 and 63-68 (the "Confirmed Claims") are patentable and/or confirmed. The Owner submits the following remarks and arguments regarding the rejections of the remaining claims.

TABLE OF CONTENTS

1. REMARKS/ARGUMENT	3
A. The Pending Rejections.....	3
B. The Wetmore Reference	3
C. The '552 Patent.....	7
D. Responses to Rejections.....	7
i. The Invariant Reference Claims	7
(1) The Inputs to Wetmore Are Not Executable.....	8
(2) No "References" In Wetmore Are Reflected As Invariant.....	10
(3) Wetmore Fails To Disclose Reflecting "Substantially Each" Reference Invariant.....	12
(4) Wetmore Fails To Disclose Generation of A "Compact Difference Result"	13
ii. The Distinct Label Mark Claims	15
(1) The Inputs to Wetmore Are Not Executable.....	15
(2) No "References" In Wetmore Are Replaced By a Distinct Label Mark... ..	15
(3) Wetmore Fails To Disclose "For Substantially Each Reference Entry . . . Replacing The Reference . . . By a Distinct Label Mark"	16
(4) Wetmore Fails To Disclose Generation of A "Compact Difference Result"	17
iii. The Rejections Under 35 U.S.C. § 103.....	17
E. Comments on Statement of Reasons For Patentability And/Or Confirmation.....	19
F. Conclusion	19

1. REMARKS/ARGUMENT

A. The Pending Rejections

Claims 1-4, 8-11, 14-17, 21-24, 27, 28, 35-38, 42-45, 48-51, 55-58, 61 and 62 stand rejected (the "Rejected Claims") under 35 U.S.C. §102(b) as allegedly being anticipated by U.S. Patent No. 5,481,713 issued to Wetmore et al. (hereinafter "Wetmore"). This rejection is respectfully traversed.

As an initial matter, the Office Action is slightly ambiguous regarding the basis for the rejection of claims 2, 3, 9, 10, 15, 16, 22, 23, 36, 37, 43, 44, 49, 50, 56 and 57. On page 5 of the Office Action, those claims are listed as being rejected over Wetmore. However, the Office has not identified any disclosure of Wetmore that teaches the relevant limitations. Instead, it appears that the Office intended to reject those claims only under 35 U.S.C. § 103, for the reasons stated on page 18 of the Office Action. Accordingly, this response treats claims 2, 3, 9, 10, 15, 16, 22, 23, 36, 37, 43, 44, 49, 50, 56, and 57 as rejected only under § 103.

