UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Apple Inc., Google Inc., and Motorola Mobility LLC

Petitioners,

v.

Arendi S.A.R.L.

Patent Owner.

Case No. IPR2014-00208

Patent No. 7,917,843

PATENT OWNER ARENDI S.A.R.L.'S PRELIMINARY RESPONSE UNDER 35 U.S.C. § 313 and 37 C.F.R. § 42.107

TABLE OF CONTENTS

EXHIBIT LIST

Arendi Exhibit Number	Description
2001	American Heritage College dictionary 3rd edition 1997 definition of the term "configure".

TABLE OF AUTHORITIES

Cases

Statutes

INTRODUCTION

Patent Owner Arendi S.A.R.L. ("Arendi" or "Patent Owner") respectfully requests that the Board decline to initiate *inter partes* review of claims 1-44 of U.S. Patent No. 7,917,843 (the "'843 Patent") because Petitioners Apple Inc., Google Inc., and Motorola Mobility LLC ("Petitioners") have failed to show that they have a reasonable likelihood of prevailing with respect to any of the challenged claims. 35 U.S.C. § 314.

Petitioners have submitted proposed grounds for challenge based on anticipation or obviousness. However, for each proposed ground, at least one claim element is missing from the relied-upon reference or combination of references. Thus, Petitioners have failed to meet its initial burden to show that each element was known in the prior art.

I. OVERVIEW OF THE '843 PATENT

The '843 Patent is directed, among other things, to computer-implemented processes for automating a user's interaction between a first application, such as a word processing application or spreadsheet application, on the one hand, and a second application, such as contact management application having a database, on the other hand. In the '843 Patent, Exhibit 1001, Figs. 1 and 2 are flow charts showing for these interactions a number of scenarios, which are described from col. 4, line 25-col. 5, line 53. Further details of the interactions are provided in

discussion thereafter of the other figures of the '843 Patent, and the discussion includes references back to relevant portions of the flow charts in Figs. 1 and 2. Fig. 1 is reproduced below.

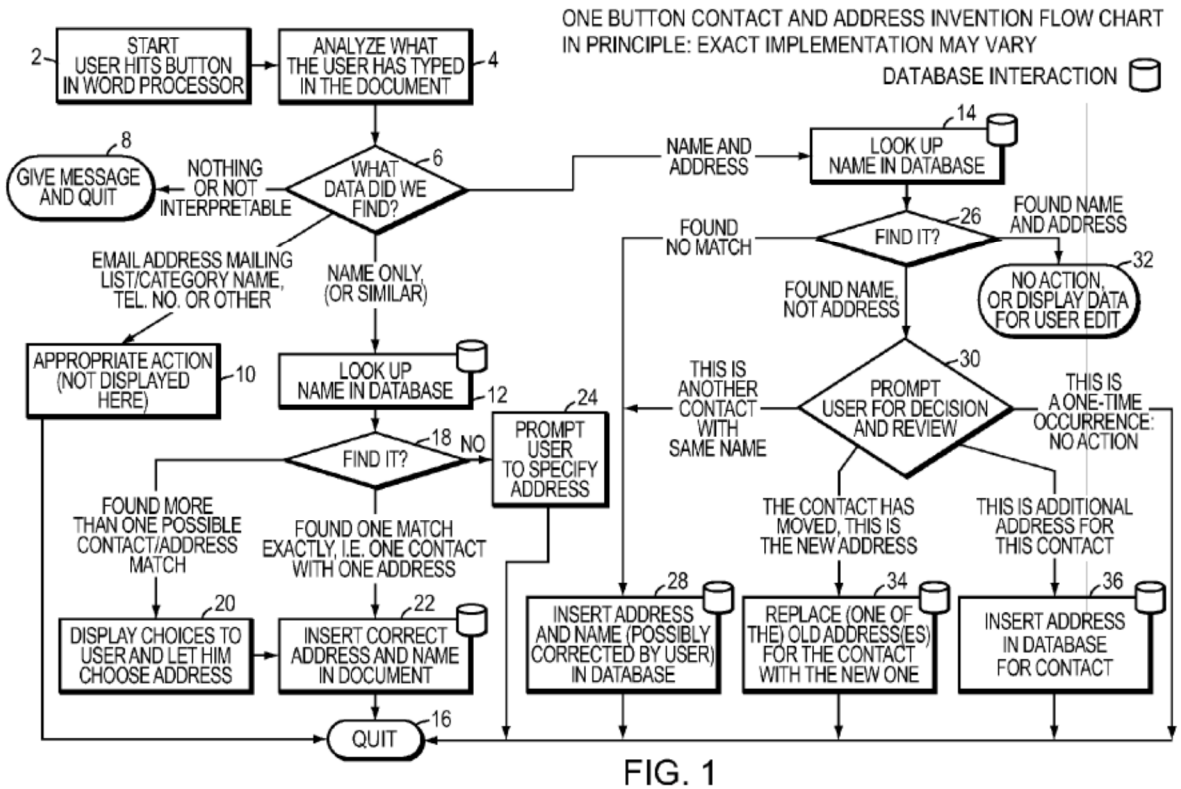


FIG. 1

The user interface of the first application includes a One Button 42 that the user can select to initiate the Patent's functions. See '843 Patent, Exhibit 1001, Fig. 1, step 2; Fig. 3. In various scenarios, after the user has clicked on the One Button 42, text in a document in the first application is analyzed (in step 2 of Fig. 1) to identify information, such as names, persons, companies, and addresses. *Id.*, col. 4, lines 32-39. The second application receives this information as a search term, which it uses to look up and retrieve related information from its database.

Id., Fig. 3, steps 12 and 14. The type of the latter information depends on the type of the former. For example, if the search term is a name, the second application may retrieve an address, related to the name, from the database. *Id*., col. 5, line 61-col. 6, line 3. Likewise, if the search term is a name of a mailing list, the second application may retrieve mailing or e-mail addresses for members of the group. *Id*., col. 4, lines 16-18.

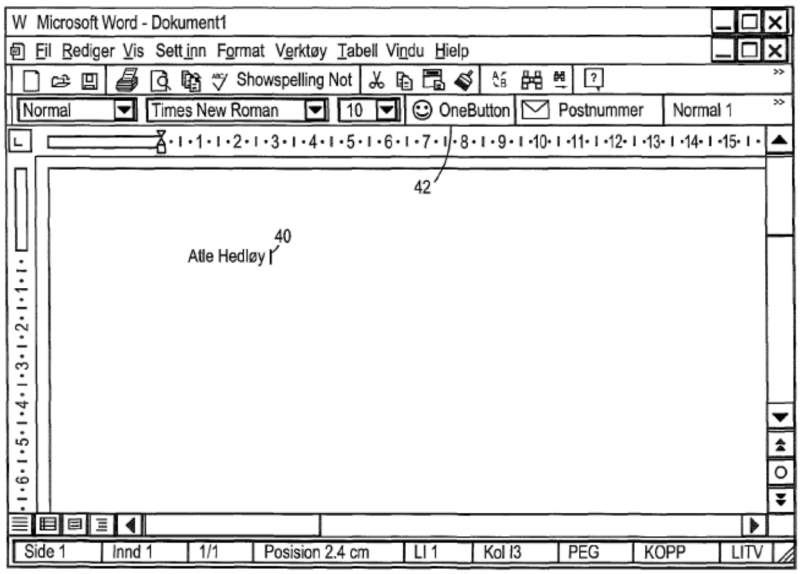


FIG. 3

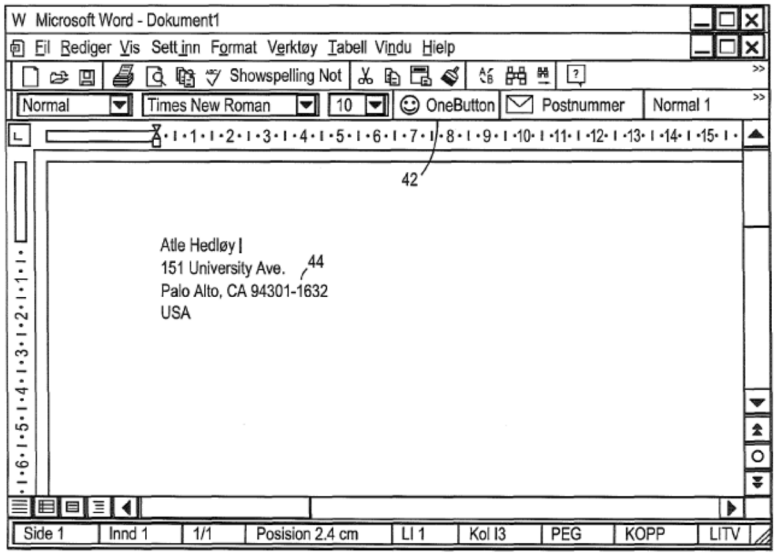


FIG. 4

Once the related information has been obtained from the database, a number of different scenarios can follow. In particular, the word processing application can either insert the related information into the document, or display the related information. Which

action the application performs depends on the type of information (e.g., name, name and address) identified in the document.

