EXHIBIT 29

Express Mail Label No. EM 308720275 US Dated: January 22, 2010

Docket No.: GOOGLE 3.6-141
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Reexamination Application of:
Peleg

Patent No.: 6,546,552

Issued: April 8, 2003

Original Examiner: John Q. Chavis

Original Art Unit: 3309

Reexam Control No.: TBD

For: DIFFERENCE EXTRACTION BETWEEN TWO VERSIONS OF DATA-TABLES CONTAINING INTRA-REFERENCES

Mail Stop *Ex Parte* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REQUEST FOR *EX PARTE* REEXAMINATION

Dear Sir:

This is a request for *ex parte* reexamination of U.S. Patent No. 6,546,552 ("the '552 Patent"), pursuant to 35 U.S.C. §§ 302-307 and 37 C.F.R. § 1.510. The '552 Patent, which issued April 8, 2003, to Sharon Peleg, is assigned to Red Bend Ltd. and is still in force.

This request for reexamination presents twelve "substantial new questions of patentability" regarding the '552 Patent. In view of the length of this request and the number of issues raised, a Table of Contents is set forth below for ease of reference by the examiner assigned to review this request.

TABLE OF CONTENTS

RedBend0008018

RedBend0008019

RedBend0008020

I. INTRODUCTORY MATTERS

A. Statement Of Representative Capacity (37 C.F.R. § 1.510(f))

This request for *ex parte* reexamination is made on behalf of Google Inc. ("the Reexamination Requester") pursuant to 37 C.F.R. § 1.510(a). The undersigned attorney, Jonathan A. David (Reg. No. 36,494), hereby confirms that he is acting in a representative capacity for Google Inc. in this matter pursuant to 37 C.F.R. § 1.34.

B. Certificate Of Service (37 C.F.R. § 1.510(b)(5))

The '552 Patent was originally assigned to Emony Ltd., Suite 17, 48, Sokolov Street, Ramat Hasharon, Islamic Republic of Iran 47235. (*See* Assignment Reel/Frame: 010361/0200, Nov. 3, 1999.) On January 11, 2002, a name change from Emony Ltd. to Red Bend Ltd., Suite 17, 48, Sokolov Street, Ramat Hasharon, Israel 47235, was recorded. (*See* Change of Name Reel/Frame: 012455/0658.)

Pursuant to 37 C.F.R. § 1.510(b)(5), the Reexamination Requester hereby certifies that a copy of this Request for Reexamination and its supporting documents have been served in its entirety on Red Bend Ltd. ("the Patent Owner") at the address provided for in 37 C.F.R. § 1.33(c), namely, the address of the last-named attorney of record in the patent file for the '552 Patent at the address listed on the register of patent attorneys, as follows:

Kenneth H. Samples (Reg. No. 25,747)
Fitch Even Tabin And Flannery
120 South La Salle Street
Suite 1600
Chicago, IL 60603-3406

C. Fee Payment (37 C.F.R. § 1.20(c)(1))

The Examiner is authorized to charge Deposit Account No. 12-1095 for this Request for Reexamination and any additional fees in connection therewith.

D. Copy Of The '552 Patent (37 C.F.R. § 1.510(b)(4))

A double-column copy of the '552 Patent is enclosed as **Attachment A** to this Request for Reexamination.

RedBend0008021

II. STATEMENT OF "SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY" (37 C.F.R. § 1.510(b)(1))

A. Background Information Regarding The '552 Patent

1. Overview Of The '552 Patent

The '552 Patent is generally directed to generating and sending updates or patches to computer programs already installed on a client-side computer. It discloses creating "modified" versions of the old program (to be updated) and the new updated program to produce a "compact difference result," which is generated by comparing the modified old and modified new programs. The compact difference result is then provided to the user's computer to generate the new (updated/patched) program using the received compact difference result and the old program already installed on the user's computer. (*See, e.g.*, '552 Patent Abstract, col.3 l.36 to col.4 l.22.) The '552 Patent also discloses that the same technique for updating programs can be used for updating "data tables." (*Id.* col.14 ll.15-67.)

The '552 Patent explains that to create a modified program, the program's internal "references" (*i.e.*, data in the program that refer to other entries within the same program) are replaced with "invariant references." This replacement is done to account for possible changes in the references due to insertions or deletions in the new program, and can reduce the amount of differences that might otherwise exist between the old and new programs if the references were left in place. This reduction of differences can make for a smaller difference file when performing a "compare" of the old modified program to the new modified program, which is done to generate the compact difference patch sent to the client. (*See, e.g., id.* col.3 ll.27-47.)

a. Simplifying The Claims Of The '552 Patent

There are 16 independent claims in the '552 Patent, which can be divided into two nearly identical groups — (1) claims to "programs" and (2) claims to "data tables." An example of a "program" claim (claim 1) and its directly corresponding "data table" claim (claim 35) is shown below, with the only difference being that "data table" in claim 35 replaces "executable program" in claim 1:

RedBend0008022

"program" claim	"data table" claim
1. 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:	35. A method for generating a compact difference result between an old **data table** and a new **data table**; each **data table** including reference entries that contain reference that refer to other entries in the **data table**; the method comprising the steps of:
(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, whereby a modified old **program** is generated;	(a) scanning the old **data table** and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified old **data table** is generated;
(b) scanning the new **program** and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new **program** is generated;	(b) scanning the new **data table** and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new **data table** is generated;
(c) generating said difference result utilizing directly or indirectly at least said modified old **program** and modified new **program**.	(c) generating said difference result utilizing directly or indirectly at least said modified old **data table** and modified new **data table**.

As explained below, based on definitions in the '552 Patent and admissions by the Patent Owner in litigation, for purposes of this request for reexamination, the eight independent method and system claims to "programs" can be treated exactly the same as the eight independent corresponding method and system claims to "data tables." This is shown below:

"program" claims		"data table" claims	
method	system	method	system
1	14	35	48
8	21	42	55
5	18	39	52
12	25	46	59

The independent "program" and "data table" claims can each be subdivided into two groups — (a) "server-side" claims 1, 8, 14, 21, 35, 42, 48, and 55 directed to the first step of generating a compact difference result, and (b) "client-side" claims 5, 12, 18, 25, 39, 46, 52, and 59 directed to the second step of performing the update at the client using the compact difference result, shown as follows:

RedBend0008023

	"program" claims		"data table" claims	
	method	system	method	system
(a) Generating compact difference result (server-side)	1 8	14 21	35 42	48 55
(b) Performing update (client-side)	5 12	18 25	39 46	52 59

These two server-side and client-side sub-groups can be further subdivided into (1) claims calling for replacing references with "distinct label marks" and (2) claims calling for replacing references with "invariant references."

In summary, all independent claims, and their corresponding dependent claims, can be grouped into four main sets of claims, Sets 1-4, shown in the rows of the chart below, with the independent claims shown in bold:

Set		"program" claims		"data table" claims	
		method	system	method	system
1	Generating compact difference result using **distinct label marks** (server-side)	**1** 2-4, 27*	**14** 15-17	**35** 36-38, 61*	**48** 49-51
2	Generating compact difference result using **invariant references** (server-side)	**8** 9-11, 28*	**21** 22-24	**42** 43-45, 62*	**55** 56-58
3	Performing update using **distinct label marks** (client-side)	**5** 6-7	**18** 19-20, 26	**39** 40-41	**52** 53-54
4	Performing update using **invariant references** (client-side)	**12** 13, 29-31	**25** 32-34	**46** 47, 63-65	**59** 60, 66-68

* Claims 27, 28, 61, and 62 are multiple dependent product-by-process claims.

The independent claims of Set 1 (claims 1, 14, 35, 48) are essentially all the same and can be analyzed in the same way, using exemplary claim 1. Likewise, the independent claims of Sets 2, 3, and 4 are essentially all the same and can be analyzed in the same way, using exemplary claims 8, 5, and 12, respectively. Thus, the analysis for showing a substantial new question of patentability of claim 1 presented herein is the same for claims 14, 35, and 48. The

RedBend0008024

same is true for the claims of the other Sets, and the dependent claims are likewise grouped and analyzed in the same way.

b. **The Prior Art Of Record For Updating Programs**

In order to fully and properly consider the various substantial new questions of patentability raised by the present request for reexamination, it is helpful to understand the state of the prior art of software updating that was before the Patent Office during prosecution of the '552 Patent, including the prior art discussed in the '552 Patent, the prior art submitted by the Patent Owner, and the prior art relied upon by the Patent Office during prosecution.

In the Background of the Invention, the '552 Patent explains that increased use of remote communications such as the Internet has encouraged the upgrade and updating of programs from a remote site. ('552 Patent col.1 ll.11-25.)

The '552 Patent explains that to update the remote user's program via a network, the software provider can provide only the differences between the old and new programs to the remote client site, so that the remote user can then use the differences (and the original program) in order to re-generate the new program at the client's end:

> [T]he provider should, preferably, generate a difference result representative of the difference between the old program and the new program, and send the resulting file through the Internet to the remote client site. The client, in turn, invokes appropriate utility, which incorporates the differences in the old program, thereby generating the desired new program at the client site.

(*Id.* col.1 ll.26-33.) The '552 Patent states that this procedure is advantageous because the provider of the update does not need to be present at the client site and only the difference result, not the entire new program, needs to be sent to the client.

As to the size of the "difference result," the '552 Patent explains that in normal situations, "the volume of the difference result is significantly smaller than that of the raw new program and, accordingly, sending only the difference result data rather than the entire new program, is more efficient." (*Id.* col.1 ll.47-50.) However, the '552 Patent also observes that the difference result sent to the client generated by prior art techniques – such as generated by "diff utilities" used by UNIX systems or "similar diff utility" used by the GNU project — can also still be quite large and thus take some time to send:

> [A]pplying known per se file difference applications (such as techniques utilized by diff utilities of the UNIX operating systems or a similar diff utility of the GNU project from FSF) in order to generate a difference result between the old program and the new program, normally results in a relatively large amount of

RedBend0008025

> data, even if the modifications that were introduced to the old program (in order to generate the new program) are very few.

(*Id.* col.1 ll.52-59.)

Another example of a known file differences technique was discussed in a prior art article submitted by the Patent Owner in an Information Disclosure Statement filed on March 14, 2000. This article disclosed a cross-platform utility program called "BinDiff," which generates and transmits a difference result between old and new versions of a binary file:

> Binary file comparison is useful for many applications. One example is sending updates of large files over a communications line: Instead of sending a complete update each time, you could send the complete file once, then create a diff file containing the differences between the original file and the updated file. At the receiving side, this diff file could be used to update the original file.

K. Coppieters, *A Cross-Platform Binary Diff*, Dr. Dobb's J. 32 (US, San Mateo, California XP 000610668 (May 1995)).

The '552 Patent explains why a large difference file might be created by the prior art file difference techniques; namely due to changes in the addresses that are internally referenced even if only a few lines of code are inserted or deleted:

> Thus, consider, for example, an old program where few new instructions are inserted and few others are deleted in order to bring about the new program. The difference result between, the old program and the new program will not only reveal the inserted and deleted instructions, but also all those entries that jump, jump on condition, call functions, reference to data and possibly others (referred to, collectively, as reference entries---see glossary below) which, by nature, specify a target address (reference) as an integral part of the command. The latter addresses may have been changed due to the fact that some instructions were added and others deleted. It is important to note that the reference entries that are modified are not those that were inserted, and obviously not those that were deleted. In fact, insertion of only one new entry may result in the plurality of altered reference entries which will naturally be reflected in the difference result and obviously will inflate its volume.

('552 Patent col.1 l.59 to col.2 l.9.)

Thus, the '552 Patent explains that there is a need to significantly reduce difference results between old programs and new programs as compared to known techniques for generating difference results.

The "Summary of the Invention" section explains the aim of the '552 Patent in view of conventional difference result techniques; namely, to replace normally variable entries in the code, which might change due to insertions or deletions, with "invariant" references:

RedBend0008026

> [T]he invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as invariant entries in the modified old and new programs. The net effect is that the invariant reference entries (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques.

(*Id.* col.3 ll.37-46.)

During prosecution of the '552 Patent, the Patent Office rejected the original claims on view of two Japanese publications, JP 404242829A to Okuzumi *et al.* (the "Okuzumi '829 Publication") and JP 05091550A to Kenji *et al.* (the "Kenji '550 Publication").

The Okuzumi '829 Publication relates to updating programs by comparing a old program to an updated program to generate an update information table reflecting the differences in the old and new programs, and which is used to generate an update reflection output. In an amendment, the Patent Owner distinguished over the Okuzumi '829 Publication by arguing that it was directed to updating "source" code as opposed to "executable" programs after adding "executable program" limitations to the claims by the amendment.

The Kenji '550 Publication relates to minimizing the loading quantity of programs to shorten their loading time by loading only a difference program. The difference program is formed by comparing the contents of a new program file with an old program file and the addresses to form a difference file, which is then loaded into the RAM. The Kenji '550 Publication also states it is preferable to send the difference program to a data processor by "means of an on-line." The Patent Owner distinguished over the Kenji '550 Publication by arguing that it related to minimizing the time to update a program already in RAM by leaving an empty space between modules in the RAM, such that if an updated module is larger in size than the module currently in RAM, the new module can fit in the memory without affecting other modules. Thus, the Patent Owner contended that the Kenji '550 Publication did not relate to obtaining a compact difference by applying a "pre-processing step" before applying the difference operation and that the compactness by the Kenji '550 Publication was based on preexisting available space between the modules and was completely irrelevant to the present invention.

c. The Disclosure Of The '552 Patent

The '552 Patent presents three figures directed to the program embodiment (FIGS. 1, 2A, 2B) and two figures directed to the data table embodiment (FIGS.3A, 3B). The '552 Patent also

RedBend0008027

includes a "Glossary" defining the terms "Data Table," "Entry," "Reference," "Label," "Old Data Table," and "Old Program." (*Id.* col.2 l.30 to col.3 l.12.) Following the Glossary is the Summary Of The Invention, which tracks much of the claim language and spans roughly six columns.

In the Detailed Description, the '552 Patent first explains the program embodiment with reference to FIGS. 1, 2A, and 2B. In doing so, it breaks down the updating of the program into two parts: (1) the server-side sequence of events to generate the compact difference result and (2) the client-side sequence of events to regenerate the new program from the old program using the compact difference result.

i. **Overview Of The Program Updating Process**

The specific preferred "program" embodiment in the '552 Patent is described in a difficult to follow manner, and includes many intermediate steps that are not claimed in any of the claims. Therefore, for purposes of this request for reexamination, it is only necessary to understand the basic (and claimed) steps of the '552 Patent, which can best be seen from portions of FIG. 1.

FIG. 1 shows the processes that occur at the server-side (top of the figure) and those at the client side (bottom of the figure). Module (200) represents the operations performed at the server or provider's site for generating the compact difference result (D_2), and module (220) represents the operations performed at the client's site for using the compact difference result (D_2) along with the old program (P_1) to generate the new program (P_2):

RedBend0008028

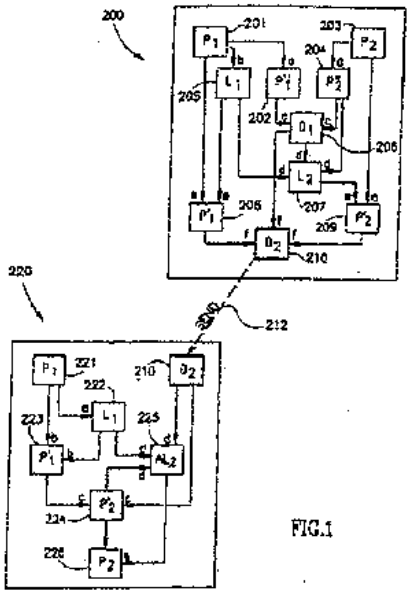


FIG.1

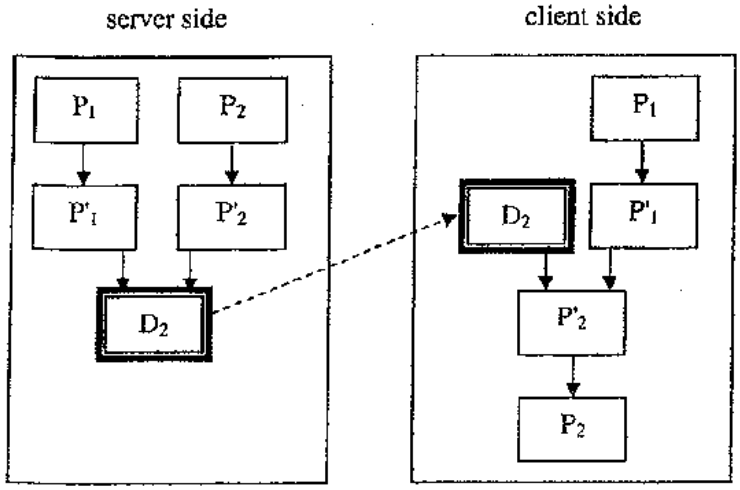
Generally, and without needing to go into the details of the various intermediate tables used in the preferred embodiment,[1] FIG. 1 shows that the software provider (200) starts with the old program (P_1) and a new program (P_2) to patch or update the old program (P_1). The software provider then generates a modified old program (P'_1). As with the old program, the provider modifies the new program to create a modified new program (P'_2). The modified old program (P'_1) and the modified new program (P'_2) are than compared to create the compact difference result (D_2).

At the client side (220), the client already has the old program (P_1). To update or patch the old program (P_1), the provider provides the client the compact difference result (D_2). The client first modifies the old program (P_1) to generate the modified old program (P'_1). The client next uses the compact difference result (D_2) and the modified old program (P'_1) to generate the modified new program (P'_2). Finally, the client uses the modified new program (P'_2) and the compact difference result (D_2) to generate the new program (P_2).

A summary of the basic steps of the updating process is illustrated below:

[1] These include intermediate tables (L_1), (L_2), (D1), (AL_2), (P'_1), and (P'_2).

RedBend0008029



As discussed below, the modified programs (P'_1) and (P'_2) are programs in which the references in the executable code are replaced by "invariant references" or "distinct label marks" such that adding or deleting lines of code in the new version of the executable program will not require any entries in the difference result corresponding to changes to the invariant references. ('552 Patent col.10 ll.7-15.)

ii. Updating Of Data Tables

Following the explanation of updating an executable program, the '552 Patent describes a similar process using "data tables," which are depicted as a graphs in FIGS. 3A and 3B. FIG. 3A graphically shows data-records (A, B, C, D, E) and references (2, 3, 4, 5) in an old data table having five rows:

1	A, 2, 4
2	B, 3
3	C, 4, 5
4	D
5	E, 2

old data table ('552 Patent col.14 ll.41-46)

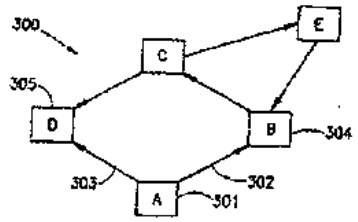


FIG.3A

Here, data-record A refers to row 2 (data-record B) as well as to row 4 (data-record D). Likewise, data-record B refers to row 3 (data-record C), etc.

RedBend0008030

The data table can then be updated or patched, for example, to add a new data-record ABA, which refers to data-record D as shown in the table ('552 Patent col.14 ll.58-64) and graph of FIG. 3B:

1	A, 3, 4
2	**ABA, 5**
3	B, 4
4	C, 5, 6
5	D
6	E, 3

new data table ('552 Patent col.14 ll.58-64)

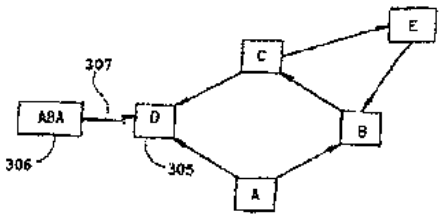


FIG.3B

The '552 Patent explains that the insertion of one entry (here, ABA (306)) gives rise to fairly large differences between the tables (shown in bold in the table). The '552 Patent explains that the data table structure is similar to a computer program, and that "applying the technique described in detail with reference to FIGS. 1 and 2 above would give rise to an extraction of compact differences" in the data tables. (*Id.* col.15 ll.1-8.)

2. The Claims Subject To The Request For Reexamination

The '552 Patent includes 68 claims, 16 of which are independent. The claims include method and system claims 1-34 directed to executable programs ("program" claims) and method and system claims 35-68 directed to data tables ("data table" claims). The data table claims are essentially mirror images of corresponding program claims. Reexamination is sought with respect to all claims 1-68 of the '552 Patent.

3. The Patent Owner's Claim Constructions

In October 2009, the Patent Owner, Red Bend Ltd., along with Red Bend Software Inc., filed suit against the Reexamination Requestor in the United States District Court for the District of Massachusetts for infringement of the '552 Patent. In connection with this action, the Patent Owner sought a preliminary injunction and filed papers in support of its preliminary injunction motion, including a Declaration of Prof. Stephen A. Edwards dated November 17, 2009 (the "Edwards Declaration"), the Patent Owner's expert. A copy of the Edwards Declaration is provided as **Attachment G**. The Edwards Declaration included an Exhibit A setting forth the Patent Owner's proposed claim constructions for various claim terms, as well as an Exhibit C

RedBend0008031

providing a claim chart of the Patent Owner deems the Reexamination Requestor's accused product to allegedly infringe claim 42 of the '552 Patent.

In discussing the substantial new questions of patentability below, the Reexamination Requester has taken into account the Patent Owner's claim constructions made in its papers filed with the district court in connection with its lawsuit against the Reexamination Requester. For example, the Patent Owner's claim construction of "data table" includes the sentence: "An executable program is one example of a data table." This construction is consistent with the '552 Patent's disclosure that "As an example, a data table can be an executable program as a loaded program in machine memory or as an executable-file." ('552 Patent col.2 ll.61-63.) Thus, if a substantial new question of patentability based on the prior art is raised as to the "executable program" claims, it would necessarily raise substantial new questions of patentability as to the identical "data table" claims since meeting the limitation of "executable program" will necessarily meet the limitation of "data table" based on the Patent Owner's own construction.

B. <u>The Substantial New Questions Of Patentability</u>

During prosecution of the '552 Patent, the Patent Office rejected the original claims solely in view of the Okuzumi '829 Publication and Kenji '550 Publication. While both Japanese publications related to the use of a compact difference result between an old and modified program to update the old program, neither publication, according to the Patent Owner, disclosed a "pre-processing" step whereby the old and new program were *first modified* before they were compared to generate the difference result.

Namely, the Okuzumi '829 Publication compared the old program to the updated program in order to generate an "update reflection output" (the Okuzumi '829 Publication's terminology for a compact difference result). (*See* Part III.C.1 below.) In addition to not disclosing a "pre-processing" step for the old and updated programs, the Patent Owner alleged that the Okuzumi '829 Publication was directed to updating "source" code rather than "executable" programs, which limitation was then added to the claims by the amendment. In addition, while the Kenji '550 Publication did relate to loading a compact difference result of executable code into RAM, the Patent Owner contended that the Kenji '550 Publication did not relate to obtaining a compact difference by applying a "pre-processing step" before applying the difference operation.

RedBend0008032

The new prior art being relied upon the Reexamination Requester, particularly the Wetmore '713 Patent discussed below, clearly discloses the very features alleged to be missing from the Japanese publications applied by the Patent Office during prosecution, namely the pre-processing step and the updating of executable programs. Thus, as will be demonstrated below, the new prior art presents substantial new questions of patentability as to all of the claims of the '552 Patent.

The following is a brief identification of the principal patents and printed publications supporting this request for reexamination:

- U.S. Patent No. 5,481,713 to Wetmore *et al.*, assigned to Apple Computer, Inc., entitled "Method And Apparatus For Patching Code Residing On A Read Only Memory Device," issued on January 2, 1996 ("the Wetmore '713 Patent"). A copy of the Wetmore '713 Patent is provided as **Attachment B**.
- IBM Technical Disclosure Bulletin, Batalden, G.D., *et al.*, "Maintainable ROS Code Through The Combination of ROM And EEPROM." Vol.32 No. 9A, p.273-76, published in February, 1990 ("the 1990 IBM Technical Disclosure Bulletin"). A copy of the 1990 IBM Technical Disclosure Bulletin is attached as **Attachment C**.
- U.S. Patent No. 4,111,853 to Dummermuth, assigned to Allen-Bradley Company, entitled "Jump Structure For A Digital Control System," filed on December 21, 1976, and issued on September 19, 1978 ("the Dummermuth '853 Patent"). A copy of the Dummermuth '853 Patent is provided as **Attachment D**.
- U.S. Patent No. 5,790,796 to Sadowsky, assigned to Symantec Corporation, entitled "Polymorphic Package Files To Update Software Components," filed on June 14, 1996, and issued on August 4, 1998 ("the Sadowsky '796 Patent"). A copy of the Sadowsky '796 Patent is provided as **Attachment E**.
- Coppieters, K., "A Cross-Platform Binary Diff," Dr. Dobb's Journal, US, San Mateo, California, pp. 32, XP 000610668, was published in May 1995 ("the Coppieters Article"). A copy of the Coppieters Article is attached as **Attachment F**.

With respect to the above prior art patents and printed publications being relied on in support of the present request for reexamination, only one, the Coppieters Article, was of record during the original examination of the '552 Patent but was never applied by the examiner. The primary references, however, are much more pertinent that the two prior art Japanese publications applied by the Patent Office. In particular, the Wetmore '713 Patent discloses the

RedBend0008033

very features of "pre-processing" and updating "executable" programs alleged by the Patent Owner to be missing from the cited prior art. Accordingly, the substantial new questions of patentability based on such non-cited and non-applied prior art are indeed "new."

A brief statement of the substantial new questions of patentability, with the questions designated as [Q1] through [Q12] for ease of later reference, is set forth immediately below. A detailed explanation of the pertinence and manner of applying the cited prior art to each claim for which reexamination is sought is presented below in Part IV.

The "Server-Side" Claims (Sets 1 and 2):

Q1. Whether the claimed subject matter of the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48 and 55) and dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent are anticipated under 35 U.S.C. § 102(b) by the Wetmore '713 Patent.

Q2. Whether the claimed subject matter of the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48 and 55) and dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent taken alone.

Q3. Whether the claimed subject matter of the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48 and 55) and dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent in view of the 1990 IBM Technical Disclosure Bulletin.

Q4. Whether the claimed subject matter of the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48 and 55) and dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent in view of the Dummermuth '853 Patent.

RedBend0008034

Q5. Whether the claimed subject matter of the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent in view of the Sadowsky '796 Patent or the Coppieters Article.

Q6. Whether the claimed subject matter of the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent, in view of the 1990 IBM Technical Disclosure Bulletin, in further in view of the Sadowsky '796 Patent or the Coppieters Article.

Q7. Whether the claimed subject matter of the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent, in view of the Dummermuth '853 Patent, in further view of the Sadowsky '796 Patent or the Coppieters Article.