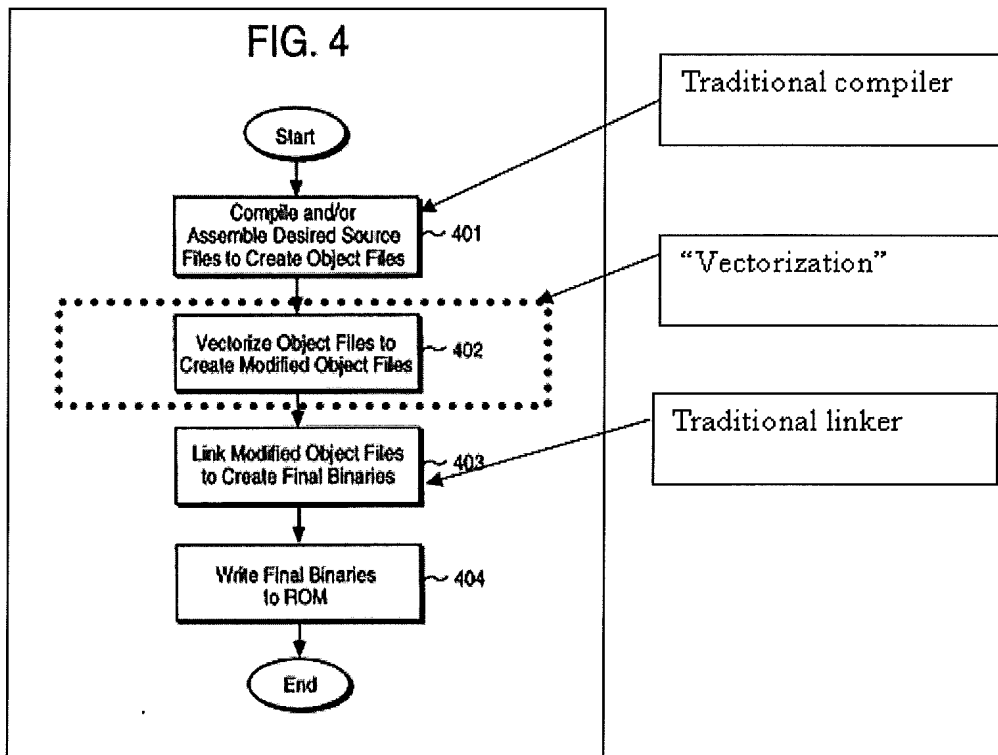
B. The Wetmore Reference

The Office has applied Wetmore either as an anticipatory or primary reference against all Rejected Claims. Wetmore, however, is directed to distribution of "Vectorized" software that is to be installed into a Read Only Memory ("ROM") at an end-user's computer, as was commonly done with the Operating System of early Macintosh computers sold by Apple -- Wetmore's assignee. (Wetmore 1:29-2:45). Vectorization of software is complex, and involves modifying the developer's software development tool-chain (e.g. compiler and linker) by adding a vectorization step in the middle of software development to create specialized binary executables for ROM:

... First, the source files are compiled ... to create object files, step 401. The object files are then vectorized to create vectorized object files, step 402. It is significant that only the object files are modified. ... Object files contain a series of defined records, each one containing specific items such as the

object code for a routine, the name of a routine, external reference from one routine to another, or comments. In object files the references to other routines have not been resolved. Therefore object files are an ideal place to alter the code without modifying the source code files. ... *The object files are then linked together to create the final binary values which will be written to ROM, step 403. This is performed through a traditional linkage editing step. Finally, after the object files have been "linked" together to create the final binaries, the ROM image is created, step 404.*

(Wetmore 6:47-67) (emphasis supplied). In other words, Wetmore requires introduction of a vectorization step (402) in the middle of the compilation chain, as shown in Wetmore's Fig. 4 (annotated below):

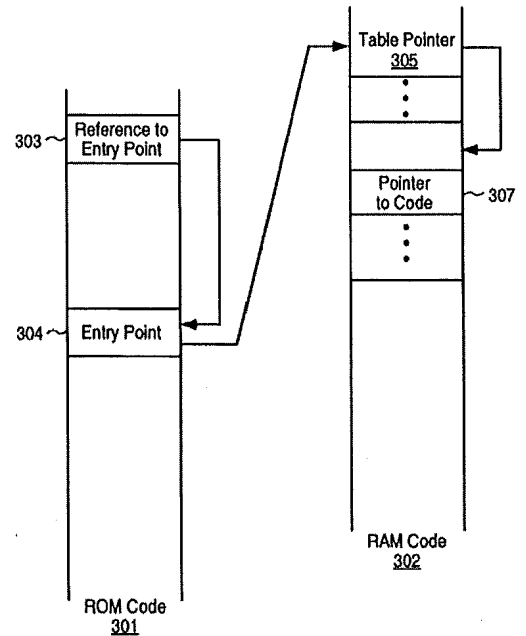


By vectorizing the software, the resulting binary file takes on different characteristics than it would have had in the absence of vectorizing. In particular, as described in Wetmore, "more entry points into the ROM are created, thus providing more locations at which the ROM may be accessed and code fixed." (Wetmore 6:8-11). Wetmore describes the "effect of vectorization" with reference to Figure 3:

"illustrated is ROM based code 301 and RAM code 302 . . . ROM code 301 will contain a reference to an entry point 303. The entry point may be a sub-routine, function, macro or a jump to a label somewhere else in the ROM code 301. . . . The reference to entry point 303 will effectively point to entry point 304. *Without vectorization, the executable code would be immediately following the entry point 304. With vectorization, the location of entry point 304 has been modified to be a reference to a table pointer 305 that resides in RAM.* The table pointer 305 is the vector in this example. The *table pointer 305 will point to vector table 306 which resides in RAM code 302* (specifically the system heap area).

As noted above, the reference to the table would in most cases include an offset into the vector table 306. Assuming the offset, *the entry 307 will contain a pointer to the location where the code to be executed would reside.* While the entry 307 may simply point back into the ROM, in the instance of a patch, the entry 307 may contain a pointer to an updated routine located somewhere in the RAM 302." (Wetmore 5:32-57) (emphasis supplied).

FIG. 3



Thus, Wetmore describes vectorization as a process of modifying the compilation and linking process in order to introduce a level of indirection to the resulting binary file so that when entering an external routine such as a subroutine or function call (5:27-32), program execution does not flow sequentially -- but is first redirected to one of several vector tables in RAM which will contain the address of the next line of code that should be executed.

Also noteworthy in Wetmore, the vectorization process operates only on "entry points" (e.g., 304) which it describes as locations in code that may be entered via a symbolic

reference. (Wetmore 7:1-8). Accordingly, the “entry points” that are vectorized are names or labels in the object files. *Id.* Indeed, Wetmore operates only on object files, which it describes as the “ideal place to alter the code” because in “object files the references to other routines have not been resolved” (Wetmore 6:57-60), meaning that they have not yet been linked, are symbolic (rather than numeric references to memory addresses), and are not yet executable.