For example, if the identified information includes only a name, a search is initiated in the database associated with the second application for the name. *Id*., Fig. 1, steps 6 and 12. If only a single entry is found in the database for the name and the entry includes a single address, then the address is inserted into the document. *Id*., Fig. 1, steps 6, 12, 18, and 22; Fig. 4; col. 3, lines 63-67; col. 4, lines 43-54; col. 5, line 61-col. 6, line 5. Figs. 3 and 4 are reproduced above. Fig. 3 shows a document displayed in Microsoft Word when the document includes solely a name, "Atle Hedloy" 40. Fig. 4 shows the document after the address has been inserted.

In another example, if the identified information includes a name and an address, a search is initiated in the database associated with the second application for the name. *Id*., Fig. 1, steps 6 and 14. If an entry matching the name and address is found, both may be displayed for the user to edit. *Id*., Fig. 1, step 32; col. 4, lines 57-64. If the name happens to be in the contact database but the address in the contact database for that name differs from the address typed by the user into the document (per Fig. 1, step 26), then the user is prompted to make a choice (per Fig. 1, step 30). The user is presented with a screen shown in Fig. 9, which is reproduced below.

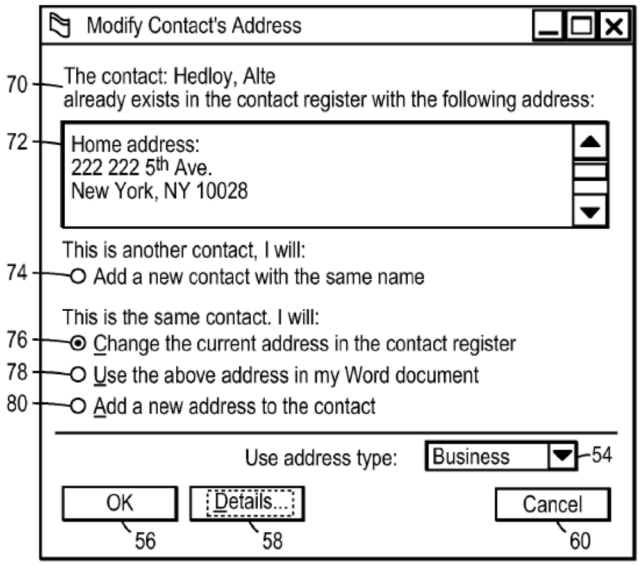


FIG. 9

Fig. 9 represents a screen presented to the user in which the user is given a series of choices that can be made in this specific context. *Id.*, col. 6, line 66-col. 7, line 14. The screen reproduces the name that is both in the document and in the contact database, and it also displays the address that is in the contact database for that name. Thus, the screen displays the name and address retrieved from the database that is related to the name and address from the document.

II. CLAIM CONSTRUCTION

In an *inter partes* review, the Patent Trial and Appeal Board gives patent claims their "broadest reasonable interpretation in light of the specification of the patent". 35 C.F.R. § 42.100(b); *Phillips v. AWH Corp.*, 415 F.3d 1303, 1316 (Fed. Cir. 2005) (*en banc*). The prosecution history is also relevant to identify the correct construction of claim terms. *Phillips v. AWH Corp.*, 415 F.3d at 1317. Extrinsic evidence may also be relevant to establish the meaning of terms, but such

evidence is only relevant to the extent it is consistent with the specification and file history. *Id.* at 1319.

Patent Owner Arendi proposes construction of certain claim terms below pursuant to the broadest reasonable interpretation consistent with the specification standard. The proposed claim constructions are offered for the sole purpose of this proceeding and thus do not necessarily reflect appropriate claim constructions to be used in litigation and other proceedings wherein a different claim construction standard applies.

A. "an input device, configured by the first computer program"

Independent claims 1, 20, 23, and 42 all recite the limitation "providing an input device, configured by the first computer program". Therefore, according to this limitation, a first computer program must "configure" the input device. Words of a claim must be given their plain meaning, which refers to the ordinary and customary meaning given to the words by one of ordinary skill in the art. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005) (*en banc*). Dictionary definitions may be used to determine the ordinary and customary meaning of words. *Ferguson Beauregard/Logic Controls v. Mega Systems*, 350 F.3d 1327, 1338 (Fed. Cir. 2003) (Dictionary definitions were used to determine the ordinary

and customary meaning of the words "normal" and "predetermine" to those skilled in the art.)

In this situation, we turn to the American Heritage College Dictionary 3rd edition 1997 for a definition of "configure". This dictionary defines "configure" as "to design, arrange, set up, or shape with a view to specific applications or uses". See Exhibit 2001. When this definition is applied to the claim limitations, the claims consequently require that the first computer program set up the input device so that it can be used. Therefore, "an input device, configured by the first computer program" should be construed as "an input device, set up by the first computer program for use".

Petitioners seek to interpret the limitation differently by ignoring a word in the claim. The independent claims of the subject patent require "providing an input device, configured by the first computer program" (emphasis added). The claim requires both "providing" and "configuring" and both words must be considered in evaluating the claim for obviousness. "All words in a claim must be considered in judging the patentability of that claim against the prior art". *In re Wilson*, 424 F.2d 1382, 1385 (CCPA 1970).

In violation of this principle, Petitioners seek to construe "providing an input device, configured by the first computer program" as "providing an interface to receive the user command". See Petition, page 7. In their proposed claim

construction, Petitioners have ignored the word "configuring" altogether by collapsing the separate requirements of "providing" and "configuring" into the single requirement of "providing". As a result, the Petitioners' proposed claim construction fails to account for each and every limitation of the claims in violation of the requirement that "[a]ll words in a claim must be considered in judging the patentability of that claim against the prior art". *In re Wilson*, *id*.

Therefore, the Patent Trial and Appeal Board should reject Petitioners' proposal and adopt Patent Owner's construction of "an input device, configured by the first computer program" as "an input device, set up by the first computer program for use".

III. OVERVIEW OF THE PRIOR ART

A. Overview of LiveDoc

LiveDoc concerns structure detection within a document where a "structure" represents meaningful bits of syntactically - regular information. LiveDoc allows a user to perform a function based upon an identified structure. To accomplish this goal LiveDoc constructs "a means of passing text from a user's document for matching against a collection of recognizers". Exhibit 1006, page 53. Thus, LiveDoc operates outside of any application program and outside of the document under the control of the application program.

The LiveDoc architecture is shown in Fig. 3 at page 56. As can be seen from the labels in the right-hand column in Fig. 3, the Applications (such as word processing) are shown separately from the LiveDoc Manager and from the Analyzer server. Further, the LiveDoc manager communicates with an external application (i.e. a text editor) using API callbacks. *Id*. at 57.

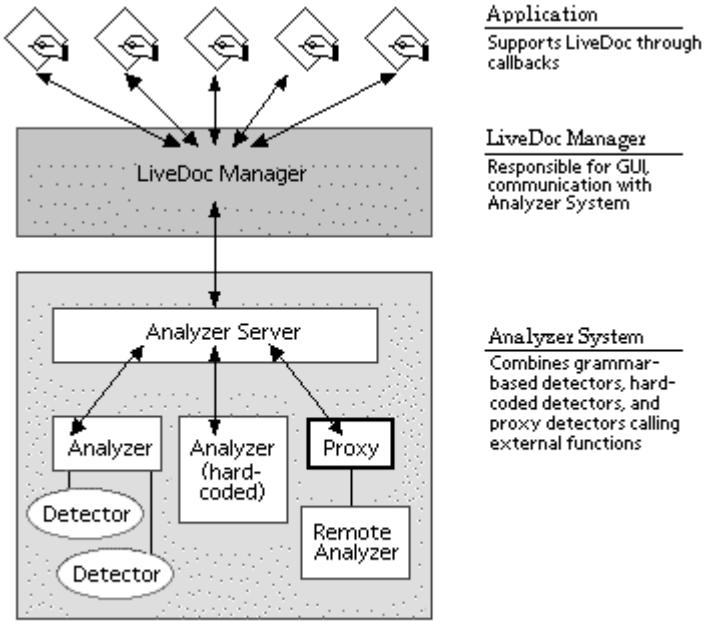


Figure 3: The high-level LiveDoc architecture.