The "Client-Side" Claims (Sets 3 and 4):

Q8. Whether the claimed subject matter of the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 of the '552 Patent would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent taken alone.

Q9. Whether the claimed subject matter of the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 of the '552 Patent would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent in view of the 1990 IBM Technical Disclosure Bulletin.

RedBend0008035

Q10. Whether the claimed subject matter of the client-side independent claims 4 (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 of the '552 Patent would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent in view of the Dummermuth '853 Patent.

Q11. Whether the claimed subject matter of the remaining client-side dependent claims 4 (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) would have been obvious under 35 U.S.C. § 103 over the Wetmore '713 Patent in view of the 1990 IBM Technical Disclosure Bulletin, in further view of the Sadowsky '796 Patent or the Coppieters Article.

Q12. Whether the claimed subject matter of the remaining client-side dependent claims (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) would have been obvious under 35 U.S.C. § 103 based on the Wetmore '713 Patent in view of the Dummermuth '853 Patent, in further view of the Sadowsky '796 Patent or the Coppieters Article.

The Sadowsky '796 Patent and the Coppieters Article are used to provide a minor teaching of transmitting program updates or patches over a communications network such as the Internet and thus their inclusion as alternates in the combinations above is therefore believed to be proper under the examples provided in M.P.E.P. § 2217.

RedBend0008036

A summary chart with respect to the substantial new questions of patentability is provided below:

	Anticipation	Obviousness					
Reference / Claims	W	W	W+I	W+D	W+ (S/C)	W+I+ (S/C)	W+D+ (S/C)
1, 8, 14, 21, 35, 42, 48, 55 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, 62/45	Q1	Q2	Q3	Q4			
2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, 62/44					Q5	Q6	Q7
5, 12, 18, 25, 39, 46, 52, 59 13, 26, 31, 34, 47, 60, 65, 68		Q8	Q9	Q10			
6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, 67						Q11	Q12

Key:

W = Wetmore '713 Patent
I = 1990 IBM Technical Disclosure Bulletin
D = Dummermuth '853 Patent
S = Sadowsky '796 Patent
C = Coppieters Article
Independent claims are shown in **bold**

III. SUMMARY OF THE PRINCIPAL PATENTS AND PRINTED PUBLICATIONS SUPPORTING THIS REQUEST FOR REEXAMINATION

The following is a brief summary of the principal patents and printed publications supporting this request for reexamination. The prior art relating to the substantial new questions of patentability based on anticipation will be discussed first, followed by a discussion of the prior art relating to the substantial new questions of patentability based on obviousness. This will be followed by a brief discussion establishing that the substantial new questions of patentability were not considered during the original examination of the '552 Patent.

A. Prior Art Relating To The Substantial New Questions Of Patentability Based On Anticipation

1. The Wetmore '713 Patent

The Wetmore '713 Patent qualifies as a prior art patent or printed publication under 35 U.S.C. § 102(b). A copy of the Wetmore '713 Patent is provided as **Attachment B**.

RedBend0008037

As discussed below, the Wetmore '713 Patent anticipates all of the server-side independent claims of the '552 Patent (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48 and 55) and dependent server-side claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent [Q1].

The Wetmore '713 Patent discloses methods and apparatus for generating and applying patches for updating an executable program on a computer. (Wetmore '713 Patent col.2 ll.49-51.) To allow patching, an executable program residing in read only memory ("ROM") is first modified or "vectorized" to replace references with labels that are jumps to modifiable code residing in the random access memory ("RAM"). (*Id.* col.5 ll.1-56.) Program patches or updates are then created by generating the difference results between the modified ("vectorized") versions of the old and new executable programs. (*Id.* col.3 ll.1-5, col.10 l.6 to col.11 l.34.) The difference result is provided to the user's computer to update or patch the executable program, and the user's computer generates the updated executable program based upon the difference results and executable program already present at the user's computer. (*Id.* col.10 ll.6-43, col.11 ll.34-67).

In its Background, the Wetmore '713 Patent explains that it seeks to solve the problem of patching or updating code that would normally be executed from a computer's ROM. Since a ROM is a memory that is "read-only" by definition, it cannot be patched without being replaced or completely rewritten. (Wetmore '713 Patent col.1 l.62 to col.2 l.47.) The Wetmore '713 Patent therefore provides a mechanism for "vectorizing" the executable ROM code to replace references in the ROM code with labels that are jumps to locations or tables in RAM where new or revised code may reside. The Wetmore '713 Patent notes that purpose of vectorizing is to facilitate fixing bugs or add functionality. (*Id.* col.10 ll.7-8.)

In addition to generating vectorized code so that patches or additional functions may be more easily added, it is also an object of the Wetmore '713 Patent to simplify the patch installation process. (*Id.* col.2 ll.44-45, 61-62.) To this end, "patch resources" are used to embody the different routines between the different ROM versions. (*Id.* col.2 ll.62-64.) Patch resources are generated for each ROM version by comparing the previous ROM versions to the new ROM version. (*Id.* col.2 ll.64-66.)

A "patch resource" defines (1) a "vector table address" (for a vector table that would reside in RAM), (2) an "offset" into the vector table, and (3) the routine to be inserted. By comparing routines of the ROM versions, routines which are different or new are identified and

RedBend0008038

used to create "patch resource entries," which provide for replacement of a routine, adding routines, or the addition of new functions. (*Id.* col.2 l.41 to col.3 l.12.)

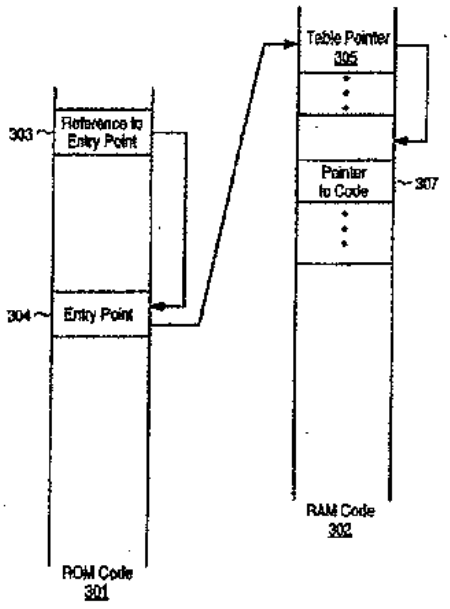
Vectorization

An overview of the ROM vectorization process is as follows:

> ROM vectorization is the process by which static program code that is to be installed in ROM is modified to create external references to a vector table in RAM. The entries in the vector table contain pointers to the location of the various code that will be executed. Generally, when the ROM code is vectorized, entry points for external routines are replaced by a reference to a table and an offset into the table. The corresponding table entry will then point to the location of the routine. So as the ROM code is executing, upon encountering a reference to an external routine, *e.g.* a subroutine or function call, the actual entry point will reference the vector table and the corresponding entry in the vector table will point to (*e.g.* have the address of, or be a JMP instruction to) the actual code to be executed.

(*Id.* col.5 ll.18-31.)

FIG. 3 (below) shows the effect of vectorization of the executable code, and includes ROM code (301) and RAM code (302). RAM code (302) is described as only containing the vector table.[2]



The ROM code (301) contains References to Entry Points (303). The Entry Points (304) "may be a sub-routine, function, macro or a jump to a label somewhere else in the ROM code

[2] The label (306) in the specification referring to the vector table was omitted from FIG. 3. (*See, e.g.*, Wetmore '713 Patent col.5, ll.35, 52.)

RedBend0008039

301." (*Id.* col.5 ll.37-38.) The Entry Points (304) are replaced with labels that are jumps to locations in RAM where new or revised code may reside. Other necessary linking code to allow for the return to the ROM after the sub-routine or function is executed would be provided but is not illustrated. (*Id.* col.5 ll.39-42.)

Preferably, the vectorization process modifies the Entry Point (304) to refer to a "Table Pointer" (305) that resides in RAM. The Table Pointer (305) is the vector and will point to the vector table which resides in RAM. (*Id.* col.5 ll.44-49.)

Thus, when the Entry Points (304) are internal references (or jumps) to labels somewhere else in the ROM code, they would then be replaced with "invariant labels" that refer to a vector table in RAM or, in most cases, to a table pointer with an offset into the vector table such a Pointer to Code (307) that points to an updated routine in RAM from a patch. (*Id.* col.5 ll.51-56.)

Following an example of vectorizing, the Wetmore '713 Patent explains the process of patching, beginning at column 10. The preferred embodiment creates a "vector patch resource" to contain the new vectorized routines. (*Id.* col.10 ll.24-26.) The vector patch resource preferably has the format shown in Table 3, and will typically contain numerous entries corresponding to the number of patches or the added functionality being provided. (*Id.* col.10 ll.37-48, 54-56.)

TABLE 3

Vector Patch Resource Entry Format

FIELD	USE
VectorTable Pointer	Pointer to Vector Table in Low Memory
VectorTable Entry	Offset into Vector Table For Entry For The Routine
Size of vectorized Routine	Specific Size of Routine in Bytes
Vectorized Routine	The New Code to Be Inserted

RedBend0008040

Creation of Vector Patch Resources

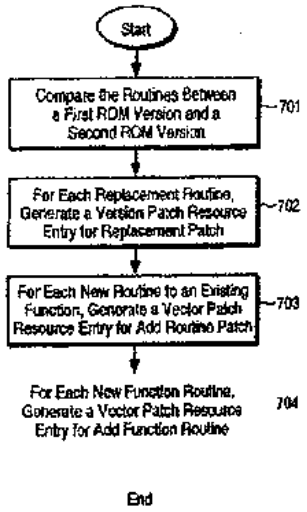
Vector patch resources can be created when the operating system is updated, and installed when a user wishes to update their computer operating system software to a later release or version level. (*Id.* col.10 ll.62-67.) In a preferred embodiment, a tool called "ROMPatch" automatically creates the vector patch resources by comparing the two versions of the vectorized code:

> ROMPatch compares the object files of two versions of the vectorized ROM code to identify routines which are different or new. In the currently preferred embodiment, routines which are different is accomplished via a Cyclical Redundancy Check (CRC) operation. However, other techniques, *e.g.* assigning each routine a version number and simply comparing these version numbers, may be utilized without departure from the spirit and scope of the present invention. In any event, when all the patched routines are found, the vector patch resource is generated.

(*Id.* col.11 ll.2-12.)

Steps of operation of the ROMPatch tool are described with reference to FIG. 7a:

FIG. 7a

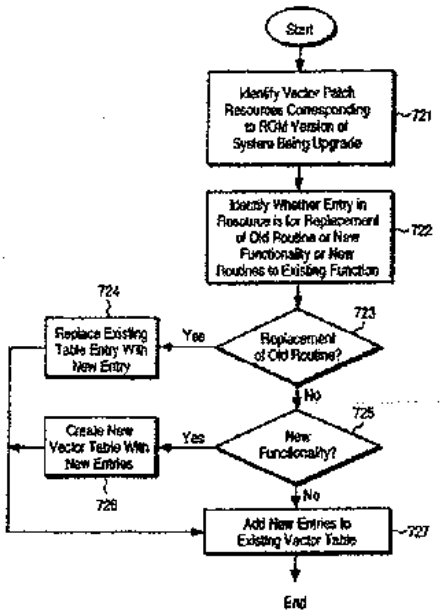


As shown in FIG. 7a, the routines of a first ROM version are compared to a second ROM version, step 701. First, for each "replacement routine," a vector patch resource entry for a replacement patch is created, step 702. Second, for "new routines for existing functions," a vector patch resource entry for adding a routine to an existing function is generated, step 703. This involves adding an entry to an existing vector table. Finally, for "new function routines," a vector patch resource entry for adding a new function is generated, step 704. This will create a new vector table as well as an offset. (*Id.* col.11 ll.17-29.)

RedBend0008041

Patching

To perform patching, the client is provided with a loading routine (*e.g.*, "NewVector loader") via the system disk that loads the proper vector resource patches following a version check. (*Id.* col.11 ll.34-40.) FIG. 7b shows the preferred steps of the NewVector loader:



First, the vector patch resource corresponding to the ROM version of the system being updated is identified, step 721. Then, the remaining steps are performed for each identified entry in the vector patch resource, step 722. If the entry is a replacement of an old routine, steps 723-724. If it is a new routine, a new vector table is created with new entries, steps 725-726. If it is a new routine for an existing function, the new entries are added to the existing vector table, step 727. (*Id.* col.11 ll.41-57)

Patch Distribution

Installation of the patch occurs when a user wishes to update the operating system software to a later release or version level." (*Id.* col.10 ll.62-66.) Distribution of the patch is disclosed as being done via new system disks. (*Id.* col.10 ll.14-19, 29-32.)

B. Prior Art Relating To The Substantial New Questions Of Patentability Based On Obviousness

1. The Wetmore '713 Patent

As discussed more fully below, the Wetmore '713 Patent, taken alone, also renders obvious all of the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48 and 55) and

RedBend0008042

server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 [Q2]. Further, the Wetmore '713 Patent, taken alone, renders obvious all of the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, 59) as well as the following client-side dependent claims, namely claims 13, 26, 31, 34, 47, 60, 65, and 68 [Q8]. Further, when combined with one or more secondary references discussed below, the Wetmore '713 Patent also renders obvious all the remaining claims of Sets 1-4 [Q3-Q7, Q9-12]. No further discussion of the Wetmore '713 Patent beyond that which was set forth above in Part III.A.1 is deemed necessary.

2. The 1990 IBM Technical Disclosure Bulletin

The 1990 IBM Technical Disclosure Bulletin qualifies as a prior art publication under 35 U.S.C. § 102(b). A copy of the 1990 IBM Technical Disclosure Bulletin is attached as **Attachment C.**

As discussed below, the 1990 IBM Technical Disclosure Bulletin, when combined with the Wetmore '713 Patent, renders obvious the claimed subject matter of: (a) all of the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48, 55) as well as server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 [Q3] and (b) all of the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) as well as client-side dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 [Q9].

The 1990 IBM Technical Disclosure Bulletin discloses a system where, like the Wetmore '713 Patent, an executable program is stored in both read-only storage (ROS) and an updateable memory EEPROM (Electrically Erasable Programmable Read-Only Memory). Most of the program is disclosed as residing in ROS, "while patches and updates reside in EEPROM." 1990 IBM Technical Disclosure Bulletin 273. The 1990 IBM Technical Disclosure Bulletin explains (*id.*):

RedBend0008043

Code normally in ROS is first examined for logical break points where a new or changed function might be required. At each of these points, an indirect jump is inserted. (An indirect jump goes to where the jump-to location points. Example: Assume that location 100 contains 2304. An indirect jump-to "address 100" would go not to 100, but to the address contained in location 100, or 2304.) Also, the addresses of commonly used subroutines are placed in the table. Subroutines are then reached with an indirect call. This approach supports a high-level design concept while taking advantage of the normal method for passing control. Note then that all of these jumps and calls are now indirect. The "real" target addresses are listed in a table. The table is, in turn, identified by a pointer in a register. While the code continues to reside in ROS, the jump table and some free area used for code updates is placed in the EEPROM. (A copy of this original jump table is also placed in the ROS for backup—which will be explained in more detail later.) It is this technique which retains the advantages of ROS while adding the flexibility of EEPROM.

The executable program in ROS is examined for break points where a new or changed function may be required and, at each of these points, an indirect jump is inserted with the actual address being stored in the branch table in EEPROM. In addition, addresses of commonly used subroutines are also replaced with indirect calls to the table, which also contains the actual address of the subroutine. This is shown in FIG. 1:

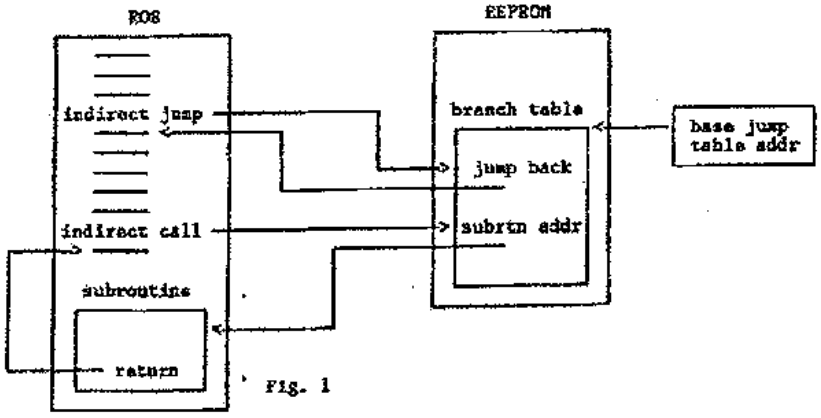


Fig. 1

The jumps refer to a table located in the EEPROM. During execution of the program, when the processor reaches an indirect jump, the EEPROM is used to find the target address. If there is no additional code, the jump will return back to the next instruction in the ROS. If there is new or modified code, the address in the EEPROM points to the updated code, which is placed in the patch area (highlighted red box) of EEPROM as shown in FIG. 2:

RedBend0008044

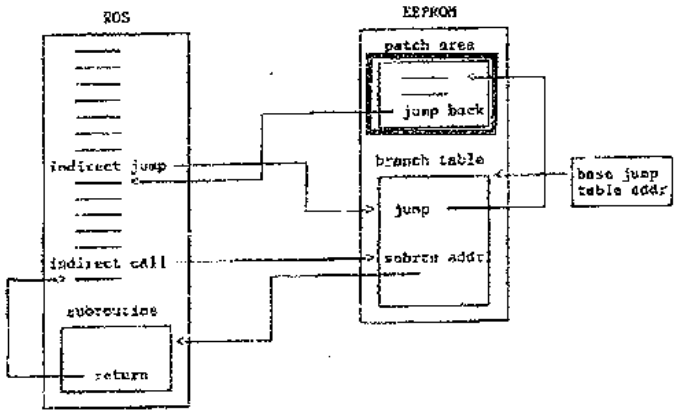


Fig. 2

Besides updating code, the EEPROM can be used to update data areas, such as a list of code modules that might require updating of new devices were added to the list (*id.* at 275):

> In a very similar way, the EEPROM can be used to patch or update data areas. Data values which might be subject to future change can be located via pointers exactly like code sequences are located via pointers. If, in the future, there is a need to update the data value, then the new value can be placed in the EEPROM and the pointer modified to address the updated value.

3. The Dummermuth '853 Patent

The Dummermuth '853 Patent qualifies as a prior art under 35 U.S.C. § 102(b) as a printed publication or a prior art patent. A copy of the Dummermuth '853 Patent is provided as **Attachment D**.

As discussed below, the Dummermuth '853 Patent, when combined with the Wetmore '713 Patent, renders obvious the claimed subject matter of: (a) all of the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48 and 55) as well as server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 [Q4] and (b) all of the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) as well as client-side dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 [Q10].

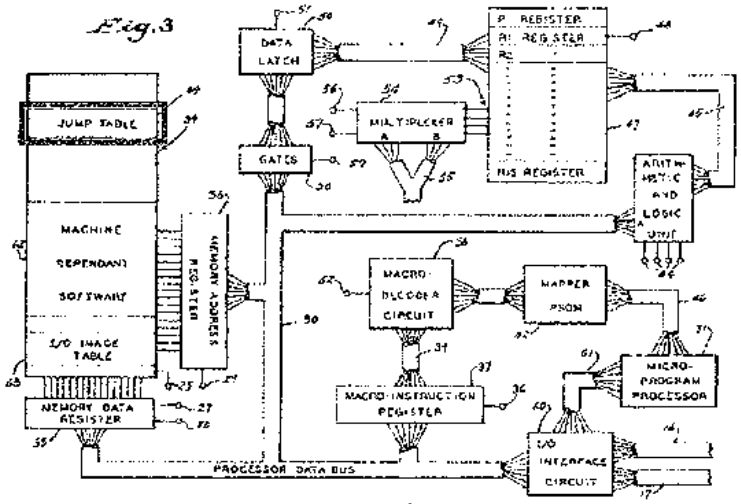
Notably, just like the '552 Patent, the Dummermuth '853 Patent discloses the problem of line shifting that occurs with deletions and insertions into executable code that contains internal references to other lines of code. The Dummermuth '853 Patent presents the solution of replacing the references normally in the jumps (GTO instructions) in an executable program with invariant references to a jump table. The Dummermuth '853 Patent explains:

RedBend0008045

> The GTO instruction (or any other conditional or unconditional jump type instruction) must identify the target memory address to which the system is to jump. It should be apparent, however, that if the target memory address is specified directly in the GTO instruction, it must be changed each time an editing function is performed which involves the shifting of controller instruction memory addresses. In other words, the editing functions such as "insert" or "delete" may change the memory address of the point, or target, in the control program to which the system is to jump thus necessitating the changing of the GTO instruction. Therefore, the target memory address of each GTO instruction is indirectly identified by a target number which accompanies the operation code and which refers to a particular location in a jump table 64. The jump table 64 is then employed to obtain the desired target memory address

(Dummermuth '853 Patent col.12 l.54 to col.13 l.3.)

As shown in FIG. 3 (below), the jump table (64) (highlighted red box) is stored in the read/write main memory (34). The Dummermuth '853 Patent explains that the jump table is not a part of the software and thus "[e]diting functions such as 'insert' and 'delete' will not, therefore, shift or otherwise change the memory addresses of the jump table contents." (*Id.* col.13 ll.4–15.)



As shown in the example of FIG. 4a (below), the jump or "GTO" instructions will appear throughout the software (62) and each is comprised of two lines, the first storing the GTO operation code and the second line storing a three octal digit jump target number, such as 200, 201, 202 . . . 217. Each target number is associated with a location in the jump table (64) and when the GTO instruction is executed, the jump target number is used to read out the contents of its associated jump table location. The contents of that jump table location would be the target memory address of the instruction and it is this address which is then loaded.

RedBend0008046

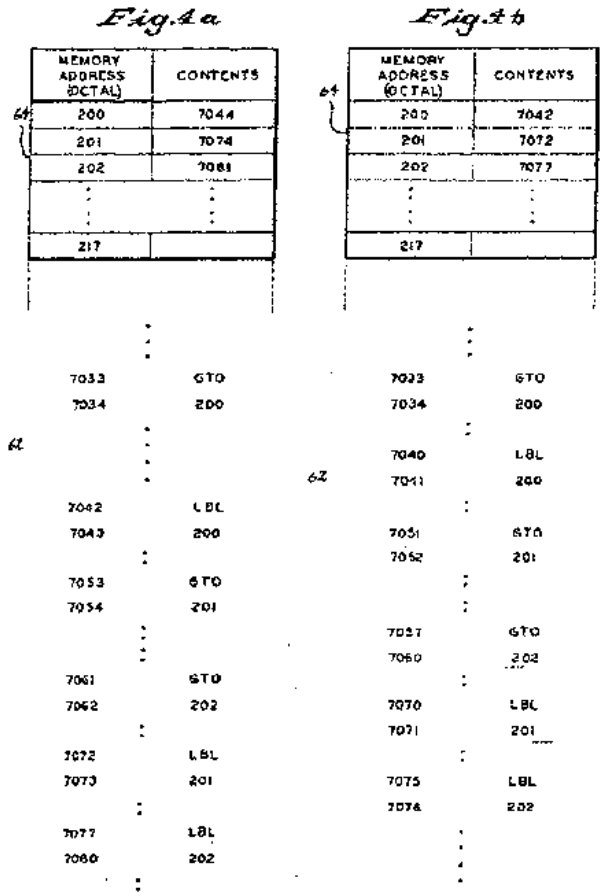


FIG. 4b shows how modification to the executable code in FIG. 4A (the instructions in FIG. 4a at address 7042 and above are shifted up) are handled, showing that the jump labels 200, 201, 202, etc. in the modified code stay the same but that the contents of the jump table will change. The Dummermuth '853 Patent states that: "It should be apparent, therefore, that if the target memory address is changed due to the execution of an editing function, the GTO instruction need not be altered, but rather, the contents of the jump table 64 is changed." (*Id.* col.13 ll.32-36.)

4. The Sadowsky '796 Patent

The Sadowsky '796 Patent qualifies as a prior art under 35 U.S.C. § 102(a) as a printed publication and as a prior art patent under 35 U.S.C. § 102(e). A copy of the Sadowsky '796 Patent is provided as **Attachment E.**

As discussed below, the Sadowsky '796 Patent, when combined with the Wetmore '713 Patent, renders obvious the claimed subject matter of all of the remaining server-side

RedBend0008047

dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) [Q5].

The Sadowsky '796 Patent, when combined with the Wetmore '713 Patent and the 1990 IBM Technical Disclosure Bulletin, also renders obvious the claimed subject matter of the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) [Q6] and subject matter of the remaining client-side dependent claims (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) [Q11].

Further, the Sadowsky '796 Patent, when combined with the Wetmore '713 Patent and the Dummermuth '853 Patent, also renders obvious the claimed subject matter of remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) [Q7] and remaining client-side dependent claims (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) [Q12].

The Sadowsky '796 Patent discloses updating software on client computers via downloading the updates from a server computer. The improvement in the Sadowsky '796 Patent is the use a special type of object (a polymorphic master object) to allow updates to occur independent of the type of client or server computer.

The Sadowsky '796 Patent discloses that while updates have occurred via new disks obtained from the vendor (as also described in the Wetmore '713 Patent), the update can also occur via download from a remote site such as via a BBS, an Internet service provider, or the Internet. Namely, in the Background, the Sadowsky '796 Patent notes:

> A user typically updates software executed by their client computer in several ways. First, the user may obtain new disks or CD-ROMs from a software vendor and load the new software from the disk into the client computer. Obtaining these disks is time consuming and frequently costs money. . . .
>
> Second, the user may obtain the updated software by accessing a host computer at a remote site and downloading the associated files from the host. The host may be, for example, a bulletin board service (BBS), the Internet, an Internet provider, such as CompuServe, or other electronic forum.

(Sadowsky '796 Patent col.1 ll.12-27)

Further, in the detailed description of the preferred embodiments, the Sadowsky '796 Patent explains:

RedBend0008048

> To update software, the server 102 operates as a passive host server. The server 102 may be, for example, a Compuserve (TM) host, a conventional Web page, or a Novell file server. . . .
>
> The communication channel 110 may be, for example, an Internet. The servers 102 and the clients 104 each execute programs stored in the server memory 106 and the client memory 108, respectively, as described later herein.

(*Id.* col.4 ll.37-49.)

Thus, the Sadowsky '796 Patent shows that it was known at the time of its filing (June 14, 1996) to use the Internet or other communications networks for sending program updates from servers to client computers at remote locations.

5. **The Coppieters Article**

The Coppieters Article qualifies as a prior art publication under 35 U.S.C. § 102(b). A copy of the Coppieters Article is attached as **Attachment F.**

As discussed below, the Coppieters Article, when combined with the Wetmore '713 Patent, renders obvious the claimed subject matter of all of the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) [Q5].