Using Wetmore’s technique, the binary code that performs the subroutines or function calls can be replaced (or new subroutines or function calls can be added) in a subsequent version of the binary file by sending to the user in a system disk the new code for those subroutines or function calls. (Wetmore 10:24-53; 11:45-48). Significantly, therefore, Wetmore does not utilize a binary difference result algorithm, such as those prior art techniques discussed in the specification of the ‘552 Patent, to prepare a difference result. (‘552 Patent at 1:50-2-12. See also Coppieters, K. “A Cross-Platform Binary Diff” of record in the ‘552 patent). Instead, Wetmore describes a far less granular comparison of the “object files of two versions of the vectorized ROM code to *identify routines* which are different or new” (Wetmore 11:1-4) (emphasis supplied), which new routines are then wholly transferred to the client. So, if even one line of code were changed in a routine, the entire routine would be identified for transfer to the client, where the new version of the routine would be loaded into RAM for subsequent execution in place of the obsolete version of the routine in ROM (Wetmore 2:4-5; 5:51-57; 10:35-53), and the vector tables in RAM would be modified to point to the updated location of the new version. (Wetmore 11:35-67).

In other words, Wetmore permits updates of software by transmitting entire routines to the client, which are executed instead of obsolete routines in ROM. Notably, this would result in the transmission of *large* difference results, because if even a single line of binary code were added to a routine, the *entire* routine would need to be included in Wetmore’s difference result. This would potentially result in a larger difference result than even the prior art difference techniques, such as those described in Coppieters.

C. The '552 Patent

In marked contrast, the techniques of the '552 Patent are substantially different and are directed towards a substantially different problem than Wetmore. The '552 Patent is directed to a process for creating or applying a compact difference result as between two executable programs (claim 1-34) or two data tables (claims 35-68), involving the use of modifications performed on the old and new programs/data tables where the modifications include the use of distinct label marks (claims 1-7, 13-20, 26-27, 35-41, 47-54, 61) and/or invariant references (8-12, 21-25, 28-34, 42-46, 55-60, 62-68) for the purpose of distributing a *compact* difference result. The '552 Patent is not concerned with updating programs previously stored in a Read-Only Memory, and does not require changes to the developer's compilation tool-chain in order to function. Indeed, the techniques of the '552 Patent can be applied to an arbitrary executable program or data table, and it is not necessary to have access to the source code or pre-linked object files of the old or new programs/data tables being updated. Further, the difference result generated by the techniques of the '552 Patent is compact, *i.e.* smaller than traditional binary difference results, and unlike Wetmore, would not re-transmit an entire routine to the client when only a single instruction in that routine was changed.

D. Responses to Rejections

As discussed herein with respect to the claims under reexamination, these important differences between the disclosure of Wetmore and the '552 Patent show that Wetmore does not anticipate or render obvious any of the Rejected Claims.

i. The Invariant Reference Claims

The Office has rejected Claims 8, 11, 21, 24, and 28 (hereinafter the "Executable Program Invariant Reference Claims") and Claims 42, 45, 55, 58, and 62 (hereinafter the "Data Table Invariant Reference Claims") (collectively, the "Executable Program Invariant Reference Claims" and the "Data Table Invariant Reference Claims" are referred to herein

as the “Invariant Reference Claims”) under 35 U.S.C. § 102(b) over Wetmore. The Owner respectfully traverses.

Exemplary claim 8 is set forth below:

8. A method for generating a compact difference result between an *old executable program and a new executable program*; each program including reference entries that contain *reference that refer to other entries in the program*; the method comprising the steps of:

(a) generating a modified old program *utilizing at least said old program*;

(b) generating a modified new program *utilizing at least said new program*,

said modified old program and modified new program have at least the following characteristics:

(i) substantially each reference in an entry in said old program that is different than corresponding entry in said new program due to delete/insert modifications that form part of the transition between said old program and new program are reflected as invariant references in the corresponding entries in said modified old and modified new programs;

(c) generating said compact difference result utilizing at least said modified new program and modified old program.

(1) The Inputs to Wetmore Are Not Executable

Regarding the Executable Program Invariant Reference Claims, Wetmore fails to disclose several claim elements. In particular, these claims require that old and new *executable* programs be input into the method (claims 8, 11) or system (21, 24) and that those old and new executable programs be used to generate a “modified old program” and a “modified new program” such that certain references (*i.e.* substantially each reference in an entry in one program that is different than a corresponding entry in the other program due to

insert/deletes that occurred in the transition between the two programs) are reflected as invariant references in the modified forms. (This, of course, does not mean that the modified programs need to themselves be in executable form, but only that the inputs or starting old and new programs that are to be modified, are executable. The preamble and claim text make this clear.)