The LiveDoc application receives only the text from the text editor (application in Fig. 3) and analyzes the text independently of the actual document in the text editor using a set of detectors under the control of an analyzer server.

If one is viewing a document in a word processing program on a computer that is running LiveDoc, the structures identified by LiveDoc are not visible in the word processing program itself. In order for the discovered structures to be visible to a user, the user must enter "LiveDoc mode" by pressing and holding the function key, causing the LiveDoc Manager to update "the display to present the highlight information over the discovered structures". *Id*. at 56. The user can then

use the mouse to move over a highlighted item and press the mouse button that causes the LiveDoc Manager to present a menu of functions associated with the highlighted item.

> LiveDoc knows where these structures appear in the text passed to it- an e-mail address might appear in characters 150 through 162 of the window's contents - but it has no idea where in the window those characters physically appear, and, thus, where the highlights should appear: this is information held by the application, not by LiveDoc. Hence, LiveDoc must ask the application for the information about the structures it has found via a callback. Once this information is available, the highlights and their associated mouse-sensitive regions can be constructed.

Id. The overlaid highlights are independent and separate from the text editor and the document. Fig. 2 shown below show some of the actions that LiveDoc allows for a recognized structure.

Each of the functions shown involves using the recognized text with an external application.

"Our initial implementation of LiveDoc as LiveSimpleText assumed that actions would be handled by external applications, such as a Web browser presenting the page pointed to by a URL[:]" *Id.* at 57.

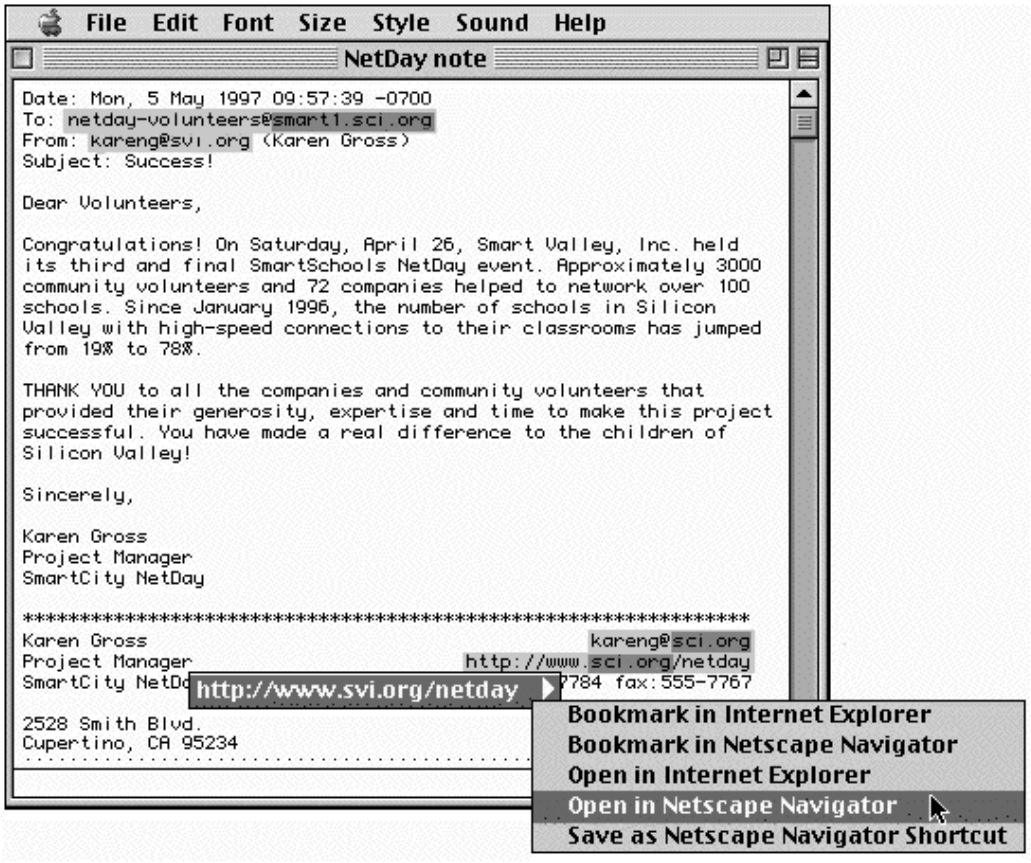
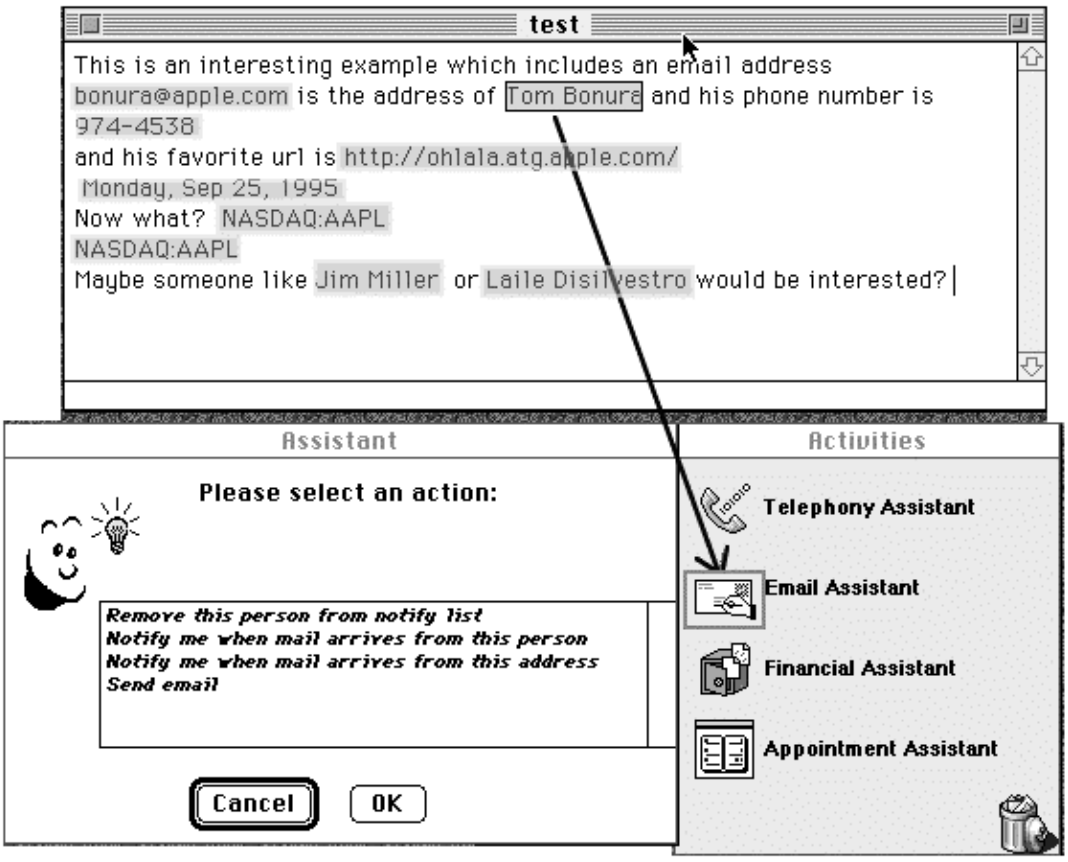


Figure 2: A sample interaction with LiveDoc. Note the highlighting of the discovered structures, the menu of actions available or the selected structure, and the nested highlighting of nested structures.

B. **Overview of Drop Zones**

Drop Zones extends on LiveDoc where a user that has entered LiveDoc mode may be presented with an interface that interprets the meaning of the identified and selected structure and presents recommended appropriate actions. Operation of the Drop Zones system uses Live Doc windows, as shown in Figures 1 and 2 of Drop Zones. The caption for Figure 1 states that "Drop zone is shown in

the window labeled ‘Activities’. The window at the top called ‘Test’ is a LiveDoc window showing proper names, e-mail addresses, phone number, URL, date and stock market ticker codes”. *Id*. at 60. These identified “structures” are shown in the LiveDoc window as highlighted. *Id*. Similarly in Figure 2 (reproduced below), which illustrates “A user interaction with Drop Zones”, the same LiveDoc window is displayed. *Id.* To use Drop Zones, the user must first enter “LiveDoc mode” by pressing and holding a function key in order to cause highlighting to be displayed



over the document. Then, as discussed in connection with Figure 2, the user must select a structure in a LiveDoc window. *Id.*

In the LiveDoc window (identified as the window "test" in Figure 2), the user uses the mouse to select an item of information that has been highlighted (here the name Tom Bonura) and (while still holding down the mouse button), then drags the selected item to the window labeled "Activities" over a desired category (here "Email Assistant") and then drops the selected name on the category (by releasing the mouse button). Dropping the item causes a menu of actions to appear in the Assistant window (shown to the left of the Activities window in Figure 2), and from that menu, the mouse is used to select a desired action. *Id*. at 60-61.