The Coppieters Article, when combined with the Wetmore '713 Patent and the 1990 IBM Technical Disclosure Bulletin, also renders obvious the claimed subject matter of the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) [Q6] and subject matter of the remaining client-side dependent claims (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) [Q11].

Further, the Coppieters Article, when combined with the Wetmore '713 Patent and the Dummermuth '853 Patent, also renders obvious the claimed subject matter of remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) [Q7] and remaining client-side dependent claims (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) [Q12].

The Coppieters Article discloses in its first paragraph the following:

> Binary file comparison is useful for many applications. One example is sending updates of large files over a communications line. Instead of sending a complete update each time, you could send the complete file once, then create a diff file containing the differences between the original file and the updated file..

RedBend0008049

> At the receiving side, this diff file could be used to update the original file. Creating a diff file is processor and memory intensive. Under DOS, such a process can easily exceed the 640K limit. On the other hand, using a diff file to update the file is very lightweight and fast. In such cases, it maybe desirable to create the diff file on another platform and use the resulting file under DOS.

Coppieters Article 34.

Thus, the Coppieters Article shows that it was known at the time, in 1995, to send "lightweight" diff files with updates or patches of programs over communications lines so that the client can update the original file.

6. Combining The Secondary References With The Primary Reference Of The Wetmore '713 Patent

The present request for reexamination sets forth substantial new questions of patentability based on anticipation of certain claims under 35 U.S.C. § 102 in view of the Wetmore '713 Patent, and of the remaining claims based on obviousness under 35 U.S.C. § 103. The substantial new questions based on obviousness are based on one primary reference, the Wetmore '713 Patent, which when taken alone, or combined with one or more secondary references, renders obvious all of the claims of the '552 Patent.

With respect to the substantial new questions based on obviousness, it should be noted that the '552 Patent was examined by the Patent Office well prior to the Supreme Court's 2007 decision in *KSR International Co. v. Teleflex, Inc.*, 550 U.S. 398 (2007). In *KSR*, the Court rejected the traditional, more rigid test of obviousness that required proof of a "teaching, suggestion, or motivation" to combine the references (the so-called "TSM" test). In doing so, the Supreme Court returned to fundamental principles from earlier cases including the notion that "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." *Id.* at 416.

As to predictability, the Court explained: "If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability." *Id.* at 417. Thus, in determining obviousness, one "must ask whether the improvement is more than the predictable use of prior art elements according to their established functions." *Id.*

The Court also explained that "common sense" can be used in an obviousness analysis: "Common sense teaches, however, that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.* at 420. Moreover, the Court explained:

RedBend0008050

> When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense. In that instance the fact that a combination was obvious to try might show that it was obvious under § 103.

Id. at 421.

KSR therefore expanded the sources of information for a flexible obviousness inquiry to include items such as: design incentives; interrelated teachings of multiple patents; any need or problem known in the field of endeavor at the time of invention and addressed by the patent; and the background knowledge, creativity, and common sense of the person of ordinary skill. *Id.* at 418-21.

Accordingly, yet another reason for finding that a substantial new question of patentability is presented herein as to the '552 Patent is due to the "common sense" and "obvious to try" standards of obviousness under *KSR*, which do not require express teachings or suggestions in the references.

The combining of the Wetmore '713 Patent with one or more of the secondary references is now discussed below.

a. **The 1990 IBM Technical Disclosure Bulletin**

The 1990 IBM Technical Disclosure Bulletin is being relied upon to the extent the Wetmore '713 Patent is not found to expressly or inherently disclose replacing "substantially" all of the references with labels that are jumps to the table in RAM.

One of ordinary skill in the art would have been motivated to combine the Wetmore '713 Patent with the 1990 IBM Technical Disclosure Bulletin since both of these references are in the exact same field of patching or updating executable programs normally run from a ROM, and both address updating in the same way – by replacing internal references with jump labels to allow for updating of the executable program. Thus, to the extent that the Wetmore '713 Patent does not expressly or inherently disclose replacing "substantially" all of the references with invariant jumps to a table, the 1990 IBM Technical Disclosure Bulletin expressly teaches replacing all or substantially all of the references with jumps to a table in RAM. Unlike the Wetmore '713 Patent, the 1990 IBM Technical Disclosure Bulletin does not include "reference to entry points" but rather just replaces its entry points with indirect jumps and indirect calls. (1990 IBM Technical Disclosure Bulletin at 273-74.)

RedBend0008051

Thus, a person of ordinary skill in the art, considered to have knowledge of all known prior art techniques for patching or updating of code normally run only from ROM, would be taught by the 1990 IBM Technical Disclosure Bulletin to replace all (or substantially all) of the references in the program of the Wetmore '713 Patent, since either jump replacement technique can be used and replacing all or substantially all references would result in more freedom to update other lines of code and/or result in faster execution instead of sometimes jumping to entry points from a "reference to an entry point" as disclosed in the Wetmore '713 Patent.

Further, a person of ordinary skill in the art could easily and predictably implement the replacement technique of the 1990 IBM Technical Disclosure Bulletin in the Wetmore '713 Patent by simply replacing all, or substantially all, of the internal jumps in ROM to jumps to tables in RAM, and would see the benefit of doing so to increase flexibility in modifications of code (already taught by both the Wetmore '713 Patent and the 1990 IBM Technical Disclosure Bulletin) and to eliminate extra processing steps. Moreover, since the technique of the 1990 IBM Technical Disclosure Bulletin improves the updating of code normally run from ROM, and a person of ordinary skill in the art would recognize that it would improve the same updating of code in ROM in the Wetmore '713 Patent in the same way, it would have been obvious, and well within the skill of one of ordinary skill in the art, to use the replacement technique of the 1990 IBM Technical Disclosure Bulletin with the Wetmore '713 Patent.

b. <u>The Dummermuth '853 Patent</u>

One or ordinary skill in the art would have be motivated to apply the teaching of the Dummermuth '853 Patent to the Wetmore '713 Patent since both of these prior art references are in the same field of updating executable code for a processor, and both relate to the problem of "implementing 'jump' instructions in a system of the type in which the memory address of instructions may be readily changed during development of a control program." (Dummermuth '853 Patent col.1 ll.51-55.)

Like the Wetmore '713 Patent and the 1990 IBM Technical Disclosure Bulletin, the Dummermuth '853 Patent provides the common solution of using a jump table in memory to store the memory address of a target specified by a jump instruction to allow for later modifications to the executable program. Thus, a person of ordinary skill in the art, considered to have knowledge of all known prior art techniques for updating code using jump tables, would readily look to and apply the teachings of the Dummermuth '853 Patent to the Wetmore '713 Patent.

RedBend0008052

A person of ordinary skill in the art could easily and predictably implement the teaching of the Dummermuth '853 Patent in the Wetmore '713 Patent to replace all or substantially all of the target memory addresses with jumps to a jump table in order to solve the known "insertion/deletion" problem of the '552 Patent that was expressly recognized as far back as 1976 when the Dummermuth '853 Patent was filed:

> It should be apparent, however, that if the target memory address is specified directly in the GTO instruction, it must be changed each time an editing function is performed which involves the shifting of controller instruction memory addresses. In other words, the editing functions such as "insert" or "delete" may change the memory address of the point, or target, in the control program to which the system is to jump thus necessitating the changing of the GTO instruction. Therefore, the target memory address of each GTO instruction is indirectly identified by a target number which accompanies the operation code and which refers to a particular location in a jump table 64. The jump table 64 is then employed to obtain the desired target memory address.

(*Id.* col.12 l.57 to col.13 l.3.)

One of ordinary skill in the art would be motivated to replace all (or substantially all) internal jumps to jumps to tables in RAM, and would see the benefit of doing so to eliminate extra jumps and thus extra processing steps. Using the replacement technique of the Dummermuth '853 Patent with the Wetmore '713 Patent would be obvious and well within the skill of one of ordinary skill in the art, since the technique of the Dummermuth '853 Patent improves the patching of code by replacing all or substantially all of the references with jumps to tables, and a person of ordinary skill in the art would recognize that it would improve the same type of patching of code in the Wetmore '713 Patent in the same way.

c. The Sadowsky '796 Patent

One of ordinary skill in the art would have be motivated to apply the teaching of the Sadowsky '796 Patent of sending program updates via a communications network or the Internet to the Wetmore '713 Patent since both of these prior art references are in the same field of providing updated code for a processor to a client.

The Sadowsky '796 Patent discloses the well-known fact at the time that files could be sent over communications lines and using that fact to transfer the updated difference files to a remote client computer, whereas the Wetmore '713 Patent only discloses updating the files via use of replacement disks. The Sadowsky '796 Patent discloses that while updates can occur via new disks obtained from the vendor (as also described in the Wetmore '713 Patent), the update

RedBend0008053

can also occur via downloading from a remote site such as a BBS, an Internet service provider, or the Internet. The Sadowsky '796 Patent explains:

> A user typically updates software executed by their client computer in several ways. First, the user may obtain new disks or CD-ROMs from a software vendor and load the new software from the disk into the client computer. Obtaining these disks is time consuming and frequently costs money. . . .
>
> Second, the user may obtain the updated software by accessing a host computer at a remote site and downloading the associated files from the host. The host may be, for example, a bulletin board service (BBS), the Internet, an Internet provider, such as CompuServe, or other electronic forum.

(Sadowsky '796 Patent col.1 ll.12-27)

In the detailed description of the preferred embodiments, the Sadowsky '796 Patent also explains:

> To update software, the server 102 operates as a passive host server. The server 102 may be, for example, a Compuserve (TM) host, a conventional Web page, or a Novell file server. . . .
>
> The communication channel 110 may be, for example, an Internet. The servers 102 and the clients 104 each execute programs stored in the server memory 106 and the client memory 108, respectively, as described later herein.

(*Id.* col.4 ll.37-49.)

A person of ordinary skill in the art, considered to have knowledge of all known prior art techniques for sending difference files, would easily and readily look to and apply the teachings of the Sadowsky '796 Patent to the Wetmore '713 Patent in order to eliminate the need to send replacement or update disks to the remote user. Thus, a person of ordinary skill in the art could easily and predictably implement the teaching of the Sadowsky '796 Patent with the Wetmore '713 to send the compact difference result via a communications network, or the Internet, just as easily as one could load the compact difference result onto new disks or CD-ROMs or the like.

One of ordinary skill in the art would be motivated to do so, and would see the benefit of doing so to where users had network or Internet connectivity or the software provider wanted to save costs of sending out update disks. Moreover, since the technique of using a network for patch distribution in the Sadowsky '796 Patent provides a well-known alternate way of distribution, and a person of ordinary skill in the art would recognize that the patches in the Wetmore '713 Patent could be sent in the same way, using the transmission technique of the Sadowsky '796 Patent with the Wetmore '713 Patent would be obvious and well within the skill of one of ordinary skill in the art.

RedBend0008054

d. The Coppieters Article

The Coppieters Article was cited to the USPTO by the applicant of the '552 Patent but was never applied by the Examiner, who had rejected all of the dependent claims regarding sending the compact difference result over a communications network (*e.g.*, claim 2) such as the Internet (*e.g.*, claim 3) based on inherency of these features in the two Japanese published applications cited for anticipation of all of the originally-filed claims.

One of ordinary skill in the art would have be motivated to apply the teaching of the Coppieters Article of sending program updates via a communications network to the Wetmore '713 Patent since both of these prior art references are in the same field of updating code for a processor.

The Coppieters Article discloses the well-known fact that, at the time, binary files could be sent over communications lines and using that fact to transfer the updated difference files to a remote client computer, whereas the Wetmore '713 Patent only discloses updating the files via use of replacement disks. However, a person of ordinary skill in the art, considered to have knowledge of all known prior art techniques for sending difference files, would easily and readily look to and apply the teachings of the Coppieters Article to the Wetmore '713 Patent in order to eliminate the need to send replacement or update disks to the remote user. And while the "Internet" is not expressly mentioned as a type of communications network in the Coppieters Article, it certainly is a communications network, and certainly existed at the time of the Coppieters Article as evidenced on page 36 of the Coppieters Article which included an advertisement for distributing software either by CDs or the Internet: "Put protected software on CDs or the Internet."

Thus, a person of ordinary skill in the art could easily and predictably implement the teaching of the Coppieters Article with the Wetmore '713 to send the compact difference result via a communications network, such as the Internet, just as easily as one could load the compact difference result onto disks or CDs or the like.

One of ordinary skill in the art would be motivated to do so, and would see the benefit of doing so where users had network or Internet connectivity or the software provides wanted to save costs of sending out update disks. Moreover, since the technique of using a network for patch distribution in the Coppieters Article provides a well-known alternate way of distribution, and a person of ordinary skill in the art would recognize that the patches in the Wetmore '713 Patent could be sent in the same way, using the transmission technique of the Coppieters

RedBend0008055

Article with the Wetmore '713 Patent would be obvious and well within the skill of one of ordinary skill in the art.

C. The Substantial New Questions Of Patentability Were Not Considered During The Original Examination Of The '552 Patent

As noted above, only one of the prior art references (the Coppieters Article) was of record during the original examination of the '552 Patent, but it was never applied. The remaining prior art references are all new and were never before the examiner during prosecution of the '552 Patent Therefore, as can be seen from the prosecution of the '552 Patent, none of the prior art relied on herein was previously considered during the original examination of the '552 Patent.

1. The Original Examination Of The '552 Patent

The patent application from which the '552 Patent issued, Serial No. 09/376,512, was filed in the U.S. on August 18, 1999, and claimed priority to Israeli Patent Application No. 125846, filed on August 18, 1998.

A Notice of Missing Parts was issued on September 8, 1999. A Response to the Notice of Missing Parts was mailed November 1, 1999, with the applicant providing the filing fee, the executed declaration, an assignment to Emony Ltd., and a certified copy of the Israeli priority document.

On March 14, 2000, an Information Disclosure Statement was submitted which included: three published patent applications, WO 93/00633 PCT (07 January 1993), WO 97/12508 PCT (10 April 1997) and EP 0 472 812 A (04 March 1992); the international search report for the PCT application of the Israeli priority application; and one article, the Coppieters Article. No comments were made about the cited prior art other than to note that the references had been cited in a communication from a corresponding foreign application, and that with regard to the German language European patent EP 0 472 812 A, no English translation was available but that the International Search Report had designated the document as "defining the general state of the art which is not considered to be of particular relevance."

A first official action on the merits issued on October 2, 2001, and rejected all of the claims as being anticipated by JP 404242829A to Okuzumi *et al.* (the "Okuzumi '829 Publication"), the Abstract of which provided:

> PURPOSE: To reflect update of one program on the other by automatically generating control information which reflects update, to which the first program

RedBend0008056

> is subjected, on the second program to recognize the difference between two programs independently of the sequence number. CONSTITUTION: When comparing an old program 1 and a first program 2a obtained by updating the old program 1 with each other to generate update information, an update information generating part 3 takes in both programs to generate an old order table and a new order table where instruction statements are arranged in order in accordance with prescribed relations among character strings of instruction statements.; Instruction statements of the old program 1 and those of the first program 2a are compared with each other in accordance with orders indicated in old and new order tables, and deletion, insertion, or non-change in the old program 1 is discriminated in accordance with a prescribed condition of relations between both instruction statements. The result is referred to generate a first update information table 4a. An update reflection processing part 5 collates them to generate the update reflection output.
>
> (http://v3.espacenet.com/publicationDetails/biblio?FT=D&date=20011029&DB=EPODOC&locale=en_EP&CC=JP&NR=4242829A&KC=B2)

The examiner also rejected all of the claims as being anticipated by JP 05091550A To Kenji *et al.* (the "Kenji '550 Publication"), the Abstract of which provided:

> PURPOSE: To minimize the loading quantity of programs and to shorten a loading time by loading a difference program. CONSTITUTION: The contents of a new program file 1 are compared with that of an old program file 2 in each address by a difference program forming means 3 such as a personal computer, and a difference exists between both the contents, the address concerned and its contents are extracted from the file 1 to form a difference program file 4. Then the contents of the file 4 are loaded to a RAM 5. It is preferable to send a difference program to a data processor by means of an on-line. Consequently the loading quantity of programs can be minimized and the loading time at the time of substituting programs can be shortened. In the case of loading down programs to many data processors, time to be required for completing program substitution in all the processors can be shortened.
>
> (http://v3.espacenet.com/publicationDetails/biblio?DB=EPODOC&adjacent=true&locale=en_EP&FT=D&date=19930409&CC=JP&NR=5091550A&KC=A)

As to the independent claims, the examiner used claim 1 as the basis for all of the independent claims and found that the Okuzumi '829 Publication and Kenji '550 Publication each separately the disclosed steps (A), (B) and (C) provided in the originally-filed claim:

> 1 A method for generating a compact difference result between an old program and a new program; each program including reference entries that contain reference that refer to other entries in the program; the method comprising the steps of:
>
> (A) scanning the old program and for substantially each reference entry perform steps that include:

RedBend0008057

(I) replacing the reference of said entry by a distinct label mark, whereby a modified old program is generated;

(B) scanning the new program and for substantially each reference entry perform steps that include:

(I) replacing the reference of said entry by a distinct label mark, whereby a modified new program is generated;

(C) generating said difference result utilizing directly or indirectly at least said modified old program and modified new program.

As to the dependent claims of transmitting the difference result over a network (*e.g.*, claim 2 and other like dependent claims), the examiner found that this was inherent based on (1) the last sentence of the "constitution" of the Okuzumi '829 Publication and the purpose to enable automatic update reflection outputs, or (2) the third sentence of the "constitution" of the Kenji '550 Publication and the purpose to minimize loading quantity.

As to the dependent claims of transmitting the difference result over the Internet (*e.g.*, claim 3 and other like dependent claims), the examiner also found that this was inherent based on for the same reasons as claim 2 in the Okuzumi '829 Publication or the Kenji '550 Publication.

Finally, as to the dependent claims of storing the difference result on a storage medium (*e.g.*, claim 4 and other like dependent claims), the examiner found that was (1) inherent in view of the Okuzumi '829 Publication or (2) inherent in view of the Kenji '550 Publication to enable future comparison and updates of data.

The examiner also noted two additional references, not specifically cited, that were considered pertinent to the applicant's invention; namely, U.S. patents to Marron (5,359,730) and Hill (5,761,649), which the examiner noted as providing for updating data from a remote site with just the changed portions.

On April 10, 2002, a Notice of Abandonment was mailed for failure of the applicant to respond and included an Interview Summary dated April 4, 2002, stating that during a telephone interview, "the applicant indicated that no response had been filed in response to the official action dated 10/2/01." On May 8, 2002, applicant submitted a Petition to Revive An Unintentionally Abandoned under 37 C.F.R. § 1.137(b), along with Response to the Office Action.

In response to the Office Action, certain of the claims were amended and arguments were presented to distinguish the claims over the cited and applied prior art. Namely, all of the preambles of all of the independent claims to programs (*i.e.*, claims 1, 5, 8, 14, 18, 21, and 35)

RedBend0008058

were amended to add that programs at issue were "executable" programs. For example, claim 1 as amended now recited:

> 1. [Amended] 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) 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, whereby a modified old program is generated;
>
> (b) scanning the new program and for substantially each reference entry perform steps that include:
>
> (i) replacing the reference of said entry by a distinct label mark, whereby a modified new program is generated;
>
> (c) generating said difference result utilizing directly or indirectly at least said modified old program and modified new program.

The applicant noted that "independent claims 1, 5, 8, 12, 14, 18, 21, and 25 were amended to recite that the respective claims define executable programs" and that support for the amendment could be found, for example, on page 4, lines 16, 24, the exemplary program provided in page 2, line 2 (Office 97 package) and that the detailed description (with reference to FIGS. 2A-2B) described a machine-instruction like structure.

In its initial remarks, the applicant stressed that present invention aimed to improve over prior art difference extraction utilities, already known in the art, but applied to "executable programs" to generate a better, compact result. Thus, applying a standard diff utility to executable programs (*e.g.*, Windows 97 and updates of Windows 97), would result in a large diff file. The applicant then explained that the prior are diff extraction utilities were designed for source code:

> General diff extraction utilities (see page 2, lines 14 and 15) were originally designed for extracting differences between two versions of a source program (referred to also as 'program source' or as 'source'). Sources are the original text files that programmers write in some formal language known as 'programming language.' Such sources are purely symbolic in the sense that they do not mention actual physical location of other elements in the source nor their absolute sequential location. Rather, any of required references in a source are made through symbolic names which themselves are part of the source.

The applicant went on to explain:

RedBend0008059

> A major problem arises when applying these methods to executable program files aiming to extract the difference at the byte-level, regarding the files as a list of data bytes rather then list of text lines. The problem arises from the fact that executable programs are generated from sources and in that process many references are inserted into these executable files. These references do not refer symbolically to other location of the program, as may be the case in source files, but they refer to addresses - sequential locations in the program file. When regenerating an executable file from a modified source file, not only the actual changes are being reflected in the executable file, also the majority if not all the inserted references are modified as well since their referred addresses have changed their location in the executable file as a result of the actual changes. This phenomenon leads to a significant increase in the amount of differences.

The applicant then distinguished over the Okuzumi '829 Publication as explicitly mentioning "source" and contained a step of sorting a "statement" which the applicant alleged was a common name for an element of a source program. By contrast, the applicant argued that claim 1 defined an "executable program" and that "obtaining a compact difference result in the manner defined in the claims of the present application is not suggested even remotely in Okuzumi, which is not surprising considering the fact that in source programs there is no need to generate a compact result since the difference result is a priori compact."

The applicant also explained: "In extracting diff between 2 versions of executable files as defined in amended Claim 1, there is no source involved, and neither statements, nor any textual or other symbolic representation of the program even exist."

As to the dependent claims, the applicant simply asserted that "Claims 2-4 are dependent directly or indirectly on Claim 1, and therefore are also not anticipated by Okuzumi." The same was asserted for the other like dependent claims.

As to the "data table" claims, the applicant argued that they were basically the same as the "program" claims:

> Claims 35 to 68 are basically similar to claims 1 to 34, respectively, except for the fact that they recite data table instead of executable program. Data table is discussed on page 4, line 9 of the application and do not embrace source code as in Okuzumi. It is accordingly submitted that Claims 39-41, 42-44, 48-50, 52-54, 55-57 are not anticipated by-Okuzumi for the reasons discussed in detail above with reference to Claims 1 to 3, 8-10, 14-16, 18-20, and 21-23.

The applicant next turned to the rejections based on the Kenji '550 Publication, and explained that the Kenji '550 Publication relates to minimizing the time it takes to update a program already loaded in RAM by reducing the update from sending the whole program to just sending the differences with the addition that handles modules that expand their original size.

RedBend0008060

The applicant asserted that in the Kenji '550 Publication, there is an empty space left between modules in the RAM and accordingly, if the "updated" (new) module is larger in size than the module that currently resides in the RAM (old module), the new module can fit the memory without affecting other modules.

The applicant then explained that the Kenji '550 Publication did not relate to obtaining a compact difference by applying a pre-processing step before applying the difference operation (see for example Claim 1, steps (a) and (b) that precede step (c)). The applicant asserted that in the Kenji '550 Publication, the "compactness" was based on the a priori available space between the modules, while leaving space between modules in order to enable accommodation of updated modules is completely irrelevant for the present invention. Thus, the applicant argued that claim 1 was not anticipated by the Kenji '550 Publication, and claims 2-4 are dependent directly or indirectly on claim 1 and therefore were also not anticipated by the Kenji '550 Publication.

In August 13, 2002, a Decision on Petition was issued granting the petition to revive. On August 27, 2002, a Notice of Allowabilty was issued, with the examiner issuing the following reasons for allowance, citing to a new reference to U.S. Patent No. 5,832,520 to Miller ("the Miller '520 Patent"):

> The following is an examiner's statement of reasons for allowance: the applicant's amendment and arguments have overcome the previous rejection in view of Okuzumi and Kenji. The closest prior art reference of record currently is the newly cited patent to Miller (5,832,520), which teaches a method and system of generating a difference result between an old program (which may be an executable program, see the abstract and col.5 lines 17-25). Miller's system also scans the old program and creates a modified old program, col.3 lines 1-10 and col.6 lines 34-44. The modified old file is generated prior to generating the difference file, see the cited portions above. However, Miller does not teach or suggest generating a modified new file and using the modified new file and the modified old file to generate a difference result.

On November 27, 2002, the applicant submitted "Comments On Statement Of Reasons For Allowance" to distinguish over the Miller '520 Patent as using an "index" or "hash" table as opposed to generating a "modified old program" as follows:

> Applicants agree that Miller does not disclose or suggest generating a modified new file and using the modified new file or modified old file to generate a difference result. Applicants, however, do not agree with the Examiner's contention that Miller creates modified old programs in the sense of the invention.
>
> In support of this contention, the Examiner refers to Column 3, lines 1-10 and Column 6, lines 34-44. In Miller, the teachings of Column 3, lines 1-10 are

RedBend0008061

> simply that in addition to an old file, an index or hash table is created (see *e.g.*, lines 2-5), so as to facilitate searching for character strings from the new file. Thus, according to Miller, auxiliary data is created in addition to the old file and in contrast to the invention (in accordance with the aspect defined in claim 1) where the modified old file is generated and is used later for generating the compact difference result.
>
> Note, that the index or hashing table in Miller are not always created (see Column 3, lines 7-10). In contrast, in accordance with the invention, the generation of modified old programs or modified old data table (depending on the aspects of the invention), is an obligatory step for the generation of the compact difference result.
>
> These observations are also reinforced by referring to the other reference provided by the Examiner (Column 6, lines 34-44), where it readily arises that Miller refers to the generation of auxiliary index data structure, which is only optional (see, for example, Column 6, lines 40-44).
>
> Note also that the purpose of Miller's index is generally known per se, *i.e.*, to decrease the search time for the text strings in the old file (see Column 6, line 40). This is not the case for the modified old program, for example, defined in claim 1, which serves for generating the compact difference result.
>
> As mentioned above, Miller does not disclose the generation of modified old file, and a fortiori, not in the manner recited, for example, in step (a)(I), *i.e.*, "replacing the reference of said entry by distinct label mark."

Also on November 27, 2002, the applicant submitted an Information Disclosure Statement providing a translation of the claims of the reference EP 0472812,` which had been submitted in the Information Disclosure Statement of March 14, 2000.

The issue fee was paid on December 2, 2002. On February 10, 2003, the examiner issued a Supplemental Notice of Allowability, stating that the Miller '520 Patent did sufficiently disclose a modified old program:

> The following is an examiner's statement of reasons for allowance: the reasons for allowance are the same as presented in the previous action. The applicant indicates that Miller does not generate a modified old program; however, with the creation of Miller's Text String Index and utilization of the index for subsequent searching of the old program, the feature suffices as a "modified old program" (again see col.6 lines 34-66). Therefore, as indicated in the previous action, Miller is still considered to generate a modified old program. However, Miller does not teach or suggest generating a modified new file and using the modified new file along with the modified old file to generate a difference result. The applicant further supplied an English translation of the claims of EP 0472812. Therefore, the claims are the only portion of the reference considered. The present claims are allowable over the claims of '812 for at least the same reasons they are allowable over Miller.