Significantly, the claimed executable programs used as inputs to the difference result generation process are not disclosed in Wetmore. It appears that the Office has considered the "object files" of Wetmore to correspond to the claimed old and new *executable* programs. (Office Action at 7)¹. Owner respectfully disagrees. The object files of Wetmore that are vectorized are not executable. (Wetmore Fig. 4; 6:45-67). See *supra* at Section 1(B), discussing Fig. 4 and Wetmore's vectorization process. Instead, those files include unresolved, symbolic references. (Wetmore 6:53-60) ("In object files the references to other routines have not been resolved"). Such a file would not become an executable binary until *after* linking, which is the next step in the Wetmore process, after vectorization. (Wetmore 6:63-67) ("The object files are *then* linked together to create the final binary values which will be written to ROM") (emphasis supplied).² Note that the process of Wetmore *only* works when the unlinked object files (containing symbolic references) are accessible. Wetmore repeatedly makes this point clear:

- "It is significant that only the object files are modified" (Wetmore 6:52-53)
- "object files are an ideal place to alter the code" (Wetmore 6:59-60)
- "An entry point may be the *name* of a routine or a *label* in the file. Generally, an entry point is merely a location in the code which may be entered via a *symbolic reference*. It is these entry points which become the code access points which are vectorized" (Wetmore 7:4-8) (emphasis supplied)

¹ To the extent the old and new linked, vectorized ROM images are viewed as the old or new executable programs, respectively, Wetmore would not anticipate the claims because (under that reading) it fails to disclose generations of a modified old or modified new program utilizing those old (or new) executable programs.

² This is not meant to suggest that the claimed executable files must exclude *all* symbolic information. To the contrary, as is well-known in the art, executable files often contain header and other symbolic information to assist the loader and operating system with loading the program image into machine memory and commencing execution, as well as for other purposes. However, the Executable Program Invariant Reference Claims cover program files that are ready to be run.

Exemplary techniques of the Executable Program Invariant Reference Claims, by contrast, can be designed to work on input files that are already executable, and thus have the advantage of being able to be employed without access to source code or unlinked object files and, as discussed previously, can be employed without modifying the developer's development tool-chain (*i.e.* the compilation and linking process used by the developer to translate source code into an executable binary file).

Therefore, Wetmore does not disclose this element of the Executable Program Invariant Reference Claims, and, consequently, at least the Executable Program Invariant Reference Claims (Claims 8, 11, 21, 24, and 28) and the claims dependent therefrom (claims 9, 10, 22 and 23) are not anticipated by or rendered obvious over Wetmore.

(2) No "References" In Wetmore Are Reflected As Invariant

Regarding both the Executable Program Invariant Reference Claims (Claims 8, 11, 21, 24, and 28) and the Data Table Invariant Reference Claims (Claims 42, 45, 55, 58, and 62) -- because Wetmore only operates on object files, it also fails to disclose in its inputs or starting program any "references," as defined by the '552 Patent and as recited in this claim limitation:

substantially each *reference in an entry in said old program* that is different than corresponding entry in said new program due to delete/insert modifications that form part of the transition between said old program and new program are reflected as invariant references in the corresponding entries in said modified old and modified new programs... (emphasis added).

It appears that the Office treated Wetmore's "external references to other routines" in the object files as the claimed "references" of the '552 Patent. (Office Action at 7-8). The Owner respectfully disagrees with this characterization. Importantly, the Glossary of '552 Patent defines "references" as follows:

Reference -- a part of the data appearing in an entry in the data table which is used to refer to some other entry from the same data table. A

reference can be either an address or a number used to compute an address.

Address -- a *number* which is uniquely assigned to a single entry by which that entry is accessed...

(‘552 Patent 2:37-45)(emphasis supplied). Accordingly, to qualify as a “reference” in the Invariant Reference Claims, the reference must be either an “address” (which is further defined to be a “number”) or a “number used to compute an address” (such as an offset). In either case, the reference must be a “number.”³ Such references are not described as being vectorized in Wetmore.

The references to external routines in Wetmore, which the Office has identified as allegedly corresponding to the claimed references (Office Action at 7), are not numeric. Instead, they are purely symbolic. See supra at Section 1(D)(i)(1), (citing Wetmore 6:52-53; 6:59-60; 7:4-8)(“An entry point may be the *name* of a routine or a *label* in the file. Generally, an entry point is merely a location in the code which may be entered via a *symbolic* reference. It is these entry points which become the code access points which are vectorized”). Wetmore describes creating numeric (as opposed to symbolic) references in the vector table object file, which is the *output of vectorization*. (Wetmore 8:21-52) (showing introduction of the entry: “jmp ([\$0584])” after vectorization took place). Accordingly, there are no “references” as claimed in the ‘552 Patent that are disclosed as being *input* to a vectorization process in Wetmore, because the numeric addresses that Wetmore describes are not present in the files input to Wetmore’s vectorization process. Therefore, Wetmore fails to disclose that “substantially each reference in an entry in said old program ... are reflected as invariant.”