C. Overview of Miller

Miller discloses systems and methods for "detecting structures in data and performing actions on detected structures" (claim 1). To achieve this function, Miller uses a computer program 165 that works outside of a document, such as a word processor document 210. Exhibit 1007, Fig. 2. The program 165 is initiated in response to user selection of a detect structures button 520: "Window 510 includes a [detect structures] button 520 for initiating program 165, although alternative mechanisms such as depressing the "option" key may be used. Upon initiation of program 165, system 100 transmits the contents of document 210 to analyzer server 220 [of the program 165]". *Id*., col. 5, lines 22-28.

This analyzer server 220 "receives data having recognizable patterns from a document 210". *Id*., Abstract, col. 3, lines 57-58. Then, Miller uses "pattern

analysis units, such as a parser and grammars or a fast string search function and dictionaries" to parse the data "for recognizable structures". *Id*., col. 3, lines 57-64. Therefore, Miller assesses the text of the document to determine if it contains any grammars or strings from the libraries: "[A]ssuming program 165 initiates with the receipt of any text, the received content or portion is scanned 820 for identifiable structures using the patterns in analyzer server 220". *Id*., col. 5, lines 56-59. Then, "[U]pon detection of a structure, analyzer server 220 links actions associated with the responsible pattern to the detected structure, using conventional pointers". *Id*.

After the structures are detected, an application program interface 230 within the program 165 subsequently "communicates with application 167 to obtain information on the identified structures so that user interface 240 can successfully present and enable selection of the actions". *Id*., col. 4, lines 2-5. Miller's user interface (240) "highlights the detected structures". *Id*., col. 4, line 10; col. 5, lines 35-37.

D. Overview of Luciw

Luciw describes logical processes, usable by a pen-based computer system that functions as a personal organizer, to provide "implicit or explicit assistance" for "user supportive information functions". Exhibit 1008, col. 4, lines 14-18 (pen-based computer system); col. 2, lines 16-19 (implicit or explicit assistance).

The pen-based computer system has a database that can be queried. *Id.*, col. 8, lines 31-34. Luciw describes "implicit" assistance, wherein a user has used a smart field to enter a word used for look up in the database or has otherwise similarly triggered a database lookup, and "explicit" assistance, wherein the user explicitly invokes assistance from the device as by using pen 38 of Fig. 2. *Id*., col. 8, lines 11-62.

The logical processes used by the Luciw device for providing implicit and explicit assistance are shown in Fig. 3 of Luciw. *Id.*, col. 8, lines 2-6. A review of Fig. 3 shows that the database is queried in step 106 if it is determined in step 104 that there is an implicit assist. On the other hand, if in step 104 it is determined that there is not an implicit assist, and if further it is determined that there is an explicit assist, there is no database query, because the only database query indicated is in step 106, exclusively where there is an implicit assist.

As an example of implicit assist, Luciw provides Figs. 4b, 4c, 5, 6a and 6b, which describe use of a "smart field". *Id.*, col. 10, line 23 et seq. (beginning discussion of smart fields in connection with Fig. 4b). According to Luciw, "[a] smart field is considered to be a predefined region on screen 52 of computer system 10 shown in FIG. 2, or a predefined region within a window which appears on screen 52". *Id.,* col. 8, lines 16-19. Fig. 4b is reproduced below.

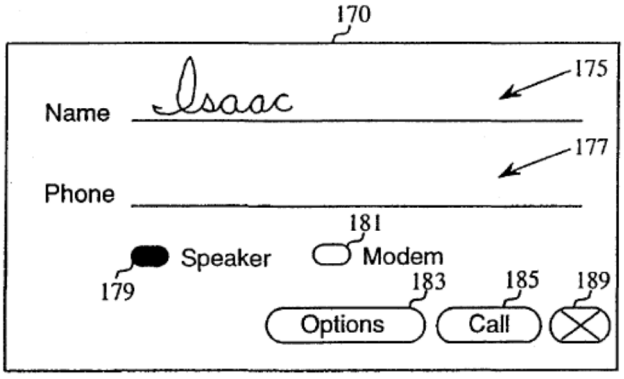


Figure 4b

According to Luciw, Fig. 4b "shows a phone slip window 170 with a smart name field 175 which has for example been evoked by either highlighting the verb 'call' or by simply writing the word on the display surface either before or after establishment of window 170". *Id.*, col. 10, lines 24-28. Operation of the phone slip window is explained in the lines thereafter in Luciw:

> Once the particular window 170 is presented to the user, the name ISAAC can be handwritten into the particular smart field 175. The assistance process recognizes the handwritten name "Isaac", and either continues operation as suggested at step 106 in FIG. 3 directly, or concurrently displays the recognized name in formal font form, as suggested in FIG. 4c, in the same position of the smart field, where formerly the handwritten name "Isaac" had been established. As will readily be recognized, window 170 in FIG. 4b may contain several smart fields, in this case for example definable for either the "name" field 175 or a "phone" field shown at step 177.

Id., col. 10, lines 27-39.

Because the user of the Luciw device uses the smart field to specify the field for which a database search is desired—a name in the name field 175 or a phone number in the phone field 177—the Luciw device uses the entered item to search

for in the database for an item that has the same value for a corresponding attribute. *Id.*, col. 10, line 51- col. 12, line 11.

E. Overview of Pandit

Pandit describes a program that enables users to identify text of interest and select an operation applicable to the text. Pandit identifies classes of text in a document and enables a user to select programs, based on the identified classes, applicable to the text. When a document is open in the program, the program provides a menu bar 13 that displays classes of text, such as "Date", "EMail", and "Phone #". Exhibit 1009, Figs. 1a-1f. The user selects text in the document by shading, underlining, or pointing and clicking on the text. *Id*., col. 2, lines 4-8. The program identifies the class of the selected text and highlights that class in the menu bar 13 using boldface type. *Id*., col. 2, lines 8-16, 51-53, 64-66 and Figs. 1a, 1c, and 1e. The boldface type indicates that the programs for that class of text have been enabled. *Id*., col. 2, lines 11-12.

When the user selects the bolded class, the program displays the programs for the class. *Id*., col. 2, lines 15-18, 20-21, 33-35. For example, if a user selects the highlighted option "Date" from the menu bar 13, the program displays potential programs that display a calendar or create an appointment based on the selected date in the document. *Id*., Fig. 1b. If a user selects the highlighted option "Email"

from the menu bar 13, the program displays potential programs that create an email message addressed to the selected email address or add the address to an address book. *Id*., Fig. 1d. If a user selects the highlighted option "Phone #" from the menu bar 13, the program displays potential programs of dialing the selected phone number, adding the phone number to an address book, or preparing a fax to be sent to the phone number. *Id*., Fig. 1f. The user selects a program to be performed by clicking on the operation or executing one or more keyboard strokes. *Id*., col. 2, lines 41-46.

IV. SINCE THE PRIOR ART DOES NOT ANTICIPATE OR RENDER ANY CLAIM OBVIOUS, NO *INTER PARTES* REVIEW SHOULD BE INITIATED

A. Overview of Reasons for Denying Inter Parties Review

Petitioners have failed to show any prior art alone or in combination to address all of the limitations of any of the independent claims. Because LiveDoc and Drop Zones describe a text editor that displays a document and a LiveDoc Manager that configures highlighting, LiveDoc and Drop Zones fail to disclose or suggest that the same "first computer program" performs both "displaying the document electronically" and "providing an input device, configured by the first computer program", and therefore Ground 1 fails to establish a *prima facie* case of obviousness.

Because, the LiveDoc Manager, and not the text editor, receives the user's selection of highlighting, LiveDoc and Drop Zones fail to disclose or suggest the claim limitation of "receipt by the first computer program of the user command from the input device" – another defect of Ground 1.

Because Miller fails to disclose how the "Detect Structures" button is configured, Miller fails to disclose or suggest "providing an input device, configured by the first computer program" – a defect in Ground 2.

Because Miller's Program 165, and not the Application 167, receives the user's selection of the "detect structures" button, Miller fails to disclose or suggest the claim limitation of "receipt by the first computer program of the user command from the input device" – another defect in Ground 2.