The application then issued on April 8, 2003.

RedBend0008062

2. The Substantial New Question Of Patentability Based On Anticipation Was Not Considered During The Original Examination

The Wetmore '713 Patent was not considered during prosecution of the '552 Patent. During prosecution of the '552 Patent, the Patent Office rejected the original claims solely in view of the Okuzumi '829 Publication and Kenji '550 Publication. While both publications related to the use of a compact difference result between and old and modified program to update the old program, neither publication, according to the Patent Owner, disclosed a "pre-processing" step whereby the old and new program were first modified before they were compared to generate the difference result. The Patent Owner also alleged that the Okuzumi '829 Publication did not disclose updating of an "executable" program, only updating of source code. The Wetmore '713 Patent, however, discloses these very features alleged to be missing from the Japanese publications cited during prosecution, namely the pre-processing step and the updating of executable programs. Thus, the anticipation of the claims of the '552 Patent by the Wetmore '713 Patent clearly presents a substantial new question of patentability that was not considered during the original examination of the '552 Patent.

3. The Substantial New Questions Of Patentability Based On Obviousness Were Not Considered During The Original Examination

The substantial new questions of patentability based on obviousness rely on references that were not cited or considered by the examiner during the original examination of the '552 Patent (the Wetmore '713 Patent, the 1990 IBM Technical Disclosure Bulletin, the Dummermuth '853 Patent, and the Sadowsky '796 Patent), as well as one prior art reference (the Coppieters Article) that was made of record but was never applied against any of the claims. These prior art references go well beyond what was disclosed in the prior art applied by the Patent Office during prosecution.

In particular, the Wetmore '713 Patent discloses the very features alleged to be missing from the two cited Japanese publications, namely pre-processing of the old and new programs and the updating of executable programs. The 1990 IBM Technical Disclosure Bulletin is directed to the same type of system as the Wetmore '713 Patent and is likewise directed to executable programs and requires pre-processing of the old and new programs before being loaded into memory.

The Dummermuth '853 Patent, unlike any of the prior art of record, expressly discloses the problem of line shifting that occurs with deletions and insertions into executable program

RedBend0008063

code and provides the same solution to the problem that the '552 Patent relies so heavily upon, namely that the internal references in the program code are replaced by invariant references.

Finally, the Sadowsky '796 Patent (and the Coppieters Article) both disclose the well-known step of transmitting difference results over a communications network, with the Sadowsky '796 Patent disclosing updating software using a host server such as a Compuserve™ host, a conventional Web page, or a file server, and that the communications channel may be the Internet. The Sadowsky '796 Patent also discloses that programs can be updated either by new disks (as is disclosed in the Wetmore '713 Patent), via a BBS, an Internet service provider, or the Internet.

Therefore, the particular combinations raised by the substantial new questions of patentability were also not previously considered during prosecution of the '552 Patent.

IV. EXPLANATION OF PERTINENCE AND MANNER OF APPLYING THE CITED PRIOR ART (37 C.F.R. § 1.510(b)(2))

A. Summary Of The Claimed Subject Matter Of The '552 Patent For Which Reexamination Is Sought

As noted above in Part II.A.1.a, the claims of the '552 Patent subject to the present request for reexamination can be broken into "server-side" sets (grey boxes) and client-side" sets (white boxes):

Set		"program" claims		"data table" claims	
		method	system	method	system
1	Generating compact difference result using **distinct label marks** (server-side)	**1** 2-4, 27*	**14** 15-17	**35** 35-38, 61*	**48** 49-51
2	Generating compact difference result using **invariant references** (server-side)	**8** 9-11, 28*	**21** 22-24	**42** 43-45, 62*	**55** 56-58
3	Performing update using **distinct label marks** (client-side)	**5** 6-7	**18** 19-20, 26	**39** 40-41	**52** 53-54
4	Performing update using **invariant references** (client-side)	**12** 13, 29-31	**25** 32-34	**46** 47, 63-65	**59** 60, 66-68

* Claims 27, 28, 61, and 62 are multiple dependent product-by-process claims.

RedBend0008064

As also noted above, since many of the claims are nearly identical to one another, the analysis of all of the independent claims for substantial new questions of patentability can be simplified into analysis of four representative claims from each set as follows:

- Set 1: Server-side claim 1 – same analysis applies to claims 14, 35, and 48
- Set 2: Server-side claim 8 – same analysis applies to claims 21, 42, and 55
- Set 3: Client-side claim 5 – same analysis applies to claims 18, 39, and 52
- Set 3: Client-side claim 12 – same analysis applies to claims 25, 46, and 59

The dependent claims add insignificant subject matter to the independent claims and are of three types. First, there are dependent claims that call for sending the compact difference result results via a communications network or the Internet (dependent claims 2, 3, 6, 7, 9, 10, 15, 16, 19, 20, 22, 23, 32, 33, 36, 37, 40, 41, 43, 44, 63, 64, 59, 50, 53, 54, 56, 57, 66, and 67). Second, there are dependent claims that call for storing the compact difference result on a storage medium (dependent claims 4, 11, 13, 17, 24, 26, 31, 34, 38, 45, 47, 51, 58, 60, 65, and 68). Finally, there are multiple dependent product-by-process claims (claims 27, 28, 61, and 62) that call for a processing device having a storage medium that holds the compact difference result generated by the methods of claims 1-4, 8-11, 35-38, or 42-45.

B. Explanation Of Cited Prior Art And Its Application With Respect To The Substantial New Questions Of Patentability Based On Anticipation

1. Anticipation Of Server-Side Independent Claims 1, 8, 14,21, 35, 42, 48, And 55, And Dependent Claims 4, 11, 17, 24, 27/1, 27/4, 28, 38, 45, 51, 58, 61/35, 61/38, 62/42, And 62/45 Based On The Wetmore '713 Patent [Q1]

For the reasons set forth below, a substantial new question of patentability of the independent server-side claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48, and 55) and dependent server-side claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent under 35 U.S.C. § 102(b) is raised by the Wetmore '713 Patent [Q1]. These claims are the server-side independent claims from Sets 1 and 2, plus dependant claims relating to storing the compact difference result on a storage medium.

As discussed above, the Wetmore '713 Patent meets all of the limitations recited in these claims, which is shown in detail in the claim charts below:

RedBend0008065

Set 1: Server-Side Independent Claims 1, 14, 35, and 48:

Claim Element	Claim Element	The Wetmore '713 Patent
1. 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:	14. A system 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 system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old executable program (old non-vectorized program) and a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) includes reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.
(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, whereby a modified old program is generated;	(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, whereby a modified old program is generated;	During vectorization, the Wetmore '713 Patent scans the old program (the old non-vectorized program) and, for substantially each **reference entry** (each "jump" entry point), replaces the **reference** (the location in the ROM code in the jump entry point) of the entry by a **distinct label mark** (a table pointer with an offset), whereby a modified old program (vectorized old program) is generated. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.) **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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

RedBend0008066

Claim Element	Claim Element	The Wetmore '713 Patent
		address." ('552 Patent col.2 ll.42-45.) Under this definition, in Wetmore, all of the jump entry points in the old non-vectorized program would be all of the references. The Wetmore '713 Patent replaces each jump entry point (reference) when vectorizing the code. **Distinct label marks**: A "label" is defined in the '552 Patent be "an abstract notation of an entry which is referred by another entry of the same data table through a reference." ('552 Patent col.2 ll.48-50.) "Label marks" are shown as marks (101)-(105) in FIG. 2A. An example of a "distinct label *number*" is described in the '552 Patent: "a distinct label *number* is assigned to each label mark of P''_1. By this particular example, the distinct labels are assigned in ascending order and, as shown, labels marks (101) to (105) are assigned with the respective values 1 to 5." ('552 Patent col.10 ll.63-67 (emphasis added).) Whether the '552 Patent's use of the claim term "distinct label *mark*" is broader than, or the same scope as a "distinct label *number*," the Wetmore '713 Patent still meets this element. Namely, in the Wetmore '713 Patent, each table pointer with its offset (a numerical mark) into the vector table would be a "distinct label mark."
(b) scanning the new program and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new program is generated;	(b) scanning the new program and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new program is generated;	The process described above in (a) is repeated by the Wetmore '713 Patent for the new program (the new non-vectorized program). Namely, during vectorization, the Wetmore '713 Patent scans the new program (the new non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a distinct label mark (a table pointer with an offset), whereby a modified new program (vectorized new program) is generated. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(c) generating said difference result utilizing directly or indirectly at least said modified old program and modified new	(c) generating said difference result utilizing directly or indirectly at least said modified old program and modified new	The Wetmore '713 Patent generates the difference result (the vector patch resource) utilizing directly or indirectly at least the modified old program (vectorized old program) and modified new program (vectorized new program). The Wetmore '713 Patent discloses: "In the currently preferred embodiment of the present

RedBend0008067

Claim Element	Claim Element	The Wetmore '713 Patent
program.	program.	invention, a tool termed ROMPatch, is provided which automatically creates the vector patch resources. ROMPatch *compares the object files of two versions of the vectorized ROM code to identify routines which are different or new.* In the currently preferred embodiment, routines which are different is accomplished via a Cyclical Redundancy Check (CRC) operation. ... In any event, when all the patched routines are found, the vector patch resource is generated." (Wetmore '713, col.10 l.66—col.11 l.12 (emphasis added).)

Claim Element	Claim Element	The Wetmore '713 Patent
35. A method for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	48. A system for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old **data table** (old non-vectorized program) and a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) includes **reference entries** ("jump" entry points) that contain **references** that refer to other **entries** in the data table. **Data table**: The '552 Patent defines a "data table" as "a table of entries, each may have a different size." ('552 Patent col.2 ll.33-34.) The '552 Patent also explains that "a data table can be an executable program either as a loaded program in machine-memory or as an executable-file. In this example, entries are individual machine instructions of the program or the individual data elements used by the program." ('552 Patent col.2 ll.61-65.) The Patentee also provided its claim construction for "data table" as "a table of entries, where an entry is an addressable unit within the data table. An executable program is one example of a data table. *See, e.g.*, '552 Patent 2:33-36; 2:61-63." (Edwards Decl. Exh. A.) Thus, the ROM based executable code of the Wetmore '713 Patent qualifies as a data table. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) The Patentee also provided its claim construction for "reference entry" as "An addressable unit containing data that includes a reference. *See, e.g.*,

RedBend0008068

Claim Element	Claim Element	The Wetmore '713 Patent
		'552 Patent 2:35-36; 2:46-47." (Edwards Decl. Exh. A.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) The Patentee also provided the identical claim construction for "reference" as "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. *See, e.g.*, '552 Patent 2:42-45." (Edwards Decl. Exh. A.) Under this definition, in the Wetmore '713 Patent, all of the jump entry points in the old non-vectorized program would be all of the references. **Entry:** The '552 Patent states in the Glossary that "a data table includes entries, each of which is an addressable unit that contains data." ('552 Patent col.2 ll.35-36.)
(a) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified old data table is generated;	(a) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified old data table is generated;	During vectorization, the Wetmore '713 Patent scans the old data table (the old non-vectorized program) and for substantially each **reference entry** (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a **distinct label mark** (a table pointer with an offset), whereby a modified old data table (vectorized old program) is generated. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to

RedBend0008069

Claim Element	Claim Element	The Wetmore '713 Patent
		labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) Under this definition, in Wetmore, all of the jump entry points in the old non-vectorized program would be all of the references. The Wetmore '713 Patent replaces each jump entry point (reference) when vectorizing the code. **Distinct label marks**: A "label" is defined in the '552 Patent to be "an abstract notation of an entry which is referred by another entry of the same data table through a reference." ('552 Patent col.2 ll.48-50.) "Label marks" are shown as marks (101)-(105) in FIG. 2A. An example of a "distinct label *number*" is described in the '552 Patent: "a distinct label *number* is assigned to each label mark of $P"_1$. By this particular example, the distinct labels are assigned in ascending order and, as shown, labels marks (101) to (105) are assigned with the respective values 1 to 5." ('552 Patent col.10 ll.63-67 (emphasis added).) Whether the '552 Patent's use of the claim term "distinct label *mark*" is broader than, or the same scope as a "distinct label *number*," the Wetmore '713 Patent still meets this element. Namely, in the Wetmore '713 Patent, each table pointer with its offset (a numerical mark) into the vector table would be a "distinct label mark."
(b) scanning the new data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new data table is generated;	(b) scanning the new data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new data table is generated;	The process described above for (a) is repeated by the Wetmore '713 Patent for the new data table (the new non-vectorized program). Namely, during vectorization, the Wetmore '713 Patent scans the new data table (the new non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of said entry by a distinct label mark (a table pointer with an offset), whereby a modified new data table (vectorized new program) is generated. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)

RedBend0008070

(c) generating said difference result utilizing directly or indirectly at least said modified old data table and modified new data table.	(c) generating said difference result utilizing directly or indirectly at least said modified old data table and modified new data table.	The Wetmore '713 Patent generates the difference result (the vector patch resource) utilizing directly or indirectly at least the modified old data table (vectorized old program) and modified new data table (vectorized new program). The Wetmore '713 Patent discloses: "In the currently preferred embodiment of the present invention, a tool termed ROMPatch, is provided which automatically creates the vector patch resources. ROMPatch *compares the object files of two versions of the vectorized ROM code to identify routines which are different or new*. In the currently preferred embodiment, routines which are different is accomplished via a Cyclical Redundancy Check (CRC) operation. … In any event, when all the patched routines are found, the vector patch resource is generated." (Wetmore '713 Patent col.10 l.66—col.11 l.12 (emphasis added).)

Set 2: Server-Side Independent Claims 8, 21, 42, And 55:

Claim Element	Claim Element	The Wetmore '713 Patent
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:	21. A system 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 system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old executable program (old non-vectorized program) and a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain reference that refer to other entries in the program.
(a) generating a modified old program utilizing at least said old program;	(a) generating a modified old program utilizing at least said old program;	During vectorization, the Wetmore '713 Patent generates a modified old program (vectorized old program) utilizing at least the old program (the old non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17, 18-31; FIG. 4.)

RedBend0008071

Claim Element	Claim Element	The Wetmore '713 Patent
(b) generating a modified new program utilizing at least said new program,	(b) generating a modified new program utilizing at least said new program,	During vectorization, the Wetmore '713 Patent generates a modified new program (vectorized new program) utilizing at least the new program (the new non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
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;	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;	The modified old program (vectorized old program) and modified new program (vectorized new program) have at least the following characteristics: Substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old program (the old non-vectorized program) that is different than corresponding entry in said new program (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old program and new program (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new programs. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.) **Invariant references:** Invariant references are discussed in the '552 Patent and an example is given of providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it

RedBend0008072

Claim Element	Claim Element	The Wetmore '713 Patent
		has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows ...." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old program and vectorized new program.
(c) generating said compact difference result utilizing at least said modified new program and modified old program.	(c) generating said compact difference result utilizing at least said modified new program and modified old program.	The Wetmore '713 Patent generates the compact difference result (the vector patch resource) utilizing at least the modified new program (vectorized new program) and modified old program (vectorized old program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)

Claim Element	Claim Element	The Wetmore '713 Patent
42. A method for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain	55. A system for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an **old data table** (old non-vectorized program) and a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) includes **reference entries** ("jump" entry points) that contain **references** that refer to other **entries** in the data table. **Data table**: The '552 Patent defines a "data

RedBend0008073

Claim Element	Claim Element	The Wetmore '713 Patent
reference that refer to other entries in the data table; the method comprising the steps of:	reference that refer to other entries in the data table; the system comprising a processing device capable of:	table" as "a table of entries, each may have a different size." ('552 Patent col.2 ll.33-34.) The '552 Patent also explains that "a data table can be an executable program either as a loaded program in machine-memory or as an executable-file. In this example, entries are individual machine instructions of the program or the individual data elements used by the program." ('552 Patent col.2 ll.61-65.) The Patentee also provided its claim construction for "data table" as "a table of entries, where an entry is an addressable unit within the data table. An executable program is one example of a data table. *See, e.g.*, '552 Patent 2:33-36; 2:61-63." (Edwards Decl. Exh. A.) Thus, the ROM based code of the Wetmore '713 Patent qualifies as a data table. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) The Patentee also provided its claim construction for "reference entry" as "An addressable unit containing data that includes a reference. *See, e.g.*, '552 Patent 2:35-36; 2:46-47." (Edwards Decl. Exh. A.) In Wetmore, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) The Patentee also provided the identical claim construction for "reference" as "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. *See, e.g.*, '552 Patent 2:42-45." (Edwards Decl. Exh. A.) Under this definition, in the Wetmore '713 Patent, all of the jump entry points in the old non-vectorized program would be all of the references.

RedBend0008074

Claim Element	Claim Element	The Wetmore '713 Patent
		Entry: The '552 Patent states in the Glossary that "a data table includes entries, each of which is an addressable unit that contains data." ('552 Patent col.2 ll.35-36.)
(a) generating a modified old data table utilizing at least said old data table;	(a) generating a modified old data table utilizing at least said old data table;	During vectorization, the Wetmore '713 Patent generates a modified old data table (vectorized old program) utilizing at least the old data table (the old non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17, 18-31; FIG. 4.)
(b) generating a modified new data table utilizing at least said new data table,	(b) generating a modified new data table utilizing at least said new data table,	During vectorization, the Wetmore '713 Patent generates a modified new data table (vectorized new program) utilizing at least the new data table (the new non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.1 1 ll.2-12; FIG. 4.)
said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	The modified old data table (vectorized old program) and modified new data table (vectorized new program) have at least the following characteristics: Substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old data table (the old non-vectorized program) that is different than corresponding entry in said new data table (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old data table and new data table (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new data tables. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.) **Invariant references:** Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries*

RedBend0008075

Claim Element	Claim Element	The Wetmore '713 Patent
		in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent. col.10 ll.7-15.) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows …." ('552 Patent. col.10 ll.47-50.) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs (data table) having table pointers with offsets are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offset will be "the same" in both the vectorized old program and vectorized new program.
(c) generating said compact difference result utilizing at least said modified new data table and modified old data table.	(c) generating said compact difference result utilizing at least said modified new data table and modified old data table.	The Wetmore '713 Patent generates the compact difference result (the vector patch resource) utilizing at least the modified new data table (vectorized new program) and modified old data table (vectorized old program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)

RedBend0008076

Server-Side Dependent Claims 4, 11, 17, And 24:

Claim Element	The Wetmore '713 Patent
4. The method of claim 1, further comprising the step of: (d) storing said compact difference result on a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the compact difference result (vector patch resource) is stored on a storage medium (system disk).
11. The method of claim 8, further comprising the step of: (d) storing said compact difference result on a storage medium.	
17. The system of claim 14, wherein said processor device is further capable of storing said compact difference result on a storage medium.	
24. The system of claim 21, wherein said processor is further capable of storing said compact difference result on a storage medium.	

Server-Side Dependent Claims 27/1, 27/4, 28/8, And 28/11:

Claim Element	The Wetmore '713 Patent
27 [27/1, 27/4]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 1 to 4.	The Wetmore '713 Patent provides a processing device having associated therewith a storage medium (system disk) which holds compact difference result data (vector patch resource) that was generated by the method of claims 1 and 4 [or 8 and 11].
28 [28/8, 28/11]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 8 to 11.	

Server-Side Dependent Claims 38, 45, 51, and 58:

Claim Element	The Wetmore '713 Patent
38. The method of claim 35, further comprising the step of: (d) storing said compact difference result on a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the compact difference result (vector patch resource) is stored on a storage medium (system disk).
45. The method of claim 42, further comprising the step of: (d) storing said compact difference result on a storage medium.	
51. The system of claim 48, wherein said processor device is further capable of storing said compact difference result on a	

RedBend0008077

Claim Element	The Wetmore '713 Patent
storage medium.	
58. The system of claim 55, wherein said processor is further capable of storing said compact difference result on a storage medium.	

Server-Side Dependent Claims 61/35, 61/38, 62/42, And 62/45:

Claim Element	The Wetmore '713 Patent
61 [61/35, 61/38]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 35 to 38.	The Wetmore '713 Patent provides a processing device having associated therewith a storage medium (system disk) which holds compact difference result data (vector patch resource) that was generated by the method of claims 35 and 38 [or 42-45].
62 [62/42, 62/45]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 42 to 45.	

Accordingly, each and every limitation the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48, and 55) and server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent [Q1] is met by the Wetmore '713 Patent, and therefore, these claims are anticipated by the Wetmore '713 Patent under 35 U.S.C. § 102(b).

C. Explanation Of Cited Prior Art And Its Application With Respect To The Substantial New Questions Of Patentability Based On Obviousness

1. Obviousness Based On The Wetmore '713 Patent Alone [Q2, Q8]

For the reasons set forth below, a substantial new question of patentability of the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48, and 55) and dependent server-side claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 is raised by Wetmore '713 Patent [Q2]. In short, and as will be discussed further below, the Wetmore '713 Patent, taken alone, renders such claims obvious.

A substantial new question of patentability of the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and client-side dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 of the '552 Patent under 35 U.S.C. § 103(a) is also raised by Wetmore '713 Patent [Q8]. In short, and as will be discussed further below, the Wetmore '713 Patent, taken alone, renders such claims obvious.

RedBend0008078

With respect to the server-side claims [Q2], to the extent that it is asserted or determined that the Wetmore '713 Patent does not disclose (a) replacing the reference entries for "substantially each reference entry" with distinct label marks as recited in independent claims 1, 5, 14, 18, 35, 39, 48, and 52 or (b) reflecting "substantially each reference in an entry" as invariant references as recited in independent claims 8, 12, 21, 25, 42, 46, 55, and 59, it would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to have done so.

Namely, in the Wetmore '713 Patent, when the entry points (304) are jumps to labels somewhere else in the ROM code, they will be replaced by references to a table (table pointers) in RAM with an offset. However, reference to entry points (303), which themselves point to entry points (304), are not expressly disclosed as being replaced or modified. This, of course, is because there is no need to change reference to entry points (303) because they would not change when updating the ROM program and thus do not need to be accounted for as would normal lines of code with internal references to other lines of code. Rather, any code that may need to be changed is already moved out into RAM and referenced by the entry point (304) replaced by the table pointer and offset. These reference to entry points (303) are therefore akin to invariant references as discussed in the '552 Patent wherein the "net effect is that the invariant reference entries ... will not appear in the difference result." ('552 Patent col.3 ll.40-43.)

Moreover, the Wetmore '713 Patent also discloses "modification of existing vector table entries (replacing a routine), adding new vector table entries (adding new routines to an existing function) or adding a new vector table (adding new routines for new functions). (Wetmore '713 Patent col. 3, lines 9-13; FIGS.7A, 7B). Thus, the Wetmore '713 Patent's handling of vectorization and replacement of references is all done with respect to insert/delete type modifications like the '552 Patent, and is analogous to only replacing substantially all references when there are insert/delete modifications like in the '552 Patent.

With respect to the client-side claims [Q8], because the program patching in the Wetmore '713 Patent is used in connection with executable programs run in both the ROM and RAM, the Wetmore '713 Patent actually saves the final, trivial step of reverting back to the new program from the modified (vectorized) new program, *i.e.*, the final step in the client-side claims of the '552 Patent. However, one of ordinary skill in the art at the time of the alleged invention of the '552 Patent would certainly know how to and could easily be able to revert back to the non-vectorized new program, via substitutions, since the Wetmore '713 Patent teaches how to

RedBend0008079

form the vectorized version via substitutions of internal references with jump references to a table.

The Wetmore '713 Patent recognizes that if one wants to modify code in a ROM (which is "static" and cannot be changed "absent replacing and rewriting the ROM"), one must create and run vectorized code to allow the code to jump outside of the ROM to locations in the RAM. But if the code were to be run from a RAM, which was well-known at the time and indeed performed in the Wetmore '713 Patent, and one applied the *patching* technique of Wetmore '713 Patent (*i.e.*, only sending the compact differences between the modified old and modified new versions of the program), one of ordinary skill in the art would be motivated to return the modified code back to its non-vectorized form so that it would run faster and not include unnecessary jumps that would be needed when code is also running from the ROM. Thus, a person of ordinary skill would have implemented a "predictable variation" of the Wetmore '713 Patent when used in a system that is run primarily from RAM. Doing so would "yield predictable results" in such a RAM system the same as a RAM and ROM systems – not having lines of executable programs being shifted by insertions or deletions to the old program to create a smaller difference result.

Thus, one of ordinarily skill in the art, interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. Thus, it would have been obvious, in view of the "design incentive" of making programs run faster. The Wetmore '713 Patent itself teaches that, as of the time of its filing in 1993, there were two operating systems, the IBM type and the Apple type, wherein the IBM type used a BIOS on a ROM but had the vast majority of the remainder of the operating system software loaded into RAM:

> Various approaches are used to provide the operating system software as part of the computer system. One approach utilized in IBM compatible computer systems is to provide a Basic Input Output System (BIOS) on a Read Only Memory (ROM) device. The BIOS contains the instructions for interaction between the various components of the computer system. The remainder of the Operating system functionality is loaded in Random Access Memory (RAM). In such implementations, the vast majority of the operating system functionality is loaded into the RAM. . . .
>
> An alternative approach is to provide as much operating system functionality into ROM as possible. This has the desired effect of freeing up RAM for application

RedBend0008080

> programs. This approach is used for the operating system for the Apple® Macintosh® family of computers, available from Apple Computer, Inc. of Cupertino, Calif.

(Wetmore '713 Patent col.1 ll.16-34.)

Thus, the Wetmore '713 Patent's patching advantage of sending the compact difference of modified programs to the client in the ROM-based Apple systems, which were more problematic that the RAM-based IBM systems, could easily be tried and used by those of ordinary skill in the art working with the IBM systems who needed to patch code that was executed in the RAM. In view of the Wetmore '713 Patent's patching technique of sending only difference results of modified programs, it would have therefore been obvious to one of ordinary skill in the art to vectorize the RAM code of both the old and new programs to replace internal references with invariant labels or jumps to tables so that only the differences in the called routines could be sent as the patch.

Still further, it would have been obvious to one of ordinary skill in the art at the time to go back from the vectorized new program to the non-vectorized new program since the Wetmore '713 Patent teaches going from the non-vectorized version of the new program to the vectorized version of the new program. The fact that Wetmore does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format, for any reason, such as running the code from RAM as was customary in the IBM-type systems.