This significant difference between Wetmore and the ‘552 Patent is a result of Wetmore’s being limited to operating on object files, which is not a restriction on the claimed

³ This is not meant to suggest that “invariant references” must be numeric. To the contrary, the specification makes clear that invariant references may be numeric or symbolic label marks. (‘552 Patent 10:47-60). It is only the references in the old and new executable programs or data tables that must be numeric.

techniques. Instead, the claimed techniques are intended to operate on files *after* references have been resolved to become numeric, as opposed to symbolic -- thereby permitting the techniques of the '552 Patent to be applied to executable files and data tables. In this manner, the techniques of the '552 Patent have utility in many contexts where Wetmore's techniques could not be used. Accordingly, for this reason alone, none of the Invariant Reference Claims (Claims 8, 11, 21, 24, 28, 42, 45, 55, 58 and 62, and therefore dependent claims 9, 10, 22, 23, 43, 44, 56 and 57) are anticipated by or rendered obvious over Wetmore.

**(3) Wetmore Fails To Disclose Reflecting
"Substantially Each" Reference Invariant**

Regarding both the Executable Program Invariant Reference Claims and the Data Table Invariant Reference Claims, Wetmore fails to disclose reflecting as invariant "substantially each reference" that has been altered as a result of delete/insert modifications between the old and new programs/data tables. Instead, Wetmore describes vectorizing *only* "entry points" to routines, rather than substantially all references that are likely to have changed due to delete/insert modifications. (Wetmore 7:1-9; *supra* at Section 1(D)(i)(2)). Assuming for the sake of argument that non-numeric labels in object files could qualify as "references," Wetmore still fails to teach, suggest or provide reason for a modification that would meet the claimed invention.

In particular, Wetmore does not teach, disclose or even suggest reflecting as invariant those references *within* routines that have changed due to delete/insert modifications (*i.e.* the non-"entry point" references). This missing disclosure is not surprising, because Wetmore's updates operate at the granularity of the routine itself. (Wetmore 5:22-31; 7:5-9) ("entry points for *external routines* are replaced"; "upon encountering a *reference to an external routine*, e.g. a subroutine or function call, the actual entry point will reference the vector table..."). As previously discussed, if any part of a routine is changed, Wetmore transmits the *entire routine* to the client as part of the patch. *See supra* at Section 1(D)(i)(4). Therefore, Wetmore does not bother making modifications

that would cause references *within* the changed routines to be reflected as invariant. This is a significant departure from the Invariant Reference Claims of the '552 Patent, which attempt to generate modified old and new programs such that "substantially each reference" that has been changed as a result of a delete/insert modification -- including the numeric references within routines that have changed due to delete/insert modifications -- are reflected as invariant. Accordingly, the Owner respectfully submits that at least the Invariant Reference Claims are not anticipated by or obvious over Wetmore.

**(4) Wetmore Fails To Disclose Generation of A
"Compact Difference Result"**

Regarding both the Executable Program Invariant Reference Claims and the Data Table Invariant Reference Claims (Claims 8, 11, 21, 24, 28, 42, 45, 55, 58 and 62, and therefore dependent claims 9, 10, 22, 23, 43, 44, 56 and 57), Wetmore fails to disclose generation of a "compact difference result." As the specification makes clear, the techniques of the '552 Patent are directed toward improving prior art "difference results" so as to generate a smaller (i.e. *compact*) difference result" compared to prior binary comparison tools, such as that described in the Coppieters reference. ('552 Patent, 3:30-46; 14:5-14). Those prior techniques looked for all "matches" between two binary files, and represented the matches as a series of references to portions of the old file where the matching portions were located. When there were portions in the new file that did not match a portion of the old file, the new portion would be included in the diff file itself. (See Coppieters at 35 (describing diff file format)). Accordingly, to minimize the size of the diff file, the prior art techniques focused on finding the largest number of matches between the old and new file as possible, because the failure to find a match meant that the size of the diff file would be increased as a result of having to include the non-matching code and/or data within the diff file.

Notably, to reduce the diff size, Coppieters recommended treating as a match "chunks" of "six or more bytes" that are found in both the old and new files. (Coppieters at 32). The result of this technique, therefore, would not include in the diff any chunks of six or

more bytes that were the same in both the new file and the old file, such as chunks of code and/or data that did not change between the two versions. Instead, as mentioned above, those chunks would be identified in the diff by specifying their location and size in the old file, so that the client could locate those chunks in the old file and make use of them when applying the “diff” to create the updated program. In addition, the diff file would include chunks that did not match (*i.e.* new code/data as well as matching chunks of less than six bytes).