Because Miller searches within the document for strings or grammars, Miller fails to disclose or suggest "performing a search using at least part of the first information as a search term … in an information source external to the document" – another defect in Ground 2.

Because the user informs the Luciw apparatus of the input's type of information, Luciw fails to disclose "analyzing, in a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information" – a defect in Ground 3.

Because Pandit's nouns and verbs are not the "types of information" contemplated by the claims, and, at best, the type of second information is decided by the user, and not dependent at least in part on the type or types of first information, Pandit fails to disclose or suggest "performing a search … wherein the specific type or types of second information [found] is dependent at least in part on the type or types of the first information [used as the search term]"—a defect in Ground 4.

Because Pandit's does not disclose searching in the address book, Pandit fails to disclose or suggest "performing a search using at least part of the first information as a search term in order to find the second information" and "causing a search for the search term" – another defect in Ground 4.

B. Because LiveDoc and Drop Zones describe a text editor that displays a document and a LiveDoc Manager that configures highlighting, LiveDoc and Drop Zones fail to disclose or suggest that the same "first computer program" performs both "displaying the document electronically" and "providing an input device, configured by the first computer program", and therefore Ground 1 fails to establish a *prima facie* case of obviousness.

Independent claim 1 requires the "first computer program" to perform at least two tasks. First, the "first computer program" must display a document electronically: "displaying the document electronically using the first computer program" (first subparagraph of claim 1). Second, the "first computer program" must configure an input device: "providing an input device, configured by the first

computer program" (fourth subparagraph of claim 1). Therefore, the same computer program must perform both of these tasks. However, in LiveDoc and Drop Zones, different computer programs perform these two tasks. In particular, the text editor displays a document electronically and the LiveDoc Manager configures the input device. Therefore, LiveDoc and Drop Zones fail to disclose or suggest a "first computer program" that performs "displaying the document electronically" and "providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation", as required by independent claim 1. Indeed, this limitation is found in all the independent claims and hence is required by all of the claims challenged in Ground 1.

Turning now to Petitioners' arguments, Patent Owner first agrees that the text editor displays the document electronically. Then, with respect to the "input device" limitation, Petitioners have equated the highlighting positioned over detected structures with this limitation: "Configured by the first computer program - LiveDoc/Drop Zones knows where to place the selectable highlights because the first application [the text entry application program] tells it where the structures are located in the document (i.e., the input device is configured by the first computer program)". See Petition, page 13.

Claim 1 requires that the input device, namely the highlighting, be "configured by the first computer program". As discussed in Section II(A), for the text editor to "configure" the highlighting, the text editor must set up the highlighting for use. We will demonstrate herein that setting up the highlighting for use is the exclusive domain of the LiveDoc Manager. The LiveDoc Manager alone analyzes the text in a document, identifies the characters in the text that corresponds to structures, obtains the physical locations within a display for those particular characters, and applies highlighting to those locations.

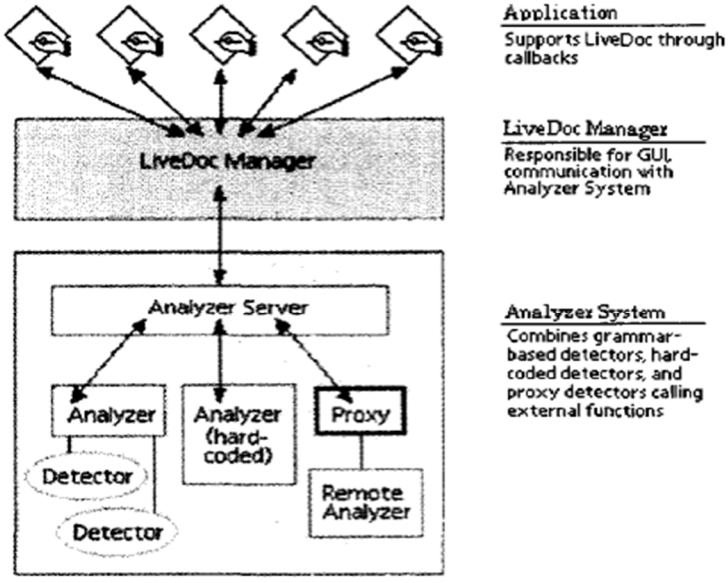


Figure 3: The high-level LiveDoc architecture

For some of these steps, the LiveDoc Manager does send calls to the text editor to obtain needed information. However, as described in Fig. 3, the LiveDoc Manager and applications (*e.g.*, text editor) are separate programs that are being executed independently:

As an independently executing program, the LiveDoc Manager unilaterally determines the information needed to set up the highlighting and requests this information from the text editor. The text editor merely responds by giving the LiveDoc Manager the screen positions of the structures to be highlighted. Since the text editor, has only a passive role in the LiveDoc Manager's preparation of the highlighting, and in fact is provided with no data whatsoever relating to the highlighting, the LiveDoc Manager implements the highlighting to "configure" the input device.

In more detail, when the user views the document, the user is in the text editor. To invoke LiveDoc, the user must exit the text editor and access the LiveDoc Manager. To do so, the user presses and holds down a function key: "Holding down a function key places the document in `LiveDoc mode' and presents the highlighted structures; releasing the function key returns the document to normal". Exhibit 1006, page 55.

"The LiveDoc Manager also controls the events that occur when the user presses the function key to enter LiveDoc mode, and when the mouse button is pressed while over a LiveDoc item. The LiveDoc Manager updates the display to present the highlight information over the discovered structures when the function key is pressed, and to remove the highlights when the function key is released. The LiveDoc Manager also receives the notification that the mouse button has been

pressed over a highlighted item; it then gets the list of actions appropriate to the selected item and presents a menu of them to the user". *Id*. at 56. There can be no mistake: the Petitioners' statement that the text editor configures the highlights (input device), and receives the user command simply has no basis in fact.

To begin, the LiveDoc Manager asks the text editor for a copy of the text currently visible in its window, the text editor sends the text to the LiveDoc Manager, and the LiveDoc Manager analyzes this text: "The receipt of these calls by the LiveDoc Manager signals the Analyzer Server to analyze the text provided by the calling application; this will typically be the text currently visible in the applications' front-most window". *Id*. Based on this analysis, the LiveDoc Manager identifies structures such as e-mail addresses. *Id*.

At this point, the LiveDoc Manager only knows which characters in the string of characters from the text editor correspond to structures: "LiveDoc knows where these structures appear in the text passed to it – an e-mail address might appear in characters 150 through 162 of the window's contents – but it has no idea where in the window those characters physically appear[.]" *Id*. Thus, the LiveDoc Manager does not know where the characters of interest physically appear in the window, and by extension, where the LiveDoc Manager should apply highlighting. To obtain this information, the LiveDoc Manager sends a call to the text editor requesting the physical locations of characters of interest: "LiveDoc must ask the

application for the information about the structures it has found via a callback". *Id*. Of course, there would be no need for the LiveDoc Manager to ask for this information if the text editor were to set up the highlight and receive the user command. Upon attaining this information, the LiveDoc Manager can apply the highlighting in the applicable locations: "Once this information is available, the highlights and their associated mouse-sensitive regions can be constructed". *Id*. This is done by "adding the notion of a sometimes-visible layer to the front of the display". *Id*. at 58. As demonstrated by these excerpts, the LiveDoc Manager sets up the highlighting for display and subsequent use and not the text editor.

Petitioners concede that the LiveDoc Manager asks the text editor for this information, and the text editor supplies the same using a callback: "See, e.g., LiveDoc at 56 ("LiveDoc knows where these structures appear in the text passed to it...but it has no idea where in the window those characters physically appear, and, thus, where the highlights should appear: this is information held by the [text editor] application, not by LiveDoc. Hence, LiveDoc must ask the application for the information about the structures it has found via callback". See Petition, pages 13-14.

In this situation, the LiveDoc Manager has not informed the text editor that the characters correspond to a structure, nor has the LiveDoc Manager indicated that the displayed characters shall be highlighted to be identified as a structure to a

user. Thus, the text editor does not know the significance of the identified characters or the intended use of the physical locations that it sends to the LiveDoc Manager. In fact, the text editor does not even know that highlighting exists or will exist, and the LiveDoc and Drop Zones references do not give any evidence that the text editor even knows that it exists when it exists, and indeed, *the text editor has no need for this information in the LiveDoc or Drop Zones systems*. The text editor just processes a request for information, which the LiveDoc Manager unilaterally uses to prepare highlighting for display.