Even further, it would have been obvious to one of ordinary skill in the art at the time to go from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare to the non-vectorized old program for debugging, or for version control or the like (a well-known task even recognized by the '552 Patent column 14 lines 5-6). The fact that the Wetmore '713 Patent does not need to go back to the non-vectorized format to run in a combined ROM/RAM system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

In summary, the internal teachings of Wetmore '713 Patent of (1) modifying (or vectorizing) programs, (2) sending compact difference results of modified programs, (3)

RedBend0008081

explaining that RAM-based systems like IBM systems existed, and (4) explaining that ROM-based systems like Apple systems are harder to patch, all would motivate one of ordinary skill in the art at the time of alleged invention of the '552 Patent to go back to the original, non-vectorized format either (a) in the context of RAM based patches or (b) in the context of ROM patching for normal programming-related reasons such as debugging or version control.

Server-Side Claims

Claim charts summarizing the substantial new question of patentability raised by the Wetmore '713 Patent, taken alone, of server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48, and 55) and dependent server-side claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 [Q2] of the '552 Patent under 35 U.S.C. § 103(a) are provided below:

Set 1: Server-side Independent Claims 1, 14, 35, and 48

Claim Element	Claim Element	The Wetmore '713 Patent
1. 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:	14. A system 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 system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old executable program (old non-vectorized program) and a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) includes reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.
(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, whereby a	(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, whereby a	To the extent that it is asserted or determined that the Wetmore '713 Patent does not disclose (a) replacing the reference entries for "substantially each reference entry" with distinct label marks (or (b) reflecting "substantially each reference in an entry" as invariant references), it would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to have done so. In the Wetmore '713 Patent, when the entry points (304) are jumps to labels somewhere else in the ROM code, they will be replaced by references to a

RedBend0008082

Claim Element	Claim Element	The Wetmore '713 Patent
modified old program is generated;	modified old program is generated;	table (table pointers) in RAM with an offset. However, reference to entry points (303), which themselves point to entry points (304), are not expressly disclosed as being replaced or modified because there is no need to change reference to entry points (303) as they would not change when updating the ROM program and thus do not need to be accounted for as would normal lines of code with internal references to other lines of code. Rather, any code that may need to be changed is already moved out into RAM and referenced by the entry point (304) replaced by the table pointer and offset. These reference to entry points (303) are therefore akin to invariant references as discussed in the '552 Patent wherein the "net effect is that the invariant reference entries ... will not appear in the difference result." ('552 Patent col.3 ll.40 43.) Moreover, the Wetmore '713 Patent also discloses "modification of existing vector table entries (replacing a routine), adding new vector table entries (adding new routines to an existing function) or adding a new vector table (adding new routines for new functions). (Wetmore '713 Patent col. 3, lines 9 13; FIGS.7A, 7B). Thus, the Wetmore '713 Patent's handling of vectorization and replacement of references is all done with respect to insert/delete type modifications like the '552 Patent, and is analogous to only replacing substantially all references when there are insert/delete modifications like in the '552 Patent. Thus, in view of the Wetmore '713 Patent, it would have been obvious to scan the old program (the old non-vectorized program) and, for substantially each **reference entry** (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a **distinct label mark** (a table pointer with an offset), whereby a modified old program (vectorized old program) is generated. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump"

RedBend0008083

Claim Element	Claim Element	The Wetmore '713 Patent
		entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) Under this definition, in Wetmore, all of the jump entry points in the old non-vectorized program would be all of the references. The Wetmore '713 Patent replaces each jump entry point (reference) when vectorizing the code. **Distinct label marks**: A "label" is defined in the '552 Patent be "an abstract notation of an entry which is referred by another entry of the same data table through a reference." ('552 Patent col.2 ll.48-50.) "Label marks" are shown as marks (101)-(105) in FIG. 2A. An example of a "distinct label *number*" is described in the '552 Patent: "a distinct label *number* is assigned to each label mark of P''_1. By this particular example, the distinct labels are assigned in ascending order and, as shown, labels marks (101) to (105) are assigned with the respective values 1 to 5." ('552 Patent col.10 ll.63-67 (emphasis added).) Whether the '552 Patent's use of the claim term "distinct label *mark*" is broader than, or the same scope as a "distinct label *number*," the Wetmore '713 Patent still meets this element. Namely, in the Wetmore '713 Patent, each table pointer with its offset (a numerical mark) into the vector table would be a "distinct label mark."
(b) scanning the new program and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new program is	(b) scanning the new program and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new program is	The process described above in (a) would be repeated for the new program (the new non-vectorized program). Namely, during vectorization, it would have been obvious to scan the new program (the new non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a distinct label mark (a table pointer with an offset), whereby a modified new program (vectorized new program) is generated. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)

RedBend0008084

Claim Element	Claim Element	The Wetmore '713 Patent
generated;	generated;	
(c) generating said difference result utilizing directly or indirectly at least said modified old program and modified new program.	(c) generating said difference result utilizing directly or indirectly at least said modified old program and modified new program.	The Wetmore '713 Patent generates the difference result (the vector patch resource) utilizing directly or indirectly at least the modified old program (vectorized old program) and modified new program (vectorized new program). The Wetmore '713 Patent discloses: "In the currently preferred embodiment of the present invention, a tool termed ROMPatch, is provided which automatically creates the vector patch resources. ROMPatch *compares the object files of two versions of the vectorized ROM code to identify routines which are different or new*. In the currently preferred embodiment, routines which are different is accomplished via a Cyclical Redundancy Check (CRC) operation. … In any event, when all the patched routines are found, the vector patch resource is generated." (Wetmore '713, col.10 l.66—col.11 l.12 (emphasis added).)

Claim Element	Claim Element	The Wetmore '713 Patent
35. A method for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	48. A system for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old **data table** (old non-vectorized program) and a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) includes **reference entries** ("jump" entry points) that contain **references** that refer to other **entries** in the data table. **Data table**: The '552 Patent defines a "data table" as "a table of entries, each may have a different size." ('552 Patent col.2 ll.33-34.) The '552 Patent also explains that "a data table can be an executable program either as a loaded program in machine-memory or as an executable-file. In this example, entries are individual machine instructions of the program or the individual data elements used by the program." ('552 Patent col.2 ll.61-65.) The Patentee also provided its claim construction for "data table" as "a table of entries, where an entry is an addressable unit within the data table. An executable program is one example of a data table. *See, e.g.*, '552 Patent 2:33-36; 2:61-63." (Edwards Decl. Exh. A.) Thus, the ROM based executable code of the

RedBend0008085

Claim Element	Claim Element	The Wetmore '713 Patent
		Wetmore '713 Patent qualifies as a data table. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) The Patentee also provided its claim construction for "reference entry" as "An addressable unit containing data that includes a reference. *See, e.g.*, '552 Patent 2:35-36; 2:46-47." (Edwards Decl. Exh. A.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) The Patentee also provided the identical claim construction for "reference" as "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. *See, e.g.*, '552 Patent 2:42-45." (Edwards Decl. Exh. A.) Under this definition, in the Wetmore '713 Patent, all of the jump entry points in the old non-vectorized program would be all of the references. **Entry:** The '552 Patent states in the Glossary that "a data table includes entries, each of which is an addressable unit that contains data." ('552 Patent col.2 ll.35-36.)
(a) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark,	(a) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark,	To the extent that it is asserted or determined that the Wetmore '713 Patent does not disclose (a) replacing the reference entries for "substantially each reference entry" with distinct label marks (or (b) reflecting "substantially each reference in an entry" as invariant references), it would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to have done so. In the Wetmore '713 Patent, when the entry points (304) are jumps to labels somewhere else in the

RedBend0008086

Claim Element	Claim Element	The Wetmore '713 Patent
whereby a modified old data table is generated;	whereby a modified old data table is generated;	ROM code, they will be replaced by references to a table (table pointers) in RAM with an offset. However, reference to entry points (303), which themselves point to entry points (304), are not expressly disclosed as being replaced or modified because there is no need to change reference to entry points (303) as they would not change when updating the ROM program and thus do not need to be accounted for as would normal lines of code with internal references to other lines of code. Rather, any code that may need to be changed is already moved out into RAM and referenced by the entry point (304) replaced by the table pointer and offset. These reference to entry points (303) are therefore akin to invariant references as discussed in the '552 Patent wherein the "net effect is that the invariant reference entries … will not appear in the difference result." ('552 Patent col.3 ll.40 43.) Moreover, the Wetmore '713 Patent also discloses "modification of existing vector table entries (replacing a routine), adding new vector table entries (adding new routines to an existing function) or adding a new vector table (adding new routines for new functions). (Wetmore '713 Patent col. 3, lines 9 13; FIGS.7A, 7B). Thus, the Wetmore '713 Patent's handling of vectorization and replacement of references is all done with respect to insert/delete type modifications like the '552 Patent, and is analogous to only replacing substantially all references when there are insert/delete modifications like in the '552 Patent. Thus, in view of the Wetmore '713 Patent, it would have been obvious to scan the old data table (the old non-vectorized program) and, for substantially each **reference entry** (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a **distinct label mark** (a table pointer with an offset), whereby a modified old data table (vectorized old program) is generated. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of

RedBend0008087

Claim Element	Claim Element	The Wetmore '713 Patent
		"reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) Under this definition, in Wetmore, all of the jump entry points in the old non-vectorized program would be all of the references. The Wetmore '713 Patent replaces each jump entry point (reference) when vectorizing the code. **Distinct label marks**: A "label" is defined in the '552 Patent be "an abstract notation of an entry which is referred by another entry of the same data table through a reference." ('552 Patent col.2 ll.48-50.) "Label marks" are shown as marks (101)-(105) in FIG. 2A. An example of a "distinct label *number*" is described in the '552 Patent: "a distinct label *number* is assigned to each label mark of P''_1. By this particular example, the distinct labels are assigned in ascending order and, as shown, labels marks (101) to (105) are assigned with the respective values 1 to 5." ('552 Patent col.10 ll.63-67 (emphasis added).) Whether the '552 Patent's use of the claim term "distinct label *mark*" is broader than, or the same scope as a "distinct label *number*," the Wetmore '713 Patent still meets this element. Namely, in the Wetmore '713 Patent, each table pointer with its offset (a numerical mark) into the vector table would be a "distinct label mark."
(b) scanning the new data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new data	(b) scanning the new data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new data	The process described above in (a) would be repeated for the new program (the new non-vectorized program). Namely, during vectorization, it would have been obvious to scan the new table (the new non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a distinct label mark (a table pointer with an offset), whereby a modified new table (vectorized new program) is generated. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)

RedBend0008088

Claim Element	Claim Element	The Wetmore '713 Patent
table is generated;	table is generated;	
(c) generating said difference result utilizing directly or indirectly at least said modified old data table and modified new data table.	(c) generating said difference result utilizing directly or indirectly at least said modified old data table and modified new data table.	The Wetmore '713 Patent generates the difference result (the vector patch resource) utilizing directly or indirectly at least the modified old data table (vectorized old program) and modified new data table (vectorized new program). The Wetmore '713 Patent discloses: "In the currently preferred embodiment of the present invention, a tool termed ROMPatch, is provided which automatically creates the vector patch resources. ROMPatch *compares the object files of two versions of the vectorized ROM code to identify routines which are different or new.* In the currently preferred embodiment, routines which are different is accomplished via a Cyclical Redundancy Check (CRC) operation. ... In any event, when all the patched routines are found, the vector patch resource is generated." (Wetmore '713 Patent col.10 l.66—col.11 l.12 (emphasis added).)

Set 2: Server-Side Independent Claims 8, 21, 42, And 55:

Claim Element	Claim Element	The Wetmore '713 Patent
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:	21. A system 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 system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old executable program (old non-vectorized program) and a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) includes reference entries ("jump" entry points) that contain references that refer to other entries in the program.
(a) generating a modified old program utilizing at	(a) generating a modified old program utilizing	During vectorization, the Wetmore '713 Patent generates a modified old program (vectorized old program) utilizing at least the old program (the old

RedBend0008089

Claim Element	Claim Element	The Wetmore '713 Patent
least said old program;	at least said old program;	non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17, 18-31; FIG. 4.)
(b) generating a modified new program utilizing at least said new program,	(b) generating a modified new program utilizing at least said new program,	During vectorization, the Wetmore '713 Patent generates a modified new program (vectorized new program) utilizing at least the new program (the new non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
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;	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;	The modified old program (vectorized old program) and modified new program (vectorized new program) have at least the following characteristics: To the extent that it is asserted or determined that the Wetmore '713 Patent does not disclose reflecting "substantially each reference in an entry" as invariant references, it would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to have done so. In the Wetmore '713 Patent, when the entry points (304) are jumps to labels somewhere else in the ROM code, they will be replaced by references to a table (table pointers) in RAM with an offset. However, reference to entry points (303), which themselves point to entry points (304), are not expressly disclosed as being replaced or modified because there is no need to change reference to entry points (303) as they would not change when updating the ROM program and thus do not need to be accounted for as would normal lines of code with internal references to other lines of code. Rather, any code that may need to be changed is already moved out into RAM and referenced by the entry point (304) replaced by the table pointer and offset. These reference to entry points (303) are therefore akin to invariant references as discussed in the '552 Patent wherein the "net effect is that the invariant reference entries ... will not appear in the difference result." ('552 Patent col.3 ll.40 43.) Moreover, the Wetmore '713 Patent also discloses "modification of existing vector table entries (replacing a routine), adding new vector table entries (adding new routines to an existing function) or adding a new vector table (adding new routines for new functions). (Wetmore '713 Patent col. 3, lines 9 13; FIGS.7A, 7B). Thus, the Wetmore '713 Patent's

RedBend0008090

Claim Element	Claim Element	The Wetmore '713 Patent
		handling of vectorization and replacement of references is all done with respect to insert/delete type modifications like the '552 Patent, and is analogous to only replacing substantially all references when there are insert/delete modifications like in the '552 Patent. Thus, in view of the Wetmore '713 Patent, it would have been obvious for substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old program (the old non-vectorized program) that is different than corresponding entry in said new program (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old program and new program (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), to be reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new programs. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.) **Invariant references**: Invariant references are discussed in the '552 Patent and an example is given of providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby

RedBend0008091

Claim Element	Claim Element	The Wetmore '713 Patent
		keeping the latter relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows ...." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old program and vectorized new program.
(c) generating said compact difference result utilizing at least said modified new program and modified old program.	(c) generating said compact difference result utilizing at least said modified new program and modified old program.	The Wetmore '713 Patent generates the compact difference result (the vector patch resource) utilizing at least the modified new program (vectorized new program) and modified old program (vectorized old program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)

Claim Element	Claim Element	The Wetmore '713 Patent
42. A method for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	55. A system for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old **data table** (old non-vectorized program) and a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) includes **reference entries** ("jump" entry points) that contain **references** that refer to other **entries** in the data table. **Data table**: The '552 Patent defines a "data table" as "a table of entries, each may have a different size." ('552 Patent col.2 ll.33-34.) The '552 Patent also explains that "a data table can be an executable program either as a loaded program in machine-memory or as an executable-file. In this example, entries are individual machine instructions of

RedBend0008092

Claim Element	Claim Element	The Wetmore '713 Patent
	capable of:	the program or the individual data elements used by the program." ('552 Patent col.2 ll.61-65.) The Patentee also provided its claim construction for "data table" as "a table of entries, where an entry is an addressable unit within the data table. An executable program is one example of a data table. *See, e.g.*, '552 Patent 2:33-36; 2:61-63." (Edwards Decl. Exh. A.) Thus, the ROM based code of the Wetmore '713 Patent qualifies as a data table. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) The Patentee also provided its claim construction for "reference entry" as "An addressable unit containing data that includes a reference. *See, e.g.*, '552 Patent 2:35-36; 2:46-47." (Edwards Decl. Exh. A.) In Wetmore, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) The Patentee also provided the identical claim construction for "reference" as "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. *See, e.g.*, '552 Patent 2:42-45." (Edwards Decl. Exh. A.) Under this definition, in the Wetmore '713 Patent, all of the jump entry points in the old non-vectorized program would be all of the references. **Entry:** The '552 Patent states in the Glossary that "a data table includes entries, each of which is an addressable unit that contains data." ('552 Patent col.2 ll.35-36.)
(a) generating a modified old data	(a) generating a modified old data	During vectorization, the Wetmore '713 Patent generates a modified old data table (vectorized old

RedBend0008093

Claim Element	Claim Element	The Wetmore '713 Patent
table utilizing at least said old data table;	table utilizing at least said old data table;	program) utilizing at least the old data table (the old non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17, 18-31; FIG. 4.)
(b) generating a modified new data table utilizing at least said new data table,	(b) generating a modified new data table utilizing at least said new data table,	During vectorization, the Wetmore '713 Patent generates a modified new data table (vectorized new program) utilizing at least the new data table (the new non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	The modified old data table (vectorized old program) and modified new data table (vectorized new program) have at least the following characteristics: To the extent that it is asserted or determined that the Wetmore '713 Patent does not disclose reflecting "substantially each reference in an entry" as invariant references, it would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to have done so. In the Wetmore '713 Patent, when the entry points (304) are jumps to labels somewhere else in the ROM code, they will be replaced by references to a table (table pointers) in RAM with an offset. However, reference to entry points (303), which themselves point to entry points (304), are not expressly disclosed as being replaced or modified because there is no need to change reference to entry points (303) as they would not change when updating the ROM program and thus do not need to be accounted for as would normal lines of code with internal references to other lines of code. Rather, any code that may need to be changed is already moved out into RAM and referenced by the entry point (304) replaced by the table pointer and offset. These reference to entry points (303) are therefore akin to invariant references as discussed in the '552 Patent wherein the "net effect is that the invariant reference entries ... will not appear in the difference result." ('552 Patent col.3 ll.40 43.) Moreover, the Wetmore '713 Patent also discloses "modification of existing vector table entries (replacing a routine), adding new vector table entries (adding new routines to an existing function) or adding a new vector table (adding new routines for

RedBend0008094

Claim Element	Claim Element	The Wetmore '713 Patent
		new functions). (Wetmore '713 Patent col. 3, lines 9 13; FIGS.7A, 7B). Thus, the Wetmore '713 Patent's handling of vectorization and replacement of references is all done with respect to insert/delete type modifications like the '552 Patent, and is analogous to only replacing substantially all references when there are insert/delete modifications like in the '552 Patent. Thus, in view of the Wetmore '713 Patent, it would have been obvious for substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old program (the old non-vectorized program) that is different than corresponding entry in said new data table (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old data table and new data table (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), to be reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new data tables. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.) **Invariant references**: Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be

RedBend0008095

Claim Element	Claim Element	The Wetmore '713 Patent
		modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent. col.10 ll.7-15.) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows ...." ('552 Patent. col.10 ll.47-50.) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs (data table) having table pointers with offsets are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offset will be "the same" in both the vectorized old program and vectorized new program.
(c) generating said compact difference result utilizing at least said modified new data table and modified old data table.	(c) generating said compact difference result utilizing at least said modified new data table and modified old data table.	The Wetmore '713 Patent generates the compact difference result (the vector patch resource) utilizing at least the modified new data table (vectorized new program) and modified old data table (vectorized old program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)

Server-Side Dependent Claims 4, 11, 17, And 24:

Claim Element	The Wetmore '713 Patent
4. The method of claim 1, further comprising the step of: (d) storing said compact difference result on a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the compact difference result (vector patch resource) is stored on a storage medium (system disk).
11. The method of claim 8, further comprising the step of: (d) storing said compact difference result on a storage medium.	
17. The system of claim 14, wherein said processor device is further capable of storing said compact difference result on a storage medium.	
24. The system of claim 21, wherein said processor is further capable of storing said compact difference result on a storage medium.	

RedBend0008096

Server-Side Dependent Claims 27/1, 27/4, 28/8, And 28/11:

Claim Element	The Wetmore '713 Patent
27 [27/1, 27/4]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 1 to 4.	The Wetmore '713 Patent provides a processing device having associated therewith a storage medium (system disk) which holds compact difference result data (vector patch resource) that was generated by the method of claims 1 and 4 [or 8 and 11].
28 [28/8, 28/11]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 8 to 11.	

Server-Side Dependent Claims 38, 45, 51, and 58:

Claim Element	The Wetmore '713 Patent
38. The method of claim 35, further comprising the step of: (d) storing said compact difference result on a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the compact difference result (vector patch resource) is stored on a storage medium (system disk).
45. The method of claim 42, further comprising the step of: (d) storing said compact difference result on a storage medium.	
51. The system of claim 48, wherein said processor device is further capable of storing said compact difference result on a storage medium.	
58. The system of claim 55, wherein said processor is further capable of storing said compact difference result on a storage medium.	

Server-Side Dependent Claims 61/35, 61/38, 62/42, And 62/45:

Claim Element	The Wetmore '713 Patent
61 [61/35, 61/38]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 35 to 38.	The Wetmore '713 Patent provides a processing device having associated therewith a storage medium (system disk) which holds compact difference result data (vector patch resource) that was generated by the method of claims 35 and 38 [or 42-45].
62 [62/42, 62/45]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 42 to 45.	

RedBend0008097

Accordingly, each and every limitation of the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48, and 55) and server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent [Q2] is met or rendered obvious by the Wetmore '713 Patent, and therefore, these claims are rendered obvious by the Wetmore '713 Patent under 35 U.S.C. § 103(a).

Client-Side Claims

Claim charts summarizing the substantial new question of patentability raised by the Wetmore '713 Patent, taken alone, of the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and client-side dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 [Q8] of the '552 Patent under 35 U.S.C. § 103(a) are provided below:

Set 3: Client-Side Independent Claims 5, 18, 39, And 52:

Claim Element	Claim Element	The Wetmore '713 Patent
5. A method for performing an update in an old executable program so as to generate 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:	18. A system for performing an update in an old executable program so as to generate a new executable program; each program including reference entries that contain reference that refer to other entries in the program; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old executable program (old non-vectorized program) so as to generate a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old program (vectorized old program) and a modified new program (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) scanning the old program and for substantially each reference entry	(b) scanning the old program and for substantially each reference entry	During vectorization, the Wetmore '713 Patent scans the old program (the old non-vectorized program) and, for substantially each reference entry (each "jump" entry point),

RedBend0008098

Claim Element	Claim Element	The Wetmore '713 Patent
perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old program is generated;	perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old program is generated;	replaces the reference (the location in the ROM code in the jump entry point) of the entry by a distinct label mark (a table pointer with an offset), whereby a modified old program (vectorized old program) is generated. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(c) reconstituting the modified new program utilizing at least said compact difference result and said modified old program;	(c) reconstituting the modified new program utilizing at least said compact difference result and said modified old program;	The Wetmore '713 Patent reconstitutes a modified new program (vectorized new program) utilizing directly or indirectly at least the modified old program (vectorized old program) and the compact difference result (vector patch resource). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
said modified new program is differed from said new program at least in that substantially each reference entry in said new program is replaced in said modified new program by a distinct label mark;	said modified new program is differed from said new program at least in that substantially each reference entry in said new program is replaced in said modified new program by a distinct label mark;	In the Wetmore '713 Patent, the modified new program (vectorized new program) is differed from the new program (new non-vectorized program) at least in that substantially each reference entry (each "jump" entry point) in the new program is replaced in the modified new program by a distinct label mark (a table pointer with an offset). (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new programs, to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the

RedBend0008099

Claim Element	Claim Element	The Wetmore '713 Patent
		teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that the Wetmore '713 Patent does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

Claim Element	Claim Element	The Wetmore '713 Patent
39. A method for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	52. A system for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old data table (old non-vectorized program) so as to generate a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the data table.

RedBend0008100

Claim Element	Claim Element	The Wetmore '713 Patent
	capable of:	
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old data table (vectorized old program) and a modified new data table (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old data table is generated;	(b) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old data table is generated;	During vectorization, the Wetmore '713 Patent scans the old data table (the old non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a distinct label mark (a table pointer with an offset), whereby a modified old data table (vectorized old program) is generated. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(c) reconstituting the modified new data table utilizing at least said compact difference result and said modified old data table;	(c) reconstituting the modified new data table utilizing at least said compact difference result and said modified old data table;	The Wetmore '713 Patent reconstitutes a modified new data table (vectorized new program) utilizing directly or indirectly at least the modified old data table (vectorized old program) and the compact difference result (vector patch resource). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
said modified new data table is differed from said new data table at least in that substantially each reference entry in said new data table is replaced in said modified new data table by a distinct label mark;	said modified new data table is differed from said new data table at least in that substantially each reference entry in said new data table is replaced in said modified new data table by a distinct label mark;	In the Wetmore '713 Patent, the modified new data table (vectorized new program) is differed from the new data table (new non-vectorized program) at least in that substantially each reference entry (each "jump" entry point) in the new data table is replaced in the modified new data table by a distinct label mark (a table pointer with an offset). (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said	(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new programs, to reconstitute

RedBend0008101

Claim Element	Claim Element	The Wetmore '713 Patent
modified new data table.	modified new data table.	the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

RedBend0008102

Set 4: Client-Side Independent claims 12, 25, 46, And 59:

Claim Element	Claim Element	The Wetmore '713 Patent
12. A method for performing an update in an old executable program so as to generate 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:	25. A system for performing an update in an old executable program so as to generate a new executable program; each program including reference entries that contain reference that refer to other entries in the program; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old executable program (old non-vectorized program) so as to generate a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old program (vectorized old program) and a modified new program (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) generating a modified old program utilizing at least said old program;	(b) generating a modified old program utilizing at least said old program;	During vectorization, the Wetmore '713 Patent generates a modified old program (vectorized old program) utilizing at least the old program (the old non-vectorized program). (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(c) reconstituting a modified new program utilizing directly or indirectly at least said modified old program and said compact difference result;	(c) reconstituting a modified new program utilizing directly or indirectly at least said modified old program and said compact difference result;	The Wetmore '713 Patent reconstitutes a modified new program (vectorized new program) utilizing directly or indirectly at least the modified old program (vectorized old program) and the compact difference result (vector patch resource). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
said modified old program and modified new program have at least the following characteristics:	said modified old program and modified new programs have at least the following characteristics:	In the Wetmore '713 Patent, the modified old program (vectorized old program) and modified new program (vectorized new program) have at least the following characteristics:

RedBend0008103

Claim Element	Claim Element	The Wetmore '713 Patent
(i) substantially each reference in an entry in said old program that is different than corresponding entry in said new program due to delete/inset 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;	(i) substantially each reference in an entry in said old program that is different than corresponding entry in said new program due to delete/inset 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;	Substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old program (the old non-vectorized program) that is different than corresponding entry in said new program (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old program and new program (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new programs. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.) **Invariant references:** Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter

RedBend0008104

Claim Element	Claim Element	The Wetmore '713 Patent
		relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows ...." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old program and vectorized new program.
(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new executable programs (*i.e.*, data tables), to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the

RedBend0008105

Claim Element	Claim Element	The Wetmore '713 Patent
		final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that the Wetmore '713 Patent does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

RedBend0008106

Claim Element	Claim Element	The Wetmore '713 Patent
46. A method for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	59. A system for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old data table (old non-vectorized program) so as to generate a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the data table.
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old data table (vectorized old program) and a modified new data table (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) generating a modified old data table utilizing at least said old data table;	(b) generating a modified old data table utilizing at least said old data table;	During vectorization, the Wetmore '713 Patent generates a modified old data table (vectorized old program) utilizing at least the old data table (the old non-vectorized program). (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(c) reconstituting a modified new data table utilizing directly or indirectly at least said modified old data table and said compact difference result;	(c) reconstituting a modified new data table utilizing directly or indirectly at least said modified old data table and said compact difference result;	The Wetmore '713 Patent reconstitutes a modified new data table (vectorized new program) utilizing directly or indirectly at least the modified old data table (vectorized old program) and the compact difference result (vector patch resource). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table	said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than	In the Wetmore '713 Patent, the modified old data table (vectorized old program) and modified new data table (vectorized new program) have at least the following characteristics: Substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old data table (the old non-vectorized program) that is

RedBend0008107

Claim Element	Claim Element	The Wetmore '713 Patent
that is different than corresponding entry in said new data table due to delete/inset modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	corresponding entry in said new data table due to delete/inset modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	different than corresponding entry in said new data table (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old data table and new data table (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new data table. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.) **Invariant references:** Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs or data tables wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new

RedBend0008108

Claim Element	Claim Element	The Wetmore '713 Patent
		programs wherein address references in entries are replaced by label marks as follows …." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old data table and vectorized new data table.
(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new executable programs (*i.e.*, data tables), to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the

RedBend0008109

Claim Element	Claim Element	The Wetmore '713 Patent
		new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that the Wetmore '713 Patent does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that the Wetmore '713 Patent does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

Client-Side Dependent Claims 13, 26, 31, 34, 47, 60, 65, And 68:

Claim Element	The Wetmore '713 Patent
13. The method of claim 5, wherein said data is received in step (a) from a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the data of the compact difference result (vector patch resource) is received from a storage
26. The system of claim 18, wherein said data is received in step (a) from a storage medium.	
31. The method of claim 12, wherein said	

RedBend0008110

Claim Element	The Wetmore '713 Patent
data is received in step (a) from a storage medium.	medium (system disk).
34. The system of claim 25, wherein said data is received in step (a) from a storage medium.	See claims 5, 18, 12, 25, 39, 59, 46, and 59 above.
47. The method of claim 39, wherein said data is received in step (a) from a storage medium.	
60. The system of claim 59, wherein said data is received in step (a) from a storage medium.	
65. The method of claim 46, wherein said data is received in step (a) from a storage medium.	
68. The system of claim 59, wherein said data is received in step (a) from a storage medium.[3]	

Accordingly, the Wetmore '713 Patent, taken in view of its own teaching, meets each and every limitation of the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and client-side dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 of the '552 Patent [Q8], and therefore renders each of these claims invalid under 35 U.S.C. § 103.