These prior art techniques, however, were not well-suited for comparing program files or data tables containing references that are numbers. In particular, those prior techniques would fail to detect a match between portions of code in the old and new files when those portions included references that had changed between versions solely due to delete/insert modifications made to other portions of the file. As explained in the ‘552 Patent: “insertion of only one new entry may result in [a] plurality of altered reference entries which will naturally be reflected in the difference result and obviously will inflate its volume.” (‘552 Patent 2:6-9).

Wetmore, in contrast, generates difference results that would be significantly larger than those created by even the Coppieters-type binary difference techniques. In particular, Wetmore describes a far less granular comparison of the “object files of two versions of the vectorized ROM code *to identify routines* which are different or new.” (Wetmore 11:1-4) (Emphasis supplied). If *any* part of the routine does not match, *the entire new routine is then transferred* to the client, where the new version would be loaded into RAM for subsequent execution in place of the obsolete version in ROM (Wetmore 2:4-5; 5:51-57; 10:35-53; 11:35-67). In other words, Wetmore results in the transmission of large difference results, because if even a single line of binary code were added to a routine, the *entire* routine would need to be included in Wetmore’s difference result. Accordingly, Wetmore does not teach, suggest or provide reason for a modification that would meet a *compact* difference result,

which would entail generation of a difference result even *smaller* than those computed by prior art techniques, such as Coppieters.

For at least the foregoing reasons, Owner respectfully traverses the rejections of the Executable Program Invariant Reference Claims (*i.e.* claims 8, 11, 21, 24, and 28) and the Data Table Invariant Reference Claims (*i.e.* claims 42, 45, 55, 58, and 62 and therefore dependent claims 9, 10, 22, 23, 43, 44, 56 and 57) under 35 U.S.C. § 102(b).

ii. The Distinct Label Mark Claims

The Office has rejected Claims 1, 4, 14, 17, and 26 (hereinafter the “Executable Program Distinct Label Mark Claims”) and Claims 35, 38, 48, 51, and 61 (hereinafter the “Data Table Distinct Label Mark Claims”) (collectively, the “Executable Program Distinct Label Mark Claims” and the “Data Table Distinct Label Mark Claims” are referred to herein as the “Distinct Label Mark Claims”) under 35 U.S.C. § 102(b) over Wetmore. The Owner respectfully traverses.

(1) The Inputs to Wetmore Are Not Executable

As to the Executable Program Distinct Label Mark Claims, each of these claims requires scanning an old and new program, both of which are *executable*. (*See e.g.* ‘552 Patent claim 1, preamble). As discussed above with respect to the Executable Program Invariant Reference Claims, Wetmore fails to disclose this limitation because the files that Wetmore operates on as part of its vectorization process are object files that contain unresolved references, prior to linking, and are not executable. *See, supra* pp. 8-10.

(2) No “References” In Wetmore Are Replaced By a Distinct Label Mark

Regarding both the Executable Program Distinct Label Mark Claims and the Data Table Distinct Label Mark Claims, Wetmore also fails to disclose a modification that replaces any “reference,” as defined in the Glossary of the ‘552 Patent and recited in this claim limitation, for example:

(a) scanning the old program and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark...

As discussed previously (see *supra* at Section 1(D)(i)(2)), the “references” of the ‘552 Patent that are initially processed or input as part of the claimed techniques are numeric. In contrast, the “references” of Wetmore that are vectorized are purely symbolic, because they are unresolved references in object files. Accordingly, Wetmore fails to teach, suggest or provide reason for a modification that would meet all elements of these claims.

(3) Wetmore Fails To Disclose “For Substantially Each Reference Entry . . . Replacing The Reference . . . By a Distinct Label Mark”

Regarding both the Executable Program Distinct Label Mark Claims and the Data Table Distinct Label Mark Claims, Wetmore fails to disclose replacing the reference by a distinct label mark for “substantially each reference entry.” (See *e.g.* 552 Patent, claim 1). Instead, Wetmore describes vectorizing *only* “entry points” to routines (Wetmore 7:1-9; *supra* at Section 1(D)(i)(2)), rather than the *references*. (‘552 Patent, claim 1). Assuming for the sake of argument that non-numeric labels in object files could qualify as “references,” Wetmore still fails to teach, suggest or provide reason for a modification that would meet the claimed invention.