Therefore, as just demonstrated, it is the LiveDoc Manager that sets up the highlighting to be used, whereas the text editor electronically displays the document. In this manner, LiveDoc describes a different program performing each of these two activities. Therefore, LiveDoc fails to disclose or suggest a "first computer program" used to configure the input device, as required by claim 1.

Further, even if Petitioners' proposed construction for the "input device, configured by the first computer program" were adopted, LiveDoc would still fail to disclose this limitation. Under Petitioners' proposal, configuring the input device would be construed as "providing an interface to receive the user command". See Petition, page 7. Since Petitioners have equated the text editor with the first computer program, when Petitioners' proposed construction is

applied to LiveDoc, the text editor must provide the highlighting to receive the user command.

However, as we have already demonstrated, the LiveDoc Manager and not the text editor provides the highlighting: "LiveDoc Manager constructs the various highlights for the discovered structures and their corresponding menu of actions". Exhibit 1006, page 56. Further, even Petitioners admit the LiveDoc Manager performs this step: "LiveDoc/Drop Zones highlights detected information". See Petition, page 13.[1] Regardless of the manner in which "an input device, configured by the first computer program" is construed, the LiveDoc Manager performs the configuring, not the text editor. Therefore, LiveDoc and Drop Zones fail to disclose or suggest a "first computer program" used to display the document

[1] Citing their expert, Menascé, Petitioners argue that "it would have been obvious for LiveDoc to contact the word processor via callback and inform it of the position of the detected structures within text, such that the word processor would then construct the highlights (input device) by mapping positions in text to positions in the visible window. See Petition, page 14. This hindsight-driven argument is inconsistent with placement, in the LiveDoc Manager, of the functionality of identifying structures and offering actions associated with the identified structures. Moreover, shifting the highlighting function does not change the fact that LiveDoc performs the configuring, not the text editor.

electronically and to configure the input device, and Ground 1 fails to make a *prima facie* case that claims 1-44 would have been obvious.

C. Because the LiveDoc Manager, and not the text editor, receives the user's selection of highlighting, LiveDoc and Drop Zones fail to disclose or suggest the claim limitation of "receipt by the first computer program of the user command from the input device", and therefore for this additional reason Ground 1 fails to establish a *prima facie* case of obviousness.

Independent claim 1 requires "receipt by the first computer program of the user command from the input device". This limitation of the claim explicitly requires the first computer program to receive the user command from the input device. We will demonstrate that in contrast to the claim, in LiveDoc and Drop Zones, the LiveDoc Manager and not the text editor receives the user command. Since the incorrect entity receives the user command, LiveDoc and Drop Zones fail to disclose or suggest "receipt by the first computer program of the user command from the input device", as required by claim 1. Indeed, this limitation is found in all the independent claims and hence is required by all of the claims challenged in Ground 1.

As discussed above in Section IV(B), Petitioners and Patent Owner agree that the text editor is the "first computer program", and Petitioners have further equated the displayed highlighting with the "input device". Further, in Petitioners'

discussion of section 1e of claim 1, which includes the limitation of the "user command", Petitioners equate this limitation with the user selection of the highlighting: "When the user selects a highlighted structure (an input device) the system determines the related actions that can be performed (initiates an operation). See, e.g., Drop Zones at 60 ("When an object is selected, it is sent to the Drop Zone control system…)". See Petition, page 13. The Petitioners do the same regarding section 1h, which includes the claim limitations of interest: "As discussed in claim le, when a user selects a highlighted structure the system determines the related actions that can be performed". See Petition, page 15.

Petitioners fail to point to anything in LiveDoc and Drop Zones disclosing or suggesting that the purported first computer program, the text editor, receives the user selection of highlighting. When the Petitioners apply LiveDoc and Drop Zones to section 1h, their analysis fails to address how the user selection is received. *Id*. Further, nowhere do the Petitioners even mention the text editor. *Id.* Petitioners' analysis of section 1h ignores the first clause, "in consequence of receipt of the first computer program of the user command from the input device", and focuses exclusively on the second clause, "causing a search: "As discussed in claim le, when a user selects a highlighted structure the system determines the related actions that can be performed. This determination is made by performing the search discussed in claim lf - *e.g.*, searching an address book (information

source) using an address book application (second computer program) to find the email address associated with an identified name. See, e.g., Drop Zones at 61 (`When objects are selected, they are inspected by the assistants in the Drop Zone. These assistants are built around a collection of facts and axioms that determine whether and how they can operate in some meaningful way on various kinds of objects.'). See also claims le and 1f". See Petition, page 15.

In fact, the LiveDoc Manager, not the text editor, receives the user selection of highlighting. LiveDoc explicitly states that the LiveDoc Manager receives the selection of a highlighted item: "The LiveDoc Manager also controls the events that occur…when the mouse button is pressed while over a LiveDoc item. The LiveDoc Manager also receives the notification that the mouse button has been pressed over a highlighted item". Exhibit 1006, page 56.

Additionally, in LiveDoc, the section titled "LiveDoc: Beyond Data Detectors" describes how the LiveDoc Manager functions; the last paragraph of this section states "What is described above is, of course, only a general design for LiveDoc". *Id*. at 55. In this section, LiveDoc describes how "[p]ointing at a highlight and pressing a mouse button then displays the menu of actions that can be applied to the structure, as shown in Fig. 2". *Id*. LiveDoc thus teaches that displaying menus of action in response to user selection of a highlighted item is part of the LiveDoc Manager's design and consequently, they must be features

within the LiveDoc Manager itself. Since the LiveDoc Manager performs an action (i.e., displays a menu) in response to a user action (i.e., selecting a highlighted item), the LiveDoc Manager necessarily receives the user action itself.

Likewise, with respect to Drop Zones, Drop Zones teaches that the LiveDoc system and not the text editor receives the user selection of highlighting. The Petitioners selectively quote Drop Zones: "See, e.g., Drop Zones at 61 ('When objects are selected, they are inspected by the assistants in the Drop Zone. These assistants are built around a collection of facts and axioms that determine whether and how they can operate in some meaningful way on various kinds of objects')", seemingly implying that the Drop Zones assistants receive the user selection of the highlighted structure. However, a further inspection of the Drop Zones reference reveals that "An interaction with the Drop Zone interface is shown in Figures 1 and 2. The window named 'test' in Figure 1 belongs to a LiveDoc-enabled word processor, LiveSimpleText (see [6]), and shows a number of structures within the document in view having been recognized by the analyzers". Exhibit 1006, page 60. "Consider Figure 2, in which the user has selected the structure Tom Bonura, which LiveDoc has identified with its personalName recognizer. When an object is selected, it is sent to the Drop Zone control system". *Id.* Clearly then, it is the LiveDoc system functionality that receives the user selection of the highlighted structure, and only then it sends it to the Drop Zone control system. Therefore, the

user command, defined by the Petitioners as the user selection of the highlighted structure, is not received by the first application, i.e. the text editor, but by the LiveDoc system.

Furthermore, in Drop Zones, selecting an object (structure), does not initiate an operation as required by the claim, but simply, as admitted by the Petitioners, sends the object "to the Drop Zone control system. Each of the assistants determines if it is able to accept and act upon the set of currently selected objects". *Id*. Only after the user has selected an assistant and an action is the operation initiated. *Id*. ("These assistants make their capabilities visible when the user selects various structures identified by LiveDoc and drags them to the assistants."). Thus, the Petitioners' analysis of the user command for the Drop Zones reference, furthermore fails because the user command does not initiate an operation as required by the claim, but simply, as admitted by the Petitioners, sends the object "to the Drop Zone control system. Each of the assistants determines if it is able to accept and act upon the set of currently selected objects". *Id*.

Nor can it be argued that it is the Drop Zones "assistants" that receive the user command as required by the claim. Drop Zones states that the "assistants" that receive the dropped items are all part of Drop Zones: "[t]he window labeled *Activities* is a Drop Zone interface to a set of interpreters or `assistants.' Each of these assistants, *E-mail*, *Telephony*, *Finance* and *Appointment*, implements a

knowledge base that can operate on appropriate sets of LiveDoc structures" (emphasis added). *Id*. Therefore, when the user drops a selected highlighted item on a Drop Zone "assistant", Drop Zones receives the command to process the highlighted item with the "assistant". Therefore, Drop Zones receives the user command.

For at least the forgoing reasons, the LiveDoc Manager in LiveDoc receives the user command, and the LiveDoc system of Drop Zones in the Drop Zones reference receives its respective user command. Neither reference describes the text editor, which Petitioners have equated with the "first computer program", as receiving the user command from the input device. As a result, both LiveDoc and Drop Zones fail to disclose or suggest "receipt by the first computer program of the user command from the input device", as required by independent claim 1 and therefore for this additional reason Ground 1 fails to make a *prima facie* case of obviousness.