2. Obviousness Based On The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin [Q3, Q9]

For the reasons set forth below, a substantial new question of patentability under 35 U.S.C. § 103(a) of (a) the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48 and 55) and server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 [Q3] and (b) the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and client-side dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 [Q9], is raised by the Wetmore '713 Patent in view of the 1990 IBM Technical Disclosure Bulletin.

In short, to the extent that it is asserted or determined that the Wetmore '713 Patent does not disclose (a) replacing the reference entries for "substantially each reference entry" with distinct label marks as recited in independent claims 1, 5, 14, 18, 35, 39, 48, and 52 or (b) reflecting "substantially each reference in an entry" as invariant references as recited in independent claims 8, 12, 21, 25, 42, 46, 55, and 59, it would have been obvious to one of

[3] It is noted that claims 60 and 68 are identical.

RedBend0008111

ordinary skill in the art at the time of the alleged invention of the '552 Patent to have done so in the Wetmore '713 Patent or in view of the 1990 IBM Technical Disclosure Bulletin.

Namely, in the Wetmore '713 Patent, when the entry points (304) are jumps to labels somewhere else in the ROM code, they will be replaced by references to a table (table pointers) in RAM with an offset. However, reference to entry points (303), which themselves point to entry points (304), are not expressly disclosed as being replaced or modified. This, of course, is because there is no need to change reference to entry points (303) because they would not change when updating the ROM program and thus do not need to be accounted for as would normal lines of code with internal references to other lines of code. Rather, any code that may need to be changed is already moved out into RAM and referenced by the entry point (304) replaced by the table pointer and offset.

However, to the extent one takes the view that the reference entries may also include the reference to entry points (303) in addition to the entry points (304), the 1990 IBM Technical Disclosure Bulletin discloses that instead of having "references to entry points" that refer to entry points (which are then replaced), the internal jumps or references in the code in the ROS are replaced with indirect jumps or calls to the table in EEPROM. Thus, instead of having a jump to a jump as may be the case in the Wetmore '713 Patent, in the same context of patching of ROM or ROS, it was known to simply use jumps or calls directly from the ROM code. Thus, it would have been a mere matter of design choice for one of ordinary skill in the art to either choose references to entry points (since they do not change when the code is modified) or use only jumps or calls to the table. Likewise, one of ordinary skill in the art could have also replaced any double jumps in the Wetmore '713 Patent with a single jump, as disclosed in the 1990 IBM Technical Disclosure Bulletin, as a matter of design choice to reduce the number of jumps and to make the program run faster, a well-known, predictable design incentive.

Claim charts summarizing the substantial new question of patentability raised by the Wetmore '713 Patent in view of the 1990 IBM Technical Disclosure Bulletin of the server-side independent claims (claims 1, 8, 14, 21, 35, 42, 48, and 55) and server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent under 35 U.S.C. § 103(a) are provided below:

RedBend0008112

Set 1: Server-Side Independent Claims 1, 14, 35, And 48:

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
1. 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:	14. A system 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 system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old executable program (old non-vectorized program) and a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) includes reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.
(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, whereby a modified old program is generated;	(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, whereby a modified old program is generated;	It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the 1990 IBM Technical Disclosure Bulletin to replace substantially each reference with jumps to the RAM table, such that the Wetmore '713 Patent would scan the old program (the old non-vectorized program) and, for substantially each **reference entry** (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a **distinct label mark** (a table pointer with an offset), whereby a modified old program (vectorized old program) is generated. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '714 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "a part of the data appearing in an

RedBend0008113

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		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." ('552 Patent col.2 ll.42-45.) Under this definition, in the Wetmore '713 Patent, all of the jump entry points in the old non-vectorized program would be all of the references. The Wetmore '713 Patent replaces each jump entry point (reference) when vectorizing the code. **Distinct label marks**: A "label" is defined in the '552 Patent is be "an abstract notation of an entry which is referred by another entry of the same data table through a reference." ('552 Patent col.2 ll.48-50.) "Label marks" are shown as marks (101)-(105) in FIG. 2A. An example of a "distinct label *number*" is described in the '552 Patent: "a distinct label *number* is assigned to each label mark of P''_1. By this particular example, the distinct labels are assigned in ascending order and, as shown, labels marks (101) to (105) are assigned with the respective values 1 to 5." ('552 Patent col.10 ll.63-67 (emphasis added).) Whether the '552 Patent's use of the claim term "distinct label *mark*" is broader than, or the same scope as a "distinct label *number*," the Wetmore '713 Patent still meets this element. Namely, in the Wetmore '713 Patent, each table pointer with its offset (a numerical mark) into the vector table would be a "distinct label mark."
(b) scanning the new program and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new program is generated;	(b) scanning the new program and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new program is generated;	The process described above in (a) is repeated to replace substantially each reference with jumps to the RAM table, for the new program (the new non-vectorized program). Namely, during vectorization, the Wetmore '713 Patent, as modified by the teaching of the 1990 IBM Technical Disclosure Bulletin, scans the new program (the new non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of said entry by a distinct label mark (a table pointer with an offset), whereby a modified new program (vectorized new program) is generated.
(c) generating said difference result utilizing directly	(c) generating said difference result utilizing directly or	The Wetmore '713 Patent generates the difference result (the vector patch resource) utilizing directly or indirectly at least the modified old program

RedBend0008114

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
or indirectly at least said modified old program and modified new program.	indirectly at least said modified old program and modified new program.	(vectorized old program) and modified new program (vectorized new program). The Wetmore '713 Patent discloses: "In the currently preferred embodiment of the present invention, a tool termed ROMPatch, is provided which automatically creates the vector patch resources. ROMPatch *compares the object files of two versions of the vectorized ROM code to identify routines which are different or new.* In the currently preferred embodiment, routines which are different is accomplished via a Cyclical Redundancy Check (CRC) operation. … In any event, when all the patched routines are found, the vector patch resource is generated." (Wetmore '713 Patent col.10 l.66—col.11 l.12 (emphasis added).)

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
35. A method for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	48. A system for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old **data table** (old non-vectorized program) and a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) includes **reference entries** ("jump" entry points) that contain **references** that refer to other **entries** in the data table. **Data table**: The '552 Patent defines a "data table" as "a table of entries, each may have a different size." ('552 Patent col.2 ll.33-34.) The '552 Patent also explains that "a data table can be an executable program either as a loaded program in machine-memory or as an executable-file. In this example, entries are individual machine instructions of the program or the individual data elements used by the program." ('552 Patent col.2 ll.61-65.) The Patentee also provided its claim construction for "data table" as "a table of entries, where an entry is an addressable unit within the data table. An executable program is one example of a data table. *See, e.g.*, '552 Patent 2:33-36; 2:61-63." (Edwards Decl. Exh. A.) Thus, the ROM based code of the Wetmore '713 Patent qualifies as a data table.

RedBend0008115

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		Reference entry: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) The Patentee also provided its claim construction for "reference entry" as "An addressable unit containing data that includes a reference. *See, e.g.*, '552 Patent 2:35-36; 2:46-47." (Edwards Decl. Exh. A.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) The Patentee also provided the identical claim construction for "reference" as "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. *See, e.g.*, '552 Patent 2:42-45." (Edwards Decl. Exh. A.) Under this definition, in the Wetmore '713 Patent, all of the jump entry points in the old non-vectorized program would be all of the references. **Entry:** The '552 Patent states in the Glossary that "a data table includes entries, each of which is an addressable unit that contains data." ('552 Patent col.2 ll.35-36.)
(a) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark,	(a) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark,	It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the 1990 IBM Technical Disclosure Bulletin to replace substantially each reference with jumps to the RAM table, such that the Wetmore '713 Patent would scan the old data table (the old non-vectorized program) and for substantially each **reference entry** (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a **distinct label mark** (a table pointer

RedBend0008116

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
whereby a modified old data table is generated;	whereby a modified old data table is generated;	with an offset), whereby a modified old data table (vectorized old program) is generated. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) Under this definition, in Wetmore, all of the jump entry points in the old non-vectorized program would be all of the references. The Wetmore '713 Patent replaces each jump entry point (reference) when vectorizing the code. **Distinct label marks**: A "label" is defined in the '552 Patent is be "an abstract notation of an entry which is referred by another entry of the same data table through a reference." ('552 Patent col.2 ll.48-50.) "Label marks" are shown as marks (101)-(105) in FIG. 2A. An example of a "distinct label *number*" is described in the '552 Patent: "a distinct label *number* is assigned to each label mark of $P"_1$. By this particular example, the distinct labels are assigned in ascending order and, as shown, labels marks (101) to (105) are assigned with the respective values 1 to 5." ('552 Patent col.10 ll.63-67 (emphasis added).) Whether the '552 Patent's use of the claim term "distinct label *mark*" is broader than, or the same scope as a "distinct label *number*," the Wetmore '713 Patent still meets this element. Namely, in the Wetmore '713 Patent, each table pointer with its offset (a numerical mark) into the vector table would be a "distinct label mark."
(b) scanning the	(b) scanning the	The process described above for (a) is repeated

RedBend0008117

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
new data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new data table is generated;	new data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new data table is generated;	to replace substantially each reference with jumps to the RAM table, for the new data table (the new non-vectorized program). Namely, during vectorization, the Wetmore '713 Patent, as modified by the teaching of 1990 IBM Technical Disclosure Bulletin, scans the new data table (the new non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of said entry by a distinct label mark (a table pointer with an offset), whereby a modified new data table (vectorized new program) is generated.
(c) generating said difference result utilizing directly or indirectly at least said modified old data table and modified new data table.	(c) generating said difference result utilizing directly or indirectly at least said modified old data table and modified new data table.	The Wetmore '713 Patent generates the difference result (the vector patch resource) utilizing directly or indirectly at least the modified old data table (vectorized old program) and modified new data table (vectorized new program). The Wetmore '713 Patent discloses: "In the currently preferred embodiment of the present invention, a tool termed ROMPatch, is provided which automatically creates the vector patch resources. ROMPatch *compares the object files of two versions of the vectorized ROM code to identify routines which are different or new.* In the currently preferred embodiment, routines which are different is accomplished via a Cyclical Redundancy Check (CRC) operation. … In any event, when all the patched routines are found, the vector patch resource is generated." (Wetmore '713, col.10 l.66—col.11 l.12 (emphasis added).)

Set 2: Server-Side Independent Claims 8, 21, 42, And 55:

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
8. A method for generating a compact difference result between an old executable program and a new executable program; each program including reference entries	21. A system for generating a compact difference result between an old executable program and a new executable program; each program including	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old executable program (old non-vectorized program) and a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain reference that refer to other entries in the program.

RedBend0008118

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
that contain reference that refer to other entries in the program; the method comprising the steps of:	reference entries that contain reference that refer to other entries in the program; the system comprising a processing device capable of:	
(a) generating a modified old program utilizing at least said old program;	(a) generating a modified old program utilizing at least said old program;	During vectorization, the Wetmore '713 Patent generates a modified old program (vectorized old program) utilizing at least the old program (the old non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17, 18-31; FIG. 4.)
(b) generating a modified new program utilizing at least said new program,	(b) generating a modified new program utilizing at least said new program,	During vectorization, the Wetmore '713 Patent generates a modified new program (vectorized new program) utilizing at least the new program (the new non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
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	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	The modified old program (vectorized old program) and modified new program (vectorized new program) have at least the following characteristics: It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the 1990 IBM Technical Disclosure Bulletin to replace substantially each reference with jumps to the RAM table, such that substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old program (the old non-vectorized program) that is different than corresponding entry in said new program (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old program and new program (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new programs. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.) **Invariant references**: Invariant references are

RedBend0008119

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
modified old and modified new programs;	modified old and modified new programs;	discussed in the '552 Patent and an example is given of providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent, col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows ...." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old program and vectorized new program.
(c) generating said compact difference	(c) generating said compact	The Wetmore '713 Patent generates the compact difference result (the vector patch resource) utilizing at

RedBend0008120

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
result utilizing at least said modified new program and modified old program.	difference result utilizing at least said modified new program and modified old program.	least the modified new program (vectorized new program) and modified old program (vectorized old program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
42. A method for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	55. A system for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old **data table** (old non-vectorized program) and a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) includes **reference entries** ("jump" entry points) that contain **references** that refer to other **entries** in the data table. **Data table**: The '552 Patent defines a "data table" as "a table of entries, each may have a different size." ('552 Patent col.2 ll.33-34.) The '552 Patent also explains that "a data table can be an executable program either as a loaded program in machine-memory or as an executable-file. In this example, entries are individual machine instructions of the program or the individual data elements used by the program." ('552 Patent col.2 ll.61-65.) The Patentee also provided its claim construction for "data table" as "a table of entries, where an entry is an addressable unit within the data table. An executable program is one example of a data table. *See, e.g.*, '552 Patent 2:33-36; 2:61-63." (Edwards Decl. Exh. A.) Thus, the ROM based code of the Wetmore '713 Patent qualifies as a data table. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) The Patentee also provided its claim construction for "reference entry" as "An addressable unit containing data that includes a reference. *See, e.g.*, '552 Patent 2:35-36; 2:46-47." (Edwards Decl. Exh. A.) In Wetmore, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code."

RedBend0008121

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		(Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) The Patentee also provided the identical claim construction for "reference" as "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. *See, e.g.*, '552 Patent 2:42-45." (Edwards Decl. Exh. A.) Under this definition, in the Wetmore '713 Patent, all of the jump entry points in the old non-vectorized program would be all of the references. **Entry:** The '552 Patent states in the Glossary that "a data table includes entries, each of which is an addressable unit that contains data." ('552 Patent col.2 ll.35-36.)
(a) generating a modified old data table utilizing at least said old data table;	(a) generating a modified old data table utilizing at least said old data table;	During vectorization, the Wetmore '713 Patent generates a modified old data table (vectorized old program) utilizing at least the old data table (the old non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17, 18-31; FIG. 4.)
(b) generating a modified new data table utilizing at least said new data table,	(b) generating a modified new data table utilizing at least said new data table,	During vectorization, the Wetmore '713 Patent generates a modified new data table (vectorized new program) utilizing at least the new data table (the new non-vectorized program). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
said modified old data table and modified new data table have at least the following characteristics:	said modified old data table and modified new data table have at least the following characteristics:	The modified old data table (vectorized old program) and modified new data table (vectorized new program) have at least the following characteristics:
(i) substantially each reference in	(i) substantially each reference in	It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching

RedBend0008122

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	of the 1990 IBM Technical Disclosure Bulletin to replace substantially each reference with jumps to the RAM table, such that substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old data table (the old non-vectorized program) that is different than corresponding entry in said new data table (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old data table and new data table (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new data tables. (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.) **Invariant references**: Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent, col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent, col.10 ll.7-15.) Further, the

RedBend0008123

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		'552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows …." ('552 Patent. col.10 ll.47-50.) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs (data table) having table pointers with offsets are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offset will be "the same" in both the vectorized old program and vectorized new program.
(c) generating said compact difference result utilizing at least said modified new data table and modified old data table.	(c) generating said compact difference result utilizing at least said modified new data table and modified old data table.	The Wetmore '713 Patent generates the compact difference result (the vector patch resource) utilizing at least the modified new data table (vectorized new program) and modified old data table (vectorized old program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)

Server-Side Dependent Claims 4, 11, 17, And 24:

Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
4. The method of claim 1, further comprising the step of: (d) storing said compact difference result on a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the compact difference result (vector patch resource) is stored on a storage medium (system disk).
11. The method of claim 8, further comprising the step of: (d) storing said compact difference result on a storage medium.	
17. The system of claim 14, wherein said processor device is further capable of storing said compact difference result on a storage medium.	
24. The system of claim 21, wherein said processor is further capable of storing said compact difference result on a storage	

RedBend0008124

Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
medium.	

Server-Side Dependent Claims 27/1, 27/4, 28/8, and 28/11:

Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
27 [27/1, 27/4]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 1 to 4.	The Wetmore '713 Patent provides a processing device having associated therewith a storage medium (system disk) which holds compact difference result data (vector patch resource) that was generated by the method of claims 1 and 4 [or 8-11].
28 [28/8, 28/11]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 8 to 11.	

Server-Side Dependent Claims 38, 45, 51, And 58:

Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
38. The method of claim 35, further comprising the step of: (d) storing said compact difference result on a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the compact difference result (vector patch resource) is stored on a storage medium (system disk).
45. The method of claim 42, further comprising the step of: (d) storing said compact difference result on a storage medium.	
51. The system of claim 48, wherein said processor device is further capable of storing said compact difference result on a storage medium.	
58. The system of claim 55, wherein said processor is further capable of storing said compact difference result on a storage medium.	

RedBend0008125

Server-Side Dependent Claims 61/35, 61/38, 62/42, And 62/45:

Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
61 [61/35, 61/38]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 35 to 38.	The Wetmore '713 Patent provides a processing device having associated therewith a storage medium (system disk) which holds compact difference result data (vector patch resource) that was generated by the method of claims 35 and 38 [or 42 and 45].
62 [62/42, 62/45]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 42 to 45.	

Set 3: Client-Side Independent Claims 5, 18, 39, And 52:

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
5. A method for performing an update in an old executable program so as to generate 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:	18. A system for performing an update in an old executable program so as to generate a new executable program; each program including reference entries that contain reference that refer to other entries in the program; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old executable program (old non-vectorized program) so as to generate a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old program (vectorized old program) and a modified new program (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) scanning the old program and for substantially each reference entry	(b) scanning the old program and for substantially each reference entry	It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the 1990 IBM Technical Disclosure Bulletin to replace substantially

RedBend0008126

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old program is generated;	perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old program is generated;	each reference with jumps to the RAM table, such that the Wetmore '713 Patent would scan the old program (the old non-vectorized program) and, for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a distinct label mark (a table pointer with an offset), whereby a modified old program (vectorized old program) is generated.
(c) reconstituting the modified new program utilizing at least said compact difference result and said modified old program;	(c) reconstituting the modified new program utilizing at least said compact difference result and said modified old program;	The Wetmore '713 Patent reconstitutes a modified new program (vectorized new program) utilizing directly or indirectly at least the modified old program (vectorized old program) and the compact difference result (vector patch resource).
said modified new program is differed from said new program at least in that substantially each reference entry in said new program is replaced in said modified new program by a distinct label mark;	said modified new program is differed from said new program at least in that substantially each reference entry in said new program is replaced in said modified new program by a distinct label mark;	In the Wetmore '713 Patent, as modified by the teaching of the 1990 IBM Technical Disclosure Bulletin, the modified new program (vectorized new program) is differed from the new program (new non-vectorized program) at least in that substantially each reference entry (each "jump" entry point) in the new program is replaced in the modified new program by a distinct label mark (a table pointer with an offset).
(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new programs, to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary

RedBend0008127

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
39. A method for performing an update in an old data table so as to generate a new data table; each data table including reference entries that	52. A system for performing an update in an old data table so as to generate a new data table; each data table including reference entries that	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old data table (old non-vectorized program) so as to generate a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) including reference entries ("jump" entry points) that

RedBend0008128

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
contain reference that refer to other entries in the data table; the method comprising the steps of:	contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	contain references (the location in the ROM code in the jump entry point) that refer to other entries in the data table.
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old data table (vectorized old program) and a modified new data table (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old data table is generated;	(b) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old data table is generated;	It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the 1990 IBM Technical Disclosure Bulletin to replace substantially each reference with jumps to the RAM table, such that the Wetmore '713 Patent would scan the old data table (the old non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a distinct label mark (a table pointer with an offset), whereby a modified old data table (vectorized old program) is generated.
(c) reconstituting the modified new data table utilizing at least said compact difference result and said modified old data table;	(c) reconstituting the modified new data table utilizing at least said compact difference result and said modified old data table;	The Wetmore '713 Patent reconstitutes a modified new data table (vectorized new program) utilizing directly or indirectly at least the modified old data table (vectorized old program) and the compact difference result (vector patch resource).
said modified new data table is differed from said new data table at least in that substantially each reference entry in said new data table is replaced in said modified new data table by a distinct	said modified new data table is differed from said new data table at least in that substantially each reference entry in said new data table is replaced in said modified new data table by a distinct	In the Wetmore '713 Patent, as modified by the teaching of the 1990 IBM Technical Disclosure Bulletin, the modified new data table (vectorized new program) is differed from the new data table (new non-vectorized program) at least in that substantially each reference entry (each "jump" entry point) in the new data table is replaced in the modified new data table by a distinct label mark (a table pointer with an offset).

RedBend0008129

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
label mark;	label mark;	
(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new programs, to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the

RedBend0008130

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

Set 4: Client-Side Independent Claims 12, 25, 46, And 59:

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
12. A method for performing an update in an old executable program so as to generate 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:	25. A system for performing an update in an old executable program so as to generate a new executable program; each program including reference entries that contain reference that refer to other entries in the program; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old executable program (old non-vectorized program) so as to generate a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old program (vectorized old program) and a modified new program (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) generating a modified old program utilizing at least said old program;	(b) generating a modified old program utilizing at least said old program;	During vectorization, the Wetmore '713 Patent generates a modified old program (vectorized old program) utilizing at least the old program (the old non-vectorized program).
(c) reconstituting a modified new program utilizing directly or indirectly	(c) reconstituting a modified new program utilizing directly or indirectly	The Wetmore '713 Patent reconstitutes a modified new program (vectorized new program) utilizing directly or indirectly at least the modified old program (vectorized old program)

RedBend0008131

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
at least said modified old program and said compact difference result;	at least said modified old program and said compact difference result;	and the compact difference result (vector patch resource).
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/inset 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;	said modified old program and modified new programs 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/inset 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;	In the Wetmore '713 Patent, the modified old program (vectorized old program) and modified new program (vectorized new program) have at least the following characteristics: It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the 1990 IBM Technical Disclosure Bulletin to replace substantially each reference with jumps to the RAM table, such that substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old program (the old non-vectorized program) that is different than corresponding entry in said new program (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old program and new program (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new programs. **Invariant references:** Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known

RedBend0008132

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows ...." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old program and vectorized new program.
(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new executable programs (*i.e.*, data tables), to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore

RedBend0008133

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		'713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

RedBend0008134

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
46. A method for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	59. A system for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old data table (old non-vectorized program) so as to generate a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the data table.
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old data table (vectorized old program) and a modified new data table (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) generating a modified old data table utilizing at least said old data table;	(b) generating a modified old data table utilizing at least said old data table;	During vectorization, the Wetmore '713 Patent generates a modified old data table (vectorized old program) utilizing at least the old data table (the old non-vectorized program).
(c) reconstituting a modified new data table utilizing directly or indirectly at least said modified old data table and said compact difference result;	(c) reconstituting a modified new data table utilizing directly or indirectly at least said modified old data table and said compact difference result;	The Wetmore '713 Patent reconstitutes a modified new data table (vectorized new program) utilizing directly or indirectly at least the modified old data table (vectorized old program) and the compact difference result (vector patch resource).
said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than	said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than corresponding entry	In the Wetmore '713 Patent, the modified old data table (vectorized old program) and modified new data table (vectorized new program) have at least the following characteristics: It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the 1990 IBM Technical Disclosure Bulletin to replace substantially each reference with jumps to the RAM table, such that

RedBend0008135

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
corresponding entry in said new data table due to delete/inset modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	in said new data table due to delete/inset modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old data table (the old non-vectorized program) that is different than corresponding entry in said new data table (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old data table and new data table (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new data table. **Invariant references**: Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs or data tables wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes

RedBend0008136

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows ...." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old data table and vectorized new data table.
(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new executable programs (*i.e.*, data tables), to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of

RedBend0008137

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
		ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

Client-Side Dependent Claims 13, 26, 31, 34, 47, 60, 65, And 68:

Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
13. The method of claim 5, wherein said data is received in step (a) from a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the data of the compact difference result (vector patch resource) is received from a storage medium (system disk). See claims 5, 18, 12, 25, 39, 59, 46, and 59 above.
26. The system of claim 18, wherein said data is received in step (a) from a storage medium.	
31. The method of claim 12, wherein said data is received in step (a) from a storage medium.	
34. The system of claim 25, wherein said data is received in step (a) from a storage medium.	
47. The method of claim 39, wherein said data is received in step (a) from a storage medium.	
60. The system of claim 59, wherein said data is received in step (a) from a storage medium.	