First, Wetmore causes *entry points* to become invariant through vectorization of the old and new object files. Wetmore does not describe replacing any *reference* to an entry point with a distinct label mark -- an enormous deficiency in Wetmore’s disclosure that the Requester seems to concede. (Request at 63 (“in the Wetmore ‘713 Patent . . . reference to entry points (303) . . . are not expressly disclosed as being replaced or modified”).) Accordingly, Wetmore fails to disclose or suggest “replacing the reference” in “substantially each reference entry,” which the Distinct Label Mark Claims (*i.e.*, Claims 1, 4, 14, 17, 27, 35, 38, 48, 51, and 61 and therefore dependent claims 2, 3, 15, 16, 36, 37, 49 and 50) require.

Second, Wetmore does not teach, disclose or even suggest replacing references that are not themselves “entry points,” such as references within the routines. Thus, as

discussed above with respect to the Invariant Reference Claims (Claims 8, 11, 21, 24, 28, 42, 45, 55, 58 and 62, and therefore dependent claims 9, 10, 22, 23, 43, 44, 56 and 57) (see supra pp. 12-13), Wetmore fails to teach or suggest this limitation.

**(4) Wetmore Fails To Disclose Generation of A
“Compact Difference Result”**

Regarding both the Executable Program Distinct Label Mark Claims and the Data Table Distinct Label Mark Claims (Claims 1, 4, 14, 17, 27, 35, 38, 48, 51 and 61, and therefore dependent claims 2, 3, 15, 16, 36, 37, 49 and 50), Wetmore fails to disclose generation of a “compact difference result.” As discussed above with respect to the Invariant Reference claims, Wetmore’s difference result simply includes the entirety of the code of a modified routine if any change is made within that routine. (See supra pp. 13-14). This level of granularity in generation of a difference result does not constitute a “compact difference result” as recited in these claims, as explained above.

iii. The Rejections Under 35 U.S.C. § 103

The Office has rejected the claims 2, 3, 9, 10, 15, 16, 22, 23, 36, 37, 43, 44, 49, 50, 56, and 57 under 35 U.S.C. § 103 over Wetmore in view of U.S. Patent No. 5,790,796 to Sadowsky (hereinafter “Sadowsky”). Owner respectfully traverses this rejection.

As discussed above, Wetmore does not disclose or suggest the features of independent claims 1, 8, 14, 21, 35, 42, 48, or 55 from which these claims depend. Accordingly, the Office has failed to make out a prima facie case of obviousness with regard to claims 2, 3, 9, 10, 15, 16, 22, 23, 36, 37, 43, 44, 49, 50, 56, and 57.

Moreover, the Owner traverses the Office’s suggestion that it would be obvious to a person of ordinary skill in the art to utilize Sadowsky’s method of transmitting a difference result over a network with Wetmore. In particular, since Wetmore is concerned with updating the operating system of a computer running in ROM and RAM, where the updated software modules to be distributed to the user must be loaded and the “patch” applied “at boot time,” the only way to implement the techniques of Wetmore would be to provide the patch on a “system disk.” (Wetmore 10:29-34; 11:35-39). The Office does not explain, and

Wetmore fails to suggest, how a patch formed using the techniques of Wetmore could be provided to a user over a network such that the patch could be subsequently applied to update the operating system (which in Wetmore is presumed to be stored in a Read-Only Memory), without making use of a "system disk" that is read "at boot time" to load the appropriate patch code into RAM. (Wetmore 11:35-39).

Additionally, the Owner respectfully suggests that the Office has employed improper hindsight reasoning in suggesting that the transmission of the Wetmore update over a network would "reduc[e] the costs associated with transferring the disks via normal transportation channels." (Office Action at 18). In particular, as of the priority date of the present invention, bandwidth was far more expensive than today, and the Owner respectfully disagrees with the Office's contention that transmitting the "patch" files of Wetmore (which, as discussed above, would contain far more data than a compact difference result of the presently claimed invention) via a network would reduce the costs associated with transferring disks, or would even be practical given the data transfer rates to end-users that were typical as of the Owner's priority date. Even if this were possible and practical, there still would remain the problem of how the user would apply the patch to his or her machine. Presumably, the user would need to first burn the received file to a CD, so that it could be used as Wetmore's "system disk" during the boot process. This would require each user wishing to update his or her system to purchase or make use of a CD recorder, in order to copy the patch files received via the network onto a "system disk." However, CD recorder technology was in its early stages as of the Owner's priority date, and was still expensive, with blank media costing approximately \$10/disk. (See Chris O'Malley, *A New Spin*, Time, Aug. 24, 1998 <<http://www.time.com/time/magazine/article/0,9171,988955,00.html>>). Thus, in view of the foregoing, Owner respectfully traverses the Office's rejections under § 103.

Owner reserves the right to submit substantial evidence of secondary considerations of non-obviousness in the unlikely event that the Office lodges further rejections under § 103.