D. Because Miller fails to disclose how the "Detect Structures" button is configured, Miller fails to disclose or suggest "providing an input device, configured by the first computer program", and therefore Ground 2 fails to establish a *prima facie* case of obviousness.

As discussed above, all of the claims require "providing an input device, configured by the first computer program". This claim limitation requires the first

computer program to configure the input device. However, Miller is silent regarding the manner in which the input device is configured. Therefore, Miller fails to describe the first computer program as configuring the input device. Petitioners effectively concede this deficiency in Miller, arguing that it would have been obvious for the first computer program in Miller to configure the input device. Petitioners rely on their expert's testimony for this point, but this testimony regarding the state of the art contradicts Miller's teachings. Therefore, the testimony of Petitioners' expert should be disregarded. As a result, Miller fails to disclose or suggest "providing an input device, configured by the first computer program", as required by the claims.

Turning now to Petitioners' arguments, Petitioners equate the application 167, a word processor, with the "first computer program" and the detect structures button 520 with the "input device". When applying Miller to section 1b of claim 1, "displaying the document electronically using the first computer program", Petitioners note that the application 167 performs the displaying: "Documents are displayed using a first computer program, such as a word processor (application 167 in Fig. 1)". See Petition, page 25. When applying Miller to section 1e of claim 1, "providing an input device, configured by the first computer program", Petitioners note that the detect structures button 520 receives user input: "The `detect structures' button 520 in Fig. 5 is an input device that allows the user to

enter a command to initiate the parsing operation. See, e.g., 5:22-37[.]" See Petition, page 26. Therefore, under Petitioners' interpretation of Miller, for Miller to fulfill the requirements of the claim limitation, the application 167 must configure the detect structures button 520.

The Petition fails to explain how the word processor in Miller configures the detect structures button 520. Nowhere does the Petition cite to passages in Miller that support this position. In fact, in Petitioners' analysis of "providing an input device, configured by the first computer program", Petitioners do not even mention configuration. Instead, their analysis for section 1e merely identifies the detect structures button 520 as the "input device:" "The `detect structures' button 520 in Fig. 5 is an input device that allows the user to enter a command to initiate the parsing operation". See Petition, page 26. Instead of relying on Miller's disclosures, Petitioners turn to their expert to discuss how configuration purportedly occurs. See Petition, pages 26-27. In this manner, Petitioners concede that Miller does not disclose "providing an input device, configured by the first computer program", as required by claim 1.

Experts may testify on what a reference implicitly describes, namely, what one of ordinary skill in the art would understand from the teachings of the reference. However, these "implicit teachings" must be consistent with the reference's explicit ones. If the reference explicitly teaches certain features, an

expert cannot credibly suggest that one of ordinary skill in the art would understand the reference to teach or contemplate contradictory features. Nevertheless, that is precisely what the expert for Petitioners does.

In his testimony, Petitioners' expert claims that "[I]t would have been obvious for the word processor program 167 to provide an interface, such as button 520, to receive a user command". Menascé Decl. 1, ¶71. "[I]t was well known to configure word processing programs to add GUI elements, such as additional menu options or button, to provide desired functionality". See Petition, pages 26-27. Here, Petitioners' expert emphasizes that configuring "word processing programs" (i.e., the application 167, or "first computer program") was well known. However, that is not what the claim limitation requires. The claim recites an "input device, configured by the first computer program". By focusing on configuration of the word processing program and not the detect structures button 520, the expert has directed his attention to the incorrect configuration process.

Regardless, the expert's testimony is still inconsistent with Miller's teaching. Although the expert has claimed that adding GUI elements to word processors was well known, applying this practice to Miller relies on a critical assumption: Miller must contemplate integrating the program 165 and the application 167. The expert envisions an embodiment in which the application 167 has been configured to

incorporate the program 165 so that it can offer the functionality of program165 as a feature.

This assumption is not supported by Miller. Nothing in Miller suggests that the program 165 is or could be integrated into the word processor 167. In fact, Miller teaches the opposite. Miller teaches that the program 165 and word processing application 167 are separate programs that execute simultaneously: "the program may be executed *during the run-time of another program*, i.e. the application which presents the document, such as Microsoft Word" (emphasis added). Exhibit 1007, col. 2, lines 42-44. Because the programs are separate, the program 165 and application 167 must communicate through the application program interface of program 165: "Since the program may be executed during the run-time of another program, *i.e.* the application which presents the document, such as Microsoft Word, an application program interface provides mechanisms for interprogram communications" (emphasis added). *Id.*, col. 2, lines 42-49.

Miller provides a few examples of how these separate programs communicate through the application program interface 230. For example, the program 165 uses this interface 230 to obtain information from the application 167 about the structures: "[A]fter identifying structures and linking actions, application program interface 230 [of program 165] communicates with application 167 to obtain information on the identified structures so that user interface 240 can

successfully present and enable selection of the actions". *Id*., col. 4, lines 1-5. In another example, the program 165 obtains a user's interactions with highlighted structures from the application 167: "User interface 240 communicates with application 167 through application program interface 230 to determine if a user has performed a mouse-down operation in a particular mouse-sensitive presentation region, thereby selecting the structure presented at those coordinates". *Id*., col. 4, lines 22-27. In this manner, instead of being integrated together, the program 165 and application 167 interact through a designated interface.

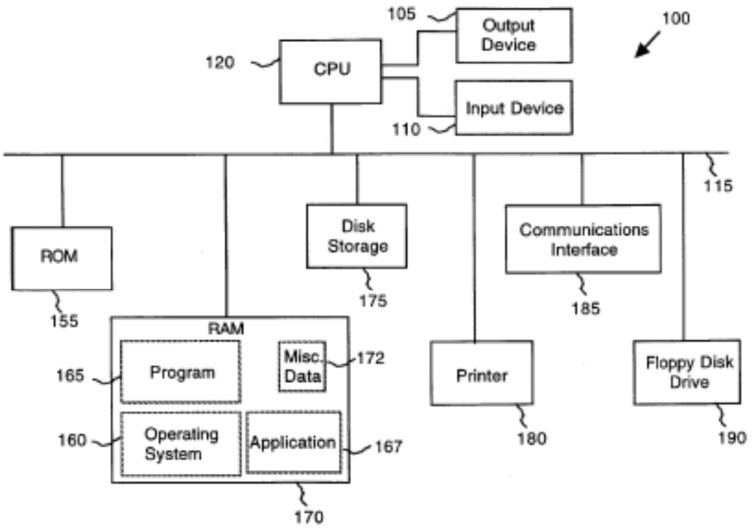


FIG. 1

Additionally, Miller consistently refers to the program 165 and application 167 as distinct entities. For example, in Fig. 1, Miller depicts the program 165 and application 167 as stored separately in random access memory (RAM) 170. Thus, Fig. 1 presents the program 165 and application 167 as discrete entities residing in different portions of the RAM 170. If the program 165 been integrated into the application 167, the program 165 would have appeared within the application 170 as a sub-entity.

In summary, Miller describes the program 165 and application 167 as separately executing entities, capable of communicating with one another via the program's 165 application program interface 230. Further, Miller describes the separate storage in memory of the program 165 and application 167. For at least the forgoing reasons, Miller's teachings indicate that the program 165 and application 167 are separate and distinct, not integrated together. Additionally, nowhere does Miller suggest that such integration is possible or advantageous. Rather, the motivation for such integration is provided entirely by the Petitioners' expert.

Since Miller presents the program 165 and application 167 as separate programs and is silent regarding benefits of their integration, Miller opposes the Petitioners' expert testimony contending that integrating structure detection into the word processor as an additional feature would have been obvious. In light of Miller's contradictions of the expert testimony's assumptions, Miller cannot be interpreted to suggest that the "first computer program", i.e. the word processor, configures the "input device", i.e. the detect structures button 520.

Furthermore, claim element 1e has additional requirements as to the "input device" required by the claims, not addressed at all by the Petitioners, namely that "the input device …allows the user to enter a user command to initiate an operation, the operation comprising (i) performing a search …" However, the

"Detect Structures" button, identified by the Petitioners as the "input device", does not initiate any search at all, but merely, at best, causes highlighting of the structures.

For at least the forgoing reasons, Miller fails to teach or suggest "providing an input device, configured by the first computer program", as required by the claims. Therefore, Ground 2 fails make a *prima facie* case that claims 1-44 would have been obvious based on Miller.