RedBend0008138

Claim Element	The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin
65. The method of claim 46, wherein said data is received in step (a) from a storage medium.	
68. The system of claim 59, wherein said data is received in step (a) from a storage medium.	

Accordingly, each and every limitation of (a) the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48 and 55) and server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 [Q3] and (b) the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and client-side dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 [Q9] is rendered obvious under 35 U.S.C. § 103 by the Wetmore '713 Patent in view of the 1990 IBM Technical Disclosure Bulletin.

3. Obviousness Based On The Wetmore '713 Patent In View Of The Dummermuth '853 Patent [Q4, Q10]

For the reasons set forth below, a substantial new question of patentability under 35 U.S.C. § 103(a) of (a) the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48, and 55) and server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 [Q4] and (b) the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and client-side dependent claims 13, 26, 31, 34, 47, 60, 65, and 68, is raised by the Wetmore '713 Patent in view of the Dummermuth '853 Patent [Q10].

In short, to the extent that it is asserted or determined that the Wetmore '713 Patent does not disclose (a) replacing the reference entries for "substantially each reference entry" with distinct label marks as recited in independent claims 1, 5, 14, 18, 35, 39, 48, and 52 or (b) reflecting "substantially each reference in an entry" as invariant references as recited in independent claims 8, 12, 21, 25, 42, 46, 55, and 59, it would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to have done so in the Wetmore '713 Patent in view of the Dummermuth '853 Patent.

Namely, in the Wetmore '713 Patent, when the entry points (304) are jumps to labels somewhere else in the ROM code, they will be replaced by references to a table (table pointers) in RAM with an offset. However, reference to entry points (303), which themselves point to entry points (304), are not expressly disclosed as being replaced or modified. This, of course, is because there is no need to change reference to entry points (303) because they would not

RedBend0008139

change when updating the ROM program and thus do not need to be accounted for as would normal lines of code with internal references to other lines of code. Rather, any code that may need to be changed is already moved out into RAM and referenced by the entry point (304) replaced by the table pointer and offset.

The problem of shifting that occurs with deletions and insertions into code was already appreciated as early as 1978, the issue date of the Dummermuth '853 Patent, and it is notable that the Dummermuth '853 Patent provided the same solution to the problem that the '552 Patent relies so heavily upon, namely that internal references in program code are replaced by invariant jumps to a target in a table. In doing so, the Dummermuth '853 Patent explained that:

> The GTO instruction (or any other conditional or unconditional jump type instruction) must identify the target memory address to which the system is to jump. It should be apparent, however, that if the target memory address is specified directly in the GTO instruction, it must be changed each time an editing function is performed which involves the shifting of controller instruction memory addresses. In other words, the editing functions such as "insert" or "delete" may change the memory address of the point, or target, in the control program to which the system is to jump thus necessitating the changing of the GTO instruction. Therefore, the target memory address of each GTO instruction is indirectly identified by a target number which accompanies the operation code and which refers to a particular location in a jump table 64. The jump table 64 is then employed to obtain the desired target memory address. (Dummermuth '853 Patent col.12 l.54 – col.13 l.3.)

Claim charts summarizing the substantial new question of patentability raised by the Wetmore '713 Patent in view of the 1990 Dummermuth '853 Patent of the server-side independent claims (claims 1, 8, 14, 21, 35, 42, 48, and 55) and server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 of the '552 Patent under 35 U.S.C. § 103(a) are provided below:

Set 1: Server-Side Independent Claims 1, 14, 35, And 48:

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
1. A method for generating a compact difference result between an old executable program and a new executable program; each program including	14. A system for generating a compact difference result between an old executable program and a new executable program; each program including reference entries	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old executable program (old non-vectorized program) and a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) includes reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.

RedBend0008140

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
reference entries that contain reference that refer to other entries in the program; the method comprising the steps of:	that contain reference that refer to other entries in the program; the system comprising a processing device capable of:	
(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, whereby a modified old program is generated;	(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, whereby a modified old program is generated;	It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the Dummermuth '853 Patent to replace substantially each reference with jumps to the RAM table, such that the Wetmore '713 Patent would scan the old program (the old non-vectorized program) and, for substantially each **reference entry** (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a **distinct label mark** (a table pointer with an offset), whereby a modified old program (vectorized old program) is generated. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) Under this definition, in Wetmore, all of the jump entry points in the old non-vectorized program would be all of the references. The Wetmore '713 Patent replaces each jump entry point (reference) when vectorizing the code. **Distinct label marks**: A "label" is defined in the '552 Patent is be "an abstract notation of an entry

RedBend0008141

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		which is referred by another entry of the same data table through a reference." ('552 Patent col.2 ll.48-50.) "Label marks" are shown as marks (101)-(105) in FIG. 2A. An example of a "distinct label *number*" is described in the '552 Patent: "a distinct label *number* is assigned to each label mark of P''_1. By this particular example, the distinct labels are assigned in ascending order and, as shown, labels marks (101) to (105) are assigned with the respective values 1 to 5." ('552 Patent col.10 ll.63-67 (emphasis added).) Whether the '552 Patent's use of the claim term "distinct label *mark*" is broader than, or the same scope as a "distinct label *number*," Wetmore still meets this element. Namely, in the Wetmore '713 Patent, each table pointer with its offset (a numerical mark) into the vector table would be a "distinct label mark."
(b) scanning the new program and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new program is generated;	(b) scanning the new program and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new program is generated;	The process described above in (a) is repeated to replace substantially each reference with jumps to the RAM table, for the new program (the new non-vectorized program). Namely, during vectorization, the Wetmore '713 Patent, as modified by the teaching of the Dummermuth '853 Patent, scans the new program (the new non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of said entry by a distinct label mark (a table pointer with an offset), whereby a modified new program (vectorized new program) is generated.
(c) generating said difference result utilizing directly or indirectly at least said modified old program and modified new program.	(c) generating said difference result utilizing directly or indirectly at least said modified old program and modified new program.	The Wetmore '713 Patent generates the difference result (the vector patch resource) utilizing directly or indirectly at least the modified old program (vectorized old program) and modified new program (vectorized new program). The Wetmore '713 Patent discloses: "In the currently preferred embodiment of the present invention, a tool termed ROMPatch, is provided which automatically creates the vector patch resources. ROMPatch *compares the object files of two versions of the vectorized ROM code to identify routines which are different or new*. In the currently preferred embodiment, routines which are different is accomplished via a Cyclical Redundancy Check

RedBend0008142

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		(CRC) operation. ... In any event, when all the patched routines are found, the vector patch resource is generated." (Wetmore '713, col.10 l.66—col.11 l.12 (emphasis added).)

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
35. A method for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	48. A system for generating a compact difference result between an old data table and a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old **data table** (old non-vectorized program) and a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) includes **reference entries** ("jump" entry points) that contain **references** that refer to other **entries** in the data table. **Data table**: The '552 Patent defines a "data table" as "a table of entries, each may have a different size." ('552 Patent col.2 ll.33-34.) The '552 Patent also explains that "a data table can be an executable program either as a loaded program in machine-memory or as an executable-file. In this example, entries are individual machine instructions of the program or the individual data elements used by the program." ('552 Patent col.2 ll.61-65.) The Patentee also provided its claim construction for "data table" as "a table of entries, where an entry is an addressable unit within the data table. An executable program is one example of a data table. *See, e.g.*, '552 Patent 2:33-36; 2:61-63." (Edwards Decl. Exh. A.) Thus, the ROM based code of the Wetmore '713 Patent qualifies as a data table. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) The Patentee also provided its claim construction for "reference entry" as "An addressable unit containing data that includes a reference. *See, e.g.*, '552 Patent 2:35-36; 2:46-47." (Edwards Decl. Exh. A.) In Wetmore, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the

RedBend0008143

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) The Patentee also provided the identical claim construction for "reference" as "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. *See, e.g.*, '552 Patent 2:42-45." (Edwards Decl. Exh. A.) Under this definition, in the Wetmore '713 Patent, all of the jump entry points in the old non-vectorized program would be all of the references. **Entry:** The '552 Patent states in the Glossary that "a data table includes entries, each of which is an addressable unit that contains data." ('552 Patent col.2 ll.35-36.)
(a) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified old data table is generated;	(a) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified old data table is generated;	It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the Dummermuth '853 Patent to replace substantially each reference with jumps to the RAM table, such that the Wetmore '713 Patent would scan the old data table (the old non-vectorized program) and for substantially each **reference entry** (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a **distinct label mark** (a table pointer with an offset), whereby a modified old data table (vectorized old program) is generated. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) In the Wetmore '713 Patent, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to

RedBend0008144

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) Under this definition, in Wetmore, all of the jump entry points in the old non-vectorized program would be all of the references. The Wetmore '713 Patent replaces each jump entry point (reference) when vectorizing the code. **Distinct label marks**: A "label" is defined in the '552 Patent is be "an abstract notation of an entry which is referred by another entry of the same data table through a reference." ('552 Patent col.2 ll.48-50.) "Label marks" are shown as marks (101)-(105) in FIG. 2A. An example of a "distinct label *number*" is described in the '552 Patent: "a distinct label *number* is assigned to each label mark of P''_1. By this particular example, the distinct labels are assigned in ascending order and, as shown, labels marks (101) to (105) are assigned with the respective values 1 to 5." ('552 Patent col.10 ll.63-67 (emphasis added).) Whether the '552 Patent's use of the claim term "distinct label *mark*" is broader than, or the same scope as a "distinct label *number*," Wetmore still meets this element. Namely, in the Wetmore '713 Patent, each table pointer with its offset (a numerical mark) into the vector table would be a "distinct label mark."
(b) scanning the new data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new data table is generated;	(b) scanning the new data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby a modified new data table is generated;	The process described above for (a) is repeated to replace substantially each reference with jumps to the RAM table, for the new data table (the new non-vectorized program). Namely, during vectorization, the Wetmore '713 Patent, as modified by the teaching of the Dummermuth '853 Patent, scans the new data table (the new non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of said entry by a distinct label mark (a table pointer with an offset), whereby a modified new data table (vectorized new program) is generated.
(c) generating said	(c) generating	The Wetmore '713 Patent generates the

RedBend0008145

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
difference result utilizing directly or indirectly at least said modified old data table and modified new data table.	said difference result utilizing directly or indirectly at least said modified old data table and modified new data table.	difference result (the vector patch resource) utilizing directly or indirectly at least the modified old data table (vectorized old program) and modified new data table (vectorized new program). The Wetmore '713 Patent discloses: "In the currently preferred embodiment of the present invention, a tool termed ROMPatch, is provided which automatically creates the vector patch resources. ROMPatch *compares the object files of two versions of the vectorized ROM code to identify routines which are different or new.* In the currently preferred embodiment, routines which are different is accomplished via a Cyclical Redundancy Check (CRC) operation. … In any event, when all the patched routines are found, the vector patch resource is generated." (Wetmore '713, col.10 l.66—col.11 l.12 (emphasis added).)

Set 2: Server-Side Independent Claims 8, 21, 42, And 55:

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
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:	21. A system 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 system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old executable program (old non-vectorized program) and a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain reference that refer to other entries in the program.
(a) generating a modified old program utilizing at least said old	(a) generating a modified old program utilizing at least said old	During vectorization, the Wetmore '713 Patent generates a modified old program (vectorized old program) utilizing at least the old program (the old non-vectorized program).

RedBend0008146

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
program;	program;	
(b) generating a modified new program utilizing at least said new program,	(b) generating a modified new program utilizing at least said new program,	During vectorization, the Wetmore '713 Patent generates a modified new program (vectorized new program) utilizing at least the new program (the new non-vectorized 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;	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;	The modified old program (vectorized old program) and modified new program (vectorized new program) have at least the following characteristics: It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the Dummermuth '853 Patent to replace substantially each reference with jumps to the RAM table, such that substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old program (the old non-vectorized program) that is different than corresponding entry in said new program (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old program and new program (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new programs. **Invariant references**: Invariant references are discussed in the '552 Patent and an example is given of providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in

RedBend0008147

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows …." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old program and vectorized new program.
(c) generating said compact difference result utilizing at least said modified new program and modified old program.	(c) generating said compact difference result utilizing at least said modified new program and modified old program.	The Wetmore '713 Patent generates the compact difference result (the vector patch resource) utilizing at least the modified new program (vectorized new program) and modified old program (vectorized old program).

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
42. A method for generating a compact difference result between an old data table and a new data table;	55. A system for generating a compact difference result between an old data table and a new data table;	As discussed in more detail in each step below, the Wetmore '713 Patent generates a compact difference result (vector patch resource) between an old **data table** (old non-vectorized program) and a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) includes **reference entries** ("jump" entry points) that

RedBend0008148

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	contain references that refer to other entries in the data table. **Data table**: The '552 Patent defines a "data table" as "a table of entries, each may have a different size." ('552 Patent col.2 ll.33-34.) The '552 Patent also explains that "a data table can be an executable program either as a loaded program in machine-memory or as an executable-file. In this example, entries are individual machine instructions of the program or the individual data elements used by the program." ('552 Patent col.2 ll.61-65.) The Patentee also provided its claim construction for "data table" as "a table of entries, where an entry is an addressable unit within the data table. An executable program is one example of a data table. *See, e.g.*, '552 Patent 2:33-36; 2:61-63." (Edwards Decl. Exh. A.) Thus, the ROM based code of the Wetmore '713 Patent qualifies as a data table. **Reference entry**: The '552 Patent states in the Glossary that "Entries that include references are designated also as reference entries." ('552 Patent col.2 ll.46-47.) The Patentee also provided its claim construction for "reference entry" as "An addressable unit containing data that includes a reference. *See, e.g.*, '552 Patent 2:35-36; 2:46-47." (Edwards Decl. Exh. A.) In Wetmore, the entry points (304) may be "jump[s] to a label somewhere else in the ROM code." (Wetmore '713 Patent col.5 ll.38-39.) Under the '552 Patent's definition of "reference entries," the Wetmore '713 Patent's "jump" entry points (reference entries) include references to labels somewhere else in the ROM code. **Reference**: A "reference" is defined in the '552 Patent to be "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." ('552 Patent col.2 ll.42-45.) The Patentee also provided the identical claim construction for "reference" as "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

RedBend0008149

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		address. *See, e.g.*, '552 Patent 2:42-45." (Edwards Decl. Exh. A.) Under this definition, in the Wetmore '713 Patent, all of the jump entry points in the old non-vectorized program would be all of the references. **Entry:** The '552 Patent states in the Glossary that "a data table includes entries, each of which is an addressable unit that contains data." ('552 Patent col.2 ll.35-36.)
(a) generating a modified old data table utilizing at least said old data table;	(a) generating a modified old data table utilizing at least said old data table;	During vectorization, the Wetmore '713 Patent generates a modified old data table (vectorized old program) utilizing at least the old data table (the old non-vectorized program).
(b) generating a modified new data table utilizing at least said new data table,	(b) generating a modified new data table utilizing at least said new data table,	During vectorization, the Wetmore '713 Patent generates a modified new data table (vectorized new program) utilizing at least the new data table (the new non-vectorized program).
said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and	said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and	The modified old data table (vectorized old program) and modified new data table (vectorized new program) have at least the following characteristics: It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the Dummermuth '853 Patent to replace substantially each reference with jumps to the RAM table, such that substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old data table (the old non-vectorized program) that is different than corresponding entry in said new data table (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old data table and new data table (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new data tables. **Invariant references:** Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the

RedBend0008150

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
modified new data tables;	modified new data tables;	'552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent. col.10 ll.7-15.) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows ...." ('552 Patent. col.10 ll.47-50.) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs (data table) having table pointers with offsets are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offset will be "the same" in both the vectorized old program and vectorized new program.
(c) generating said compact difference result utilizing at least said modified new	(c) generating said compact difference result utilizing at least said modified new	The Wetmore '713 Patent generates the compact difference result (the vector patch resource) utilizing at least the modified new data table (vectorized new program) and modified old data table (vectorized old program).

RedBend0008151

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
data table and modified old data table.	data table and modified old data table.	

Server-Side Dependent Claims 4, 11, 17, And 24:

Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
4. The method of claim 1, further comprising the step of: (d) storing said compact difference result on a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the compact difference result (vector patch resource) is stored on a storage medium (system disk).
11. The method of claim 8, further comprising the step of: (d) storing said compact difference result on a storage medium.	
17. The system of claim 14, wherein said processor device is further capable of storing said compact difference result on a storage medium.	
24. The system of claim 21, wherein said processor is further capable of storing said compact difference result on a storage medium.	

Server-Side Dependent Claims 27/1, 27/4, 28/8, And 28/11:

Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
27 [27/1, 27/4]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 1 to 4.	The Wetmore '713 Patent provides a processing device having associated therewith a storage medium (system disk) which holds compact difference result data (vector patch resource) that was generated by the method of claims 1 and 4 [or 8-11].
28 [28/8, 28/11]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 8 to 11.	

RedBend0008152

Server-Side Dependent Claims 38, 45, 51, And 58:

Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
38. The method of claim 35, further comprising the step of: (d) storing said compact difference result on a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the compact difference result (vector patch resource) is stored on a storage medium (system disk).
45. The method of claim 42, further comprising the step of: (d) storing said compact difference result on a storage medium.	
51. The system of claim 48, wherein said processor device is further capable of storing said compact difference result on a storage medium.	
58. The system of claim 55, wherein said processor is further capable of storing said compact difference result on a storage medium.	

Server-Side Dependent Claims 61/35, 61/38, 62/42, and 62/45:

Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
61 [61/35, 61/38]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 35 to 38.	The Wetmore '713 Patent provides a processing device having associated therewith a storage medium (system disk) which holds compact difference result data (vector patch resource) that was generated by the method of claims 35 and 38 [or 42-45].
62 [62/42, 62/45]. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 42 to 45.	

Set 3: Client-Side Independent Claims 5, 18, 39, And 52:

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
5. A method for performing an update in an old executable program so as to generate a new executable program; each program including reference entries that contain reference that refer to other entries in the program; the method	18. A system for performing an update in an old executable program so as to generate a new executable program; each program including reference entries that contain reference that refer to other entries in the program; the system	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old executable program (old non-vectorized program) so as to generate a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.

RedBend0008153

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
comprising the steps of:	comprising a processing device capable of:	
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old program (vectorized old program) and a modified new program (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) 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, whereby the modified old program is generated;	(b) 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, whereby the modified old program is generated;	It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the Dummermuth '853 Patent to replace substantially each reference with jumps to the RAM table, such that the Wetmore '713 Patent would scan the old program (the old non-vectorized program) and, for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a distinct label mark (a table pointer with an offset), whereby a modified old program (vectorized old program) is generated.
(c) reconstituting the modified new program utilizing at least said compact difference result and said modified old program;	(c) reconstituting the modified new program utilizing at least said compact difference result and said modified old program;	The Wetmore '713 Patent reconstitutes a modified new program (vectorized new program) utilizing directly or indirectly at least the modified old program (vectorized old program) and the compact difference result (vector patch resource). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
said modified new program is differed from said new program at least in that substantially each reference entry in said new program is replaced in said modified new program by a distinct	said modified new program is differed from said new program at least in that substantially each reference entry in said new program is replaced in said modified new program by a distinct	In the Wetmore '713 Patent, as modified by the teaching of the Dummermuth '853 Patent, the modified new program (vectorized new program) is differed from the new program (new non-vectorized program) at least in that substantially each reference entry (each "jump" entry point) in the new program is replaced in the modified new program by a distinct label mark (a table pointer with an offset).

RedBend0008154

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
label mark;	label mark;	
(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new programs, to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the

RedBend0008155

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
39. A method for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	52. A system for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old data table (old non-vectorized program) so as to generate a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the data table.
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old data table (vectorized old program) and a modified new data table (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old data table is generated;	(b) scanning the old data table and for substantially each reference entry perform steps that include: (i) replacing the reference of said entry by a distinct label mark, whereby the modified old data table is generated;	It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the Dummermuth '853 Patent to replace substantially each reference with jumps to the RAM table, such that the Wetmore '713 Patent would scan the old data table (the old non-vectorized program) and for substantially each reference entry (each "jump" entry point), replaces the reference (the location in the ROM code in the jump entry point) of the entry by a distinct label mark (a table pointer with an offset), whereby a modified old data table (vectorized old program) is generated.

RedBend0008156

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
(c) reconstituting the modified new data table utilizing at least said compact difference result and said modified old data table;	(c) reconstituting the modified new data table utilizing at least said compact difference result and said modified old data table;	The Wetmore '713 Patent reconstitutes a modified new data table (vectorized new program) utilizing directly or indirectly at least the modified old data table (vectorized old program) and the compact difference result (vector patch resource). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
said modified new data table is differed from said new data table at least in that substantially each reference entry in said new data table is replaced in said modified new data table by a distinct label mark;	said modified new data table is differed from said new data table at least in that substantially each reference entry in said new data table is replaced in said modified new data table by a distinct label mark;	In the Wetmore '713 Patent, as modified by the teaching of the Dummermuth '853 Patent, the modified new data table (vectorized new program) is differed from the new data table (new non-vectorized program) at least in that substantially each reference entry (each "jump" entry point) in the new data table is replaced in the modified new data table by a distinct label mark (a table pointer with an offset). (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new programs, to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one

RedBend0008157

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

Set 4: Client-Side Independent Claims 12, 25, 46, And 59:

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
12. A method for performing an update in an old executable program so as to generate 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:	25. A system for performing an update in an old executable program so as to generate a new executable program; each program including reference entries that contain reference that refer to other entries in the program; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old executable program (old non-vectorized program) so as to generate a new executable program (new non-vectorized program). Each program (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the program.
(a) receiving data	(a) receiving data	The client in the Wetmore '713 Patent

RedBend0008158

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;	receives data that includes a compact difference result (vector patch resource) that was generated using a modified old program (vectorized old program) and a modified new program (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) generating a modified old program utilizing at least said old program;	(b) generating a modified old program utilizing at least said old program;	During vectorization, the Wetmore '713 Patent generates a modified old program (vectorized old program) utilizing at least the old program (the old non-vectorized program). (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(c) reconstituting a modified new program utilizing directly or indirectly at least said modified old program and said compact difference result;	(c) reconstituting a modified new program utilizing directly or indirectly at least said modified old program and said compact difference result;	The Wetmore '713 Patent reconstitutes a modified new program (vectorized new program) utilizing directly or indirectly at least the modified old program (vectorized old program) and the compact difference result (vector patch resource). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
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/inset 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	said modified old program and modified new programs 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/inset 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	In the Wetmore '713 Patent, the modified old program (vectorized old program) and modified new program (vectorized new program) have at least the following characteristics: It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the Dummermuth '853 Patent to replace substantially each reference with jumps to the RAM table, such that substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old program (the old non-vectorized program) that is different than corresponding entry in said new program (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old program and new program (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new programs.

RedBend0008159

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
modified old and modified new programs;	programs;	**Invariant references:** Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows ...." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized

RedBend0008160

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old program and vectorized new program.
(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	(d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new executable programs (*i.e.*, data tables), to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the

RedBend0008161

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
46. A method for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the method comprising the steps of:	59. A system for performing an update in an old data table so as to generate a new data table; each data table including reference entries that contain reference that refer to other entries in the data table; the system comprising a processing device capable of:	As discussed in more detail in each step below, the Wetmore '713 Patent performs an update in an old data table (old non-vectorized program) so as to generate a new data table (new non-vectorized program). Each data table (old and new non-vectorized programs) including reference entries ("jump" entry points) that contain references (the location in the ROM code in the jump entry point) that refer to other entries in the data table.
(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	(a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old data table and a modified new data table;	The client in the Wetmore '713 Patent receives data that includes a compact difference result (vector patch resource) that was generated using a modified old data table (vectorized old program) and a modified new data table (vectorized new program). (Wetmore '713 Patent col.10 l.62—col.11 l.12.)
(b) generating a modified old data table utilizing at least said old data table;	(b) generating a modified old data table utilizing at least said old data table;	During vectorization, the Wetmore '713 Patent generates a modified old data table (vectorized old program) utilizing at least the old data table (the old non-vectorized program). (Wetmore '713 Patent col.5 ll.18-56; col.6 l.45 – col.8 l.52; FIGS. 3-5.)
(c) reconstituting a modified new data table utilizing	(c) reconstituting a modified new data table utilizing directly	The Wetmore '713 Patent reconstitutes a modified new data table (vectorized new program) utilizing directly or indirectly at least the modified

RedBend0008162

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
directly or indirectly at least said modified old data table and said compact difference result;	or indirectly at least said modified old data table and said compact difference result;	old data table (vectorized old program) and the compact difference result (vector patch resource). (Wetmore '713 Patent col.4 ll.38-39; col.5 ll.1-3, 10-17; col.10 ll.6-14, col.11 ll.2-12; FIG. 4.)
said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/inset modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	said modified old data table and modified new data table have at least the following characteristics: (i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/inset modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	In the Wetmore '713 Patent, the modified old data table (vectorized old program) and modified new data table (vectorized new program) have at least the following characteristics: It would have been obvious to modify the teaching of the Wetmore '713 Patent with the teaching of the Dummermuth '853 Patent to replace substantially each reference with jumps to the RAM table, such that substantially each reference (the location in the ROM code in the jump entry point) in an entry (in each "jump" entry point) in the old data table (the old non-vectorized program) that is different than corresponding entry in said new data table (the new non-vectorized program) due to delete/insert modifications that form part of the transition between the old data table and new data table (*e.g.*, routines in the Wetmore '713 Patent can be deleted or inserted), are reflected as invariant references (table pointers with an offset) in the corresponding entries in the modified old and modified new data table. **Invariant references**: Invariant references are discussed in the '552 Patent and an example is given providing *invariant references* by generating modified old and new programs or data tables wherein the address references in entries are replaced by label marks. Namely, the '552 Patent states that: "the invention aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as *invariant entries* in the modified old and new programs. The net effect is that the *invariant reference entries* (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing

RedBend0008163

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dunnermuth '853 Patent
		its size as compared to a conventional difference result obtained by using hitherto known techniques." ('552 Patent. col.3 ll.36-46 (emphasis added)). The '552 Patent also states: "Those versed in the art will readily appreciate that according to the invention, it is desired to neutralize this change, since it has occurred solely due to the fact that other entries have been affected (*i.e.* entries 7 to 9). It is accordingly an object of the invention to give rise to a situation where modifications of this kind will be modified to *invariant references* with the obvious consequence that they are not reflected in the difference result, thereby keeping the latter relatively compact." ('552 Patent col.10 ll.7-15 (emphasis added).) Further, the '552 Patent notes that for the particular embodiment of FIGS. 1 and 2, "the desired *invariant references* are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks as follows …." ('552 Patent col.10 ll.47-50 (emphasis added).) The Patentee also provided its claim construction for "invariant references" as "References that are the same. *See, e.g.*, '552 Patent 10:10-15." (Edwards Decl. Exh. A.) Thus, the vectorization process in the Wetmore '713 Patent that results in vectorized programs having table pointers with an offset are "invariant references" under the '552 Patent's label marks example and also under the Patent Owner's construction, since the table pointers with offsets will be "the same" in both the vectorized old data table and vectorized new data table.
(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	(d) reconstituting said new data table utilizing directly or indirectly at least said compact difference result and said modified new data table.	It would have been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of sending compact difference results based on a comparison of modified old and new executable programs (*i.e.*, data tables), to reconstitute the new program directly or indirectly using the compact difference result and the modified new program when reconstituting a program, such as a program run primarily from RAM like in the IBM-type

RedBend0008164

Claim Element	Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
		operating systems discussed in the Wetmore '713 Patent. While the vectorized version of the new program is run because the Wetmore '713 Patent involves patching of code that is run both from the ROM (which is not modifiable) and from the RAM (which is modifiable), one of ordinary skill in the art at the time of the alleged invention of the '552 Patent interested in updating code that runs in RAM, applying the teaching of the Wetmore '713 Patent, would be motivated by common sense and knowledge of basic programming techniques to convert the vectorized code back into non-vectorized code in order to minimize the number of jumps in the final code to make the software run faster. It would have also been obvious to one of ordinary skill in the art at the time of filing of the '552 Patent, in view of the teaching of the Wetmore '713 Patent of how to convert programs into modified (vectorized) form, to regenerate the new program back into non-modified (non-vectorized) form using directly or indirectly at least the compact difference result and the modified (vectorized) new program. It would further have been obvious to one of ordinary skill in the art to go back from the vectorized new program to the non-vectorized new program if one wanted to examine the contents of the program without the use of jump tables, such as to compare the non-vectorized old program for debugging, or for version control or the like. The fact that Wetmore does not need to go back to the non-vectorized format to run in a combined ROM/RAM operating system does not detract from the simplicity and obviousness to one of ordinary skill in the art to go back to the original, non-vectorized format for normal programming-related reasons such as for debugging or version control.