E. Comments on Statement of Reasons For Patentability And/Or Confirmation

In addition to the reasons identified by the Examiner in the Statement of Reasons for Patentability of the confirmed claims, the Owner notes that those claims would also be allowable for many of the same reasons discussed above with regard to the rejected claims. Also, the Owner notes that the claims were somewhat inaccurately paraphrased (e.g., the statement "modified old program is reconstituted" more accurately would have said "reconstituting the modified *new* program" as per claims 5 and 12, for instance), but it is understood that the Office is relying on the claim language, rather than its paraphrasing of the claims, in indicating the reasons for allowance. The Owner also respectfully disagrees with the Examiner's characterization of the teachings of Wetmore for the reasons given above.

F. Conclusion

For the reasons set forth above, Owner respectfully traverses and requests withdrawal of the outstanding rejections, and requests prompt issuance of a Notice of Intent to Issue a Reexamination Certificate (NIRC) confirming the patentability of all claims. Should any residual issues exist or arise, or the Examiner believes an interview would be helpful noting the conversation above, the Examiner is invited to contact the undersigned below.

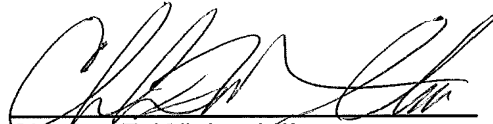
The Director is hereby authorized to charge any appropriate fees that may be required by this paper, and to credit any overpayment, to Deposit Account No. 02-4800.

Respectfully submitted,

BUCHANAN INGERSOLL & ROONEY PC

Date: June 29, 2010

By:



Charles F. Wieland III
Registration No. 33096

Customer No. 21839
703 836 6620

EXHIBIT 2



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/009,670	01/22/2010	6546552	GOOGLE 3.6-141	4316

21839 7590 08/11/2010

BUCHANAN, INGERSOLL & ROONEY PC
POST OFFICE BOX 1404
ALEXANDRIA, VA 22313-1404

EXAMINER

ART UNIT PAPER NUMBER

DATE MAILED: 08/11/2010

Please find below and/or attached an Office communication concerning this application or proceeding.



DO NOT USE IN PALM PRINTER

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

GOOGLE

LERNER, DAVID, LITTENBERG, KRUMHOLZ & MENTLIK, LLP

600 SOUTH AVENUE WEST

WESTFIELD, NJ 07090

***EX PARTE* REEXAMINATION COMMUNICATION TRANSMITTAL FORM**

REEXAMINATION CONTROL NO. 90/009,670.

PATENT NO. 6546552.

ART UNIT 3992.

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

Ex Parte Reexamination Interview Summary	Control No.	Patent Under Reexamination	
	90/009,670	6546552	
	Examiner	Art Unit	
	ANDREW L. NALVEN	3992	

All participants (USPTO personnel, patent owner, patent owner's representative):

- (1) ANDREW L. NALVEN (3) Charles Wieland
(2) Eric Keasel (4) Michael Weinberg
Albert Gagliardi

Date of Interview: 11 August 2010

Type: a) Telephonic b) Video Conference
c) Personal (copy given to: 1) patent owner 2) patent owner's representative)

Exhibit shown or demonstration conducted: d) Yes e) No.
If Yes, brief description: _____

Agreement with respect to the claims f) was reached. g) was not reached. h) N/A.
Any other agreement(s) are set forth below under "Description of the general nature of what was agreed to..."

Claim(s) discussed: 1 and 8.

Identification of prior art discussed: Wetmore.

Description of the general nature of what was agreed to if an agreement was reached, or any other comments:
Patent Owner gave a brief description of the claimed invention and the Wetmore reference. Discussed whether Wetmore taught the claimed "executable program" and the claimed invariant references or distinct label marks as required by claims 1 and 8. Examiner agrees that Wetmore does not teach the claimed invariant references or distinct label marks as claimed and defined by the specification.

(A fuller description, if necessary, and a copy of the amendments which the examiner agreed would render the claims patentable, if available, must be attached. Also, where no copy of the amendments that would render the claims patentable is available, a summary thereof must be attached.)

A FORMAL WRITTEN RESPONSE TO THE LAST OFFICE ACTION MUST INCLUDE PATENT OWNER'S STATEMENT OF THE SUBSTANCE OF THE INTERVIEW. (See MPEP § 2281). IF A RESPONSE TO THE LAST OFFICE ACTION HAS ALREADY BEEN FILED, THEN PATENT OWNER IS GIVEN **ONE MONTH** FROM THIS INTERVIEW DATE TO PROVIDE THE MANDATORY STATEMENT OF THE SUBSTANCE OF THE INTERVIEW (37 CFR 1.560(b)). THE REQUIREMENT FOR PATENT OWNER'S STATEMENT CAN NOT BE WAIVED. **EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.550(c).**

/Andrew L Nalven/ Primary Examiner, Art Unit 3992		
cc: Requester (if third party requester)		