E. Because Miller's Program 165, and not the Application 167, receives the user's selection of the "detect structures" button, Miller fails to disclose or suggest the claim limitation of "receipt by the first computer program of the user command from the input device", and therefore Ground 2 fails to establish a *prima facie* case of obviousness.

As discussed in Section IV(C), all of the claims require the limitation "receipt by the first computer program of the user command from the input device". This limitation of the claims explicitly requires the first computer program to receive the user command from the input device. We will demonstrate that in contrast to the requirements of the claim, in Miller, the program 165 and not the application 167 receives the user command. Since the incorrect entity receives the user command, Miller fails to disclose or suggest "receipt by the first computer program of the user command from the input device", as required by the claims.

As discussed above in Section IV(D), Petitioners have equated application 167, a word processor, with the "first computer program" and the detect structures button 520 with the "input device". When Petitioners discuss of section 1h of claim 1, which includes the limitation of interest, the user selection of the detect structures button 520 must be the "user command" because it is the sole action mentioned in the analysis that is taken by the user: "When the user selects the `detect structures' button 520, a search is performed". See Petition, page 27.

The Petition points to nothing in Miller that discloses or suggests that the text editor receives the user selection of the detect structures 520 button. When the Petitioners apply Miller to section 1h, the manner in which the button selection is received is absent from their discussion. *Id*. Additionally, nowhere do the Petitioners even mention the text editor. *Id*. As with Petitioners' analysis of LiveDoc and Drop Zones, Petitioners' analysis of Miller with respect to section 1h ignores the first clause, "in consequence of receipt of the first computer program of the user command from the input device", and focuses exclusively on the second clause, "causing a search:" "Program 165 is a second program and includes analyzer server 220 that performs the search discussed in claim 1f". See Petitioner, page 27.

Miller indicates that the program 165, and not the application 167, receives the selection of the detect structures button 520. Since the program 165 performs

the namesake function of the detect structures button 520, the program 165 necessarily receives an instruction from the button 520: “The program 165 of the present invention is stored in RAM 170 and causes CPU 120 to identify structures in the data presented by application 167”. Exhibit 1007, col. 3, lines 38-41. Further, Miller explicitly connects operating the button 520 with initiating the program 165: “Window 510 includes a [detect structures] button 520 for initiating program 165, although alternative mechanisms such as depressing the "option" key may be used. Upon initiation of program 165, system 100 transmits the contents of document 210 to analyzer server 220, which parses the contents based on grammars 410 and strings 420 (FIG. 4)”. *Id.*, col. 5, lines 22-28. Because the detect structures button 520 initiates program 165, which, as pointed out above, executes independently, program 165 must receive an instruction to begin in response to a user selecting the detect structures button 520. Consequently, program 165 likely receives the user command. Nowhere does Miller disclose or suggest that the application 167 receives the user command and sends the user command to the program 165, via the application program interface 230, for the program 165 to begin executing. Therefore, in Miller, the program 165 and not the application 167 receives the user command.

Because Miller fails to disclose or suggest "receipt by the first computer program of the user command from the input device", as required by independent claim 1, Ground 2 fails to establish a *prima facie* case of obviousness.

F. Because Miller searches within the document for strings or grammars, Miller fails to disclose or suggest "performing a search using at least part of the first information as a search term … in an information source external to the document", and therefore Ground 2 fails to establish a *prima facie* case of obviousness.

Independent claim 1 requires "performing a search using at least part of the first information as a search term … in an information source external to the document". This limitation poses at least two requirements that are not met by Miller. First, the limitation requires the performance of a search that uses a search term. Second, the search must be conducted "in an information source external to the document". We will show that Miller fails to disclose or suggest either of these limitations. Further, these limitations are found in all the independent claims and hence are required by all of the claims challenged in Ground 2.

Regarding the first requirement for performing a search, in Petitioners' analysis of the limitation in section 1f, Petitioners argue that Fig. 4 shows a phone number being found using a name. See Petition, page 27: "For example, in the bottom box with the identified name, the actions are "Write letter" or "Call person (retrieve #)". (Fig. 4, 420.) In order to call the person, the name must be searched

in an address book to retrieve the associated phone number, as contemplated in Fig. 4". Although the Petitioners allege that "the name must be searched in an address book to retrieve the associated phone number", there is nothing whatsoever in Miller that discloses an automated search of the address book. The description of Fig. 4, in col. 5, lines 6-17, fails to disclose a search. In fact in all of Miller, the word "search" is mentioned only in connection with searching the document. Exhibit 1007, col. 3, lines 61-64 ("Analyzer server 220 comprises one or more pattern analysis units, such as a parser and grammars or a fast string search function and dictionaries, which uses patterns to parse document 210 for recognizable structures"), col. 4, lines 58-64 ("fast string search function" in analyzer server 220), col. 6, lines 34-55 ("fast string search function" for detecting patterns in document), col. 6, lines 64-66 ("neural net for searching a graphical document 210" or "a musical library for searching a stored musical piece 210"), columns 7 and 8 (claims to "fast string search"). Miller discloses nothing about what happens if the user selects the action that is the subject of speculation by Petitioners. In fact, the action recites no search, and states, instead, "Call person (retrieve #)". The fact that the phrase "retrieve #" is in parentheses and no other action includes a phrase in parentheses, suggests that the portion of the action "retrieve #" is in a different category from other actions, possibly because it is carried out manually.

Accordingly, there is nothing in Miller that discloses or suggests searching outside of the document. Even assuming that Miller were to implement such a functionality in the "Call person" action in Fig. 4 (though it does not), it would be the only action involving a search, and therefore this single search would fail to satisfy the claim requirement that "the specific type or types of second information is dependent at least in part on the type or types of the first information". The simple reason is that there would be only one type of first information (the name) and only one type of second information (the phone number). Moreover, there would be also only one action (since there is only one operation, namely "Call person (retrieve #)", and since there is only one action, the single action would fail to meet the claim requirement that "the action is of a type depending at least in part on the type or types of the first information".

Although it is not clear, the Petitioners may also attempt to equate structure detection with the "search": "Figs. 8-10 and 5:51-6:55 describe recognizing patterns and performing actions". See Petition, page 27. The pattern recognition referenced is finding grammars in the text of a document: "Parser 310 retrieves a grammar from grammar file 320 and parses text using the retrieved grammar". Exhibit 1007, col. 4, lines 62-64. However, the grammar file does not include the actual text of the structure, i.e. the actual telephone number, e-mail address. If the grammar repository were searched for the actual text of the structure (which Miller

does not do), the search would not yield any results. Therefore, the search for grammars is not "performing a search using at least part of the first information as a search term", as required by claim 1.

Regarding the second requirement for a search "in an information source external to the document", in Petitioners' application of Miller to this limitation, the sole entities that Petitioners identify as being "external to the document" are the dictionary (sometimes called "strings") and grammars of the analyzer server 220: "Analyzer server 220 includes dictionaries or `grammars' that are external to the document". See Petition, page 27.

In fact, Miller uses the grammars to parse the contents of a document to find structures, and obtains strings from the string library and searches for them in the document. Exhibit 1007, col. 3, lines 61-64. We will first discuss Miller's treatment of grammars and then his treatment of strings to show that in each case, they are applied to the document itself and fail to disclose or suggest "performing a search … in an information source external to the document", as required by claim 1.

Regarding "grammars", Miller says: "Parser 310 retrieves a grammar from grammar file 320 and parses text using the retrieved grammar". *Id*., col. 4, lines 62-64. Regarding the "dictionary" or "string library" (also called "name library"), in Fig. 10 of Miller, box 1070 recites the step "receive library of strings" and box

1080 recites the step "detect identical strings in data" (emphasis added). Miller elaborates by teaching that "[a]s illustrated in block 1060, a fast string search function retrieves 1070 the contents of string library 420, [and] detects 1080 the strings in the data identical to those in the string library 420" (emphasis added). *Id*., col. 6, lines 43-47. Miller also teaches that "[a]ssuming program 165 initiates with the receipt of any text, the received content or portion is scanned 820 for identifiable structures using the patterns in analyzer server 220". *Id.*, col. 5, lines 56-59. Miller also refers to a "fast string search" in the sting library. A fast string search is an algorithm in which one finds a string in a document, *i.e*., each entry in the dictionary is used in a search in the document (using the fast string search algorithm) to see if the string is in the document. In other words, it is the *text* in the document that is searched to identify *strings in the dictionary/string library*, not vice versa as required by the claim. Furthermore, the claims require *first* analysis, and in a *second step* using the result of the analysis as a search term. The purported searches regarding grammars and the dictionary/string library, would have been a part of this first step, analysis, and thus cannot also be used to satisfy