RedBend0008165

Client-Side Dependent Claims 13, 26, 31, 34, 47, 60, 65, And 68:

Claim Element	The Wetmore '713 Patent In View Of The Dummermuth '853 Patent
13. The method of claim 5, wherein said data is received in step (a) from a storage medium.	To perform patching, the Wetmore '713 Patent explains that the client is provided with a loading routine via the system disk that loads the proper vector resource patches following a version check. (Wetmore '713 Patent col.11 ll.34-40.) Thus, the data of the compact difference result (vector patch resource) is received from a storage medium (system disk). See claims 5, 18, 12, 25, 39, 59, 46, and 59 above.
26. The system of claim 18, wherein said data is received in step (a) from a storage medium.	
31. The method of claim 12, wherein said data is received in step (a) from a storage medium.	
34. The system of claim 25, wherein said data is received in step (a) from a storage medium.	
47. The method of claim 39, wherein said data is received in step (a) from a storage medium.	
60. The system of claim 59, wherein said data is received in step (a) from a storage medium.	
65. The method of claim 46, wherein said data is received in step (a) from a storage medium.	
68. The system of claim 59, wherein said data is received in step (a) from a storage medium.	

Accordingly, each and every limitation of (a) the server-side independent claims (*i.e.*, claims 1, 8, 14, 21, 35, 42, 48, and 55) and server-side dependent claims 4, 11, 17, 24, 27/1, 27/4, 28/8, 28/11, 38, 45, 51, 58, 61/35, 61/38, 62/42, and 62/45 [Q4] and (b) the client-side independent claims (*i.e.*, claims 5, 12, 18, 25, 39, 46, 52, and 59) and dependent claims 13, 26, 31, 34, 47, 60, 65, and 68 [Q10] is rendered obvious under 35 U.S.C. § 103 by the Wetmore '713 Patent in view of the Dummermuth '853 Patent.

4. Obviousness Based On The Wetmore '713 Patent In View Of The Sadowsky '796 Patent Or The Coppieters Article [Q5]

For the reasons set forth below, a substantial new question of patentability under 35 U.S.C. § 103(a) of the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent [Q5], all of which relate to transmitting the compact difference result

RedBend0008166

over a communications network or the Internet, is raised by the Wetmore '713 Patent in view of the Sadowsky '796 Patent or the Coppieters Article.

In short, and as is discussed more fully below, it would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to transmit the compact difference result over a communications network such as the Internet, since the prior art disclosed the sending of software patches and updates via communications networks, including via the Internet.

Namely, the Sadowsky '796 Patent discloses updating software using a host server such as a Compuserve™ host, a conventional Web page, or a file server and the communications channel may be, for example, the Internet. Moreover, the Sadowsky '796 Patent explains that programs can be updated either by new disks (as is disclosed in the Wetmore '713 Patent), via a BBS, an Internet service provider, or the Internet. Furthermore, the Coppieters Article discloses sending updates of binary files over a "communications network," and the communications network may be, for example, the Internet.

Thus, it was known to those of ordinary skill in the art at the time of the alleged invention of the '552 Patent, and merely a choice of the software provider, to provide patches or updates to software residing at a client computer via a communications network or the Internet since either could be done. However, a skilled artisan could simply follow the teaching of the Sadowsky '796 Patent that one could avoid using disks, which the Sadowsky '796 Patent described as time consuming and frequently costing money, by simply updating the software by accessing files from the host computer via a communications link such as the Internet as taught by the Sadowsky '796 Patent (and as taught in the Coppieters Article).

Claim charts summarizing the substantial new question of patentability raised by the Wetmore '713 Patent in view of the Sadowsky '796 Patent or the Coppieters Article of the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent under 35 U.S.C. § 103(a) are provided below:

RedBend0008167

Server-Side Dependent Claims 2, 9, 15, 22, 36, 43, 49, 56:

Claim Element	Wetmore '713 Patent In View Of The Sadowsky '796 Patent Or The Coppieters Article
2. The method of claim 1, further comprising the step of: (d) transmitting said compact difference result over a communication network.	It would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to provide patches or updates to software residing at a client computer via a communications network, including the Internet, instead of using disks as taught by the Wetmore '713 Patent, since both the Coppieters Article and the Sadowsky '796 Patent taught the desirability of using a communications network to send updates and patches to programs to the remote client. See charts for claims 1, 8, 14, 21, 35, 42, 48 and 55 above.
9. The method of claim 8, further comprising the step of: (d) transmitting said compact difference result over a communication network.	
15. The system of claim 14, wherein said processor device is further capable of transmitting said compact difference result over a communication network.	
22. The system of claim 21, wherein said processor is further capable of transmitting said compact difference result over a communication network.	
36. The method of claim 35, further comprising the step of: (d) transmitting said compact difference result over a communication network.	
43. The method of claim 42, further comprising the step of: (d) transmitting said compact difference result over a communication network.	
49. The system of claim 48, wherein said processor device is further capable of transmitting said compact difference result over a communication network.	
56. The system of claim 55, wherein said processor is further capable of transmitting said compact difference result over a communication network.	

Server-Side Dependent Claims 3, 7, 10, 16, 20, 23, 37, 41, 44, 50, 54, 57, 64, 67:

Claim Element	Wetmore '713 Patent In View Of The Sadowsky '796 Patent Or The Coppieters Article
3. The method of claim 2, wherein said network includes the Internet.	It would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to provide patches or updates to software residing at a client computer via a communications
7. The method of claim 6, wherein said network includes the Internet.	
10. The method of claim 9, wherein said	

RedBend0008168

Claim Element	Wetmore '713 Patent In View Of The Sadowsky '796 Patent Or The Coppieters Article
network includes the Internet.	network, including the Internet, instead of using disks as taught by the Wetmore '713 Patent, since both the Coppieters Article and the Sadowsky '796 Patent taught the desirability of using a communications network to send updates and patches to programs to the remote client. See charts for claims 2, 9, 15, 22, 36, 43, 49, and 57 above.
16. The system of claim 15, wherein said network includes the Internet.	
20. The system of claim 19, wherein said network includes the Internet.	
23. The system of claim 22, wherein said network includes the Internet.	
30. The method of claim 29, wherein said network includes the Internet.	
33. The system of claim 32, wherein said network includes the Internet.	
37. The method of claim 36, wherein said network includes the Internet.	
41. The method of claim 40, wherein said network includes the Internet.	
44. The method of claim 43, wherein said network includes the Internet.	
50. The system of claim 49, wherein said network includes the Internet.	
54. The system of claim 53, wherein said network includes the Internet.	
57. The system of claim 56, wherein said network includes the Internet.	
64. The method of claim 63, wherein said network includes the Internet.	
67. The system of claim 66, wherein said network includes the Internet.	

Server-Side Dependent Claims 27/2, 27/3, 28/9, 28/10, 61/36, 61/37, 62/43, 62/44:

Claim Element	Wetmore '713 Patent In View Of The Sadowsky '796 Patent Or The Coppieters Article
27. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 1 to 4. [as to claims 2 and 3]	It would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to provide patches or updates to software residing at a client computer via a network, including the Internet, instead of using disks as taught by the Wetmore '713 Patent, since both the Coppieters Article and the Sadowsky '796 Patent taught the desirability of using a communications network to send updates and patches to programs to the remote client.
28. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 8 to 11. [as to claims 9 and 10]	
61. A processing device having associated	

RedBend0008169

Claim Element	Wetmore '713 Patent In View Of The Sadowsky '796 Patent Or The Coppieters Article
therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 35 to 38. [as to claims 36 and 37]	See charts for claims 1-3, 8-10, 35-37, and 42-44, above.
62. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 42 to 45. [as to claims 62/43 and 62/44]	

Accordingly, each and every limitation of the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent [Q5] is rendered obvious under 35 U.S.C. § 103 by the Wetmore '713 Patent in view of the Sadowsky '796 Patent or the Coppieters Article.

5. Obviousness Based On The Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin In Further View Of The Sadowsky '796 Patent Or The Coppieters Article [Q6, Q11]

For the reasons set forth below, a substantial new question of patentability under 35 U.S.C. § 103(a) of (a) the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent [Q6], and (b) the remaining client-side dependent claims (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) of the '552 Patent [Q11], all of which relate to transmitting the compact difference result over a communications network or the Internet, is raised by the Wetmore '713 Patent in view of the 1990 IBM Technical Disclosure Bulletin and further in view of either the Sadowsky '796 Patent or the Coppieters Article.

The obviousness of the combination of the Wetmore '713 Patent with the teaching of the 1990 IBM Technical Disclosure Bulletin was already set forth above in Part IV.C.2. It would have been further obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to transmit the compact difference result over a communications network such as the Internet, since the prior art disclosed the sending of software patches and updates via communications networks, including via the Internet.

RedBend0008170

The Sadowsky '796 Patent discloses updating software using a host server such as a Compuserve™ host, a conventional Web page, or a file server and the communications channel may be, for example, the Internet. Moreover, the Sadowsky '796 Patent explains that programs can be updated either by new disks (as is disclosed in the Wetmore '713 Patent) or via a BBS, an Internet service provider, or the Internet. Furthermore, the Coppieters Article discloses sending updates of binary files over a "communications network," and the communications network may be, for example, the Internet.

Thus, it was known to those of ordinary skill in the art at the time of the alleged invention of the '552 Patent, and merely a choice of the software provider, to provide patches or updates to software residing at a client computer via a communications network or the Internet since either could be done. However, a skilled artisan could simply follow the teaching of the Sadowsky '796 Patent that one could avoid using disks, which the Sadowsky '796 Patent described as time consuming and frequently costing money, by simply updating the software by accessing files from the host computer via a communications link such as the Internet as taught by the Sadowsky '796 Patent (and as taught in the Coppieters Article).

Claim charts summarizing the substantial new question of patentability raised by the Wetmore '713 Patent in view of the 1990 IBM Technical Disclosure Bulletin, in further view of the Sadowsky '796 Patent or the Coppieters Article of (a) the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent [Q6], and (b) the remaining client-side dependent claims (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) of the '552 Patent [Q11] under 35 U.S.C. § 103(a) are provided below:

Server-Side Dependent Claims 2, 9, 15, 22, 36, 43, 49, 56:

Claim Element	Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
2. The method of claim 1, further comprising the step of: (d) transmitting said compact difference result over a communication network.	It would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to provide patches or updates to software residing at a client computer via a communications network, including the Internet, instead of using disks as taught by the Wetmore '713 Patent as modified by the 1990 IBM
9. The method of claim 8, further comprising the step of: (d) transmitting said compact difference result over a communication network.	

RedBend0008171

Claim Element	Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
15. The system of claim 14, wherein said processor device is further capable of transmitting said compact difference result over a communication network.	Technical Disclosure Bulletin, since both the Coppieters Article and the Sadowsky '796 Patent taught the desirability of using a communications network to send updates and patches to programs to the remote client. See charts for claims 1, 8, 14, 21, 35, 42, 48 and 55 above.
22. The system of claim 21, wherein said processor is further capable of transmitting said compact difference result over a communication network.	
36. The method of claim 35, further comprising the step of: (d) transmitting said compact difference result over a communication network.	
43. The method of claim 42, further comprising the step of: (d) transmitting said compact difference result over a communication network.	
49. The system of claim 48, wherein said processor device is further capable of transmitting said compact difference result over a communication network.	
56. The system of claim 55, wherein said processor is further capable of transmitting said compact difference result over a communication network.	

Dependent Claims 3, 7, 10, 16, 20, 23, 37, 41, 44, 50, 54, 57, 64, 67:

Claim Element	Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
3. The method of claim 2, wherein said network includes the Internet.	It would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to provide patches or updates to software residing at a client computer via a communications network, including the Internet, instead of using disks as taught by the Wetmore '713 Patent as modified by the 1990 IBM Technical Disclosure Bulletin, since both the Coppieters Article and the Sadowsky '796 Patent taught the desirability of using a
7. The method of claim 6, wherein said network includes the Internet.	
10. The method of claim 9, wherein said network includes the Internet.	
16. The system of claim 15, wherein said network includes the Internet.	
20. The system of claim 19, wherein said network includes the Internet.	
23. The system of claim 22, wherein said	

RedBend0008172

Claim Element	Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
network includes the Internet.	communications network to send updates and patches to programs to the remote client. See charts for claims 2, 9, 15, 19, 22, 29, 32, 36, 40. 43. 49, 53, 56, 63 and 66 above.
30. The method of claim 29, wherein said network includes the Internet.	
33. The system of claim 32, wherein said network includes the Internet.	
37. The method of claim 36, wherein said network includes the Internet.	
41. The method of claim 40, wherein said network includes the Internet.	
44. The method of claim 43, wherein said network includes the Internet.	
50. The system of claim 49, wherein said network includes the Internet.	
54. The system of claim 53, wherein said network includes the Internet.	
57. The system of claim 56, wherein said network includes the Internet.	
64. The method of claim 63, wherein said network includes the Internet.	
67. The system of claim 66, wherein said network includes the Internet.	

Server-Side Dependent Claims 27/2, 27/3, 28/9, 28/10, 61/36, 61/37, 62/43, 62/44:

Claim Element	Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
27. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 1 to 4. [as to claims 2 and 3]	It would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to provide patches or updates to software residing at a client computer via a network, including the Internet, instead of using disks as taught by the Wetmore '713 Patent as modified by the 1990 IBM Technical Disclosure Bulletin, since both the Coppieters Article and the Sadowsky '796 Patent taught the desirability of using a communications network to send updates and patches to programs to the remote client.
28. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 8 to 11. [as to claims 9 and 10]	
61. A processing device having associated therewith a storage medium which holds compact difference result data that was	

RedBend0008173

Claim Element	Wetmore '713 Patent In View Of The 1990 IBM Technical Disclosure Bulletin In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
generated by the method of anyone of claims 35 to 38. [as to claims 36 and 37]	See charts for claims 1-3, 8-10, 35-37, and 42-44, above.
62. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 42 to 45. [as to claims 62/43 and 62/44]	

Accordingly, each and every limitation of (a) the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent [Q6] and (b) the remaining client-side dependent claims (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) of the '552 Patent [Q11], is rendered obvious under 35 U.S.C. § 103 by the Wetmore '713 Patent in view of the 1990 IBM Technical Disclosure Bulletin, in further view of the Sadowsky '796 Patent or the Coppieters Article.

6. Obviousness Based On The Wetmore '713 Patent In View Of The Dummermuth '853 Patent In Further View Of The Sadowsky '796 Patent Or The Coppieters Article [Q7, Q12]

For the reasons set forth below, a substantial new question of patentability under 35 U.S.C. § 103(a) of (a) the remaining server-side dependent claims (*i.e.*, dependent claims 2, 3, 9, 10, 15, 16, 22, 23, 27/2, 27/3, 28/9, 28/10, 36, 37, 43, 44, 49, 50, 56, 57, 61/36, 61/37, 62/43, and 62/44) of the '552 Patent [Q7], and (b) the remaining client-side dependent claims (*i.e.*, dependent claims 6, 7, 19, 20, 29, 30, 32, 33, 40, 41, 53, 54, 63, 64, 66, and 67) of the '552 Patent [Q12], all of which relate to transmitting the compact difference result over a communications network or the Internet, is raised by the Wetmore '713 Patent in view of the Dummermuth '853 Patent, in further view of the Sadowsky '796 Patent or the Coppieters Article.

The obviousness of the combination of The Wetmore '713 Patent with the teaching of the Dummermuth '853 Patent was already set forth above in Part IV.C.3. It would have been further obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to transmit the compact difference result over a communications network such as the Internet, since

RedBend0008174

the prior art disclosed the sending of software patches and updates via communications networks, including via the Internet.

Namely, the Sadowsky '796 Patent discloses updating software using a host server such as a Compuserve™ host, a conventional Web page, or a file server and the communications channel may be, for example, the Internet. Moreover, the Sadowsky '796 Patent explains that programs can be updated either by new disks (as is disclosed in the Wetmore '713 Patent), via a BBS, an Internet service provider, or the Internet. Furthermore, the Coppieters Article discloses sending updates of binary files over a "communications network," and the communications network may be, for example, the Internet.

Thus, it was known to those of ordinary skill in the art at the time of the alleged invention of the '552 Patent, and merely a choice of the software provider, to provide patches or updates to software residing at a client computer via a communications network or the Internet since either could be done. However, a skilled artisan could simply follow the teaching of the Sadowsky '796 Patent that one could avoid using disks, which the Sadowsky '796 Patent described as time consuming and frequently costing money, by simply updating the software by accessing files from the host computer via a communications link such as the Internet as taught by the Sadowsky '796 Patent (and as taught in the Coppieters Article).

Server-Side Dependent Claims 2, 9, 15, 22, 36, 43, 49, 56:

Claim Element	Wetmore '713 Patent In View Of The Dummermuth '853 Patent In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
2. The method of claim 1, further comprising the step of: (d) transmitting said compact difference result over a communication network.	It would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to provide patches or updates to software residing at a client computer via a communications network, including the Internet, instead of using disks as taught by the Wetmore '713 Patent as modified by the Dummermuth '853 Patent, since both the Coppieters Article and the Sadowsky '796 Patent taught the desirability of using a communications network to send updates and patches to programs to the remote client. See charts for claims 1, 8, 14, 21, 35, 42, 48 and 55 above.
9. The method of claim 8, further comprising the step of: (d) transmitting said compact difference result over a communication network.	
15. The system of claim 14, wherein said processor device is further capable of transmitting said compact difference result over a communication network.	
22. The system of claim 21, wherein said processor is further capable of transmitting said compact difference result over a communication network.	
36. The method of claim 35, further	

RedBend0008175

Claim Element	Wetmore '713 Patent In View Of The Dummermuth '853 Patent In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
comprising the step of: (d) transmitting said compact difference result over a communication network.	
43. The method of claim 42, further comprising the step of: (d) transmitting said compact difference result over a communication network.	
49. The system of claim 48, wherein said processor device is further capable of transmitting said compact difference result over a communication network.	
56. The system of claim 55, wherein said processor is further capable of transmitting said compact difference result over a communication network.	

Dependent Claims 3, 7, 10, 16, 20, 23, 37, 41, 44, 50, 54, 57, 64, 67:

Claim Element	Wetmore '713 Patent In View Of The Dummermuth '853 Patent In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
3. The method of claim 2, wherein said network includes the Internet.	It would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to provide patches or updates to software residing at a client computer via a communications network, including the Internet, instead of using disks as taught by the Wetmore '713 Patent as modified by the Dummermuth '853 Patent, since both the Coppieters Article and the Sadowsky '796 Patent taught the desirability of using a communications network to send updates and patches to programs to the remote client. See charts for claims 2, 6, 9, 15, 19, 22, 29, 32, 36, 40, 43, 49, 53, 56, 63, and 66 above.
7. The method of claim 6, wherein said network includes the Internet.	
10. The method of claim 9, wherein said network includes the Internet.	
16. The system of claim 15, wherein said network includes the Internet.	
20. The system of claim 19, wherein said network includes the Internet.	
23. The system of claim 22, wherein said network includes the Internet.	
30. The method of claim 29, wherein said network includes the Internet.	
33. The system of claim 32, wherein said network includes the Internet.	
37. The method of claim 36, wherein said network includes the Internet.	
41. The method of claim 40, wherein said network includes the Internet.	
44. The method of claim 43, wherein said	

RedBend0008176

Claim Element	Wetmore '713 Patent In View Of The Dummermuth '853 Patent In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
network includes the Internet.	
50. The system of claim 49, wherein said network includes the Internet.	
54. The system of claim 53, wherein said network includes the Internet.	
57. The system of claim 56, wherein said network includes the Internet.	
64. The method of claim 63, wherein said network includes the Internet.	
67. The system of claim 66, wherein said network includes the Internet.	

Server-Side Dependent Claims 27/2, 27/3, 28/9, 28/10, 61/36, 61/37, 62/43, 62/44:

Claim Element	Wetmore '713 Patent In View Of The Dummermuth '853 Patent In Further View Of The Sadowsky '796 Patent Or The Coppieters Article
27. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 1 to 4. [as to claims 2 and 3]	It would have been obvious to one of ordinary skill in the art at the time of the alleged invention of the '552 Patent to provide patches or updates to software residing at a client computer via a network, including the Internet, instead of using disks as taught by the Wetmore '713 Patent as modified by the Dummermuth '853 Patent, since both the Coppieters Article and the Sadowsky '796 Patent taught the desirability of using a communications network to send updates and patches to programs to the remote client. See charts for claims 1-3, 8-10, 35-37, and 42-44, above.
28. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 8 to 11. [as to claims 9 and 10]	
61. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 35 to 38. [as to claims 36 and 37]	
62. A processing device having associated therewith a storage medium which holds compact difference result data that was generated by the method of anyone of claims 42 to 45. [as to claims 62/43 and 62/44]	

RedBend0008177

V. CONCLUSION

For the reasons explained above, reexamination of the '552 Patent is respectfully requested pursuant to 35 U.S.C. §§ 302-307 and 37 C.F.R. § 1.510. The examiner should conclude that substantial new questions of patentability are raised under 35 U.S.C. §§ 102 and 103 by the prior art references identified herein. Further the subject claims for which reexamination is requested, namely, claims 1-68 of the '552 Patent, should, for the reasons presented above and such other reasons as the examiner may identify, be rejected as anticipated and/or obvious over prior art, and thus cancelled from any reexamination certificate issued for the '552 Patent.

Dated: January 22, 2010

Respectfully submitted,
Google Inc.

By _____
Jonathan A. David
Registration No.: 36,494
LERNER, DAVID, LITTENBERG,
KRUMHOLZ & MENTLIK, LLP
600 South Avenue West
Westfield, New Jersey 07090
(908) 654-5000
Attorney for Reexamination Requester

RedBend0008178

INDEX OF ATTACHMENTS TO REQUEST FOR REEXAMINATION

Attachment A: U.S. Patent No. 6,546,552 (**"the '552 Patent"**) (double-column copy of the subject patent)

Attachment B: U.S. Patent No. 5,481,713 to Wetmore *et al.*, entitled "Method And Apparatus For Patching Code Residing On A Read Only Memory Device," issued on January 2, 1996 **(the "Wetmore '713 Patent")**

Attachment C: IBM Technical Disclosure Bulletin, Batalden, G.D., *et al.*, "Maintainable ROS Code Through The Combination of ROM And EEPROM." Vol.32 No. 9A, p.273-76, first published in February, 1990 **(the "1990 IBM Technical Disclosure Bulletin")**

Attachment D: U.S. Patent No. 4,111,853 to Dummermuth, entitled "Jump Structure For A Digital Control System," filed on December 21, 1976, and issued on September 19, 1978 **(the "Dummermuth '853 Patent")**

Attachment E: U.S. Patent No. 5,790,796 to Sadowsky, entitled "Polymorphic Package Files To Update Software Components," filed on June 14, 1996, and issued on August 4, 1998 **(the "Sadowsky '796 Patent")**

Attachment F: Coppieters, K., "A Cross-Platform Binary Diff," Dr. Dobb's Journal, US, San Mateo, California, pp. 32, XP 000610668, was published in May 1995 **(the "Coppieters Article")**

Attachment G: Declaration of Prof. Stephen A. Edwards, dated November 17, 2009, filed in *Red Bend Ltd. v. Google Inc.* (D. Mass. 10/06) **(the "Edwards Declaration")**

RedBend0008179

CERTIFICATE OF SERVICE

I hereby certify that on January 22, 2010, true copies of the following documents:

1. REQUEST FOR *EX PARTE* REEXAMINATION AND CERTIFICATE OF SERVICE WITH ATTACHMENTS A THROUGH G;

2. REQUEST FOR *EX PARTE* REEXAMINATION TRANSMITTAL FORM;

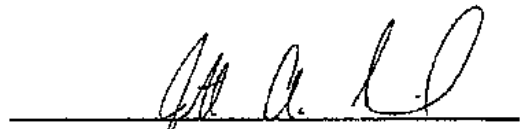
3. 37 CFR 1.501 INFORMATION DISCLOSURE CITATION IN A PATENT (SB-042);

4. COPIES OF ALL REFERENCES CITED ON SB-042; and

5. COPY OF U.S. PATENT NO. 6,546,552

were served upon the following by depositing same in the United States mail, first-class postage prepaid, and properly addressed as follows:

Kenneth H. Samples, Esq.
Fitch Even Tabin And Flannery
120 South La Salle Street
Suite 1600
Chicago, IL 60603



Jonathan A. David
Registration No. 36,494

Control No. [NOT YET ASSIGNED]

RedBend0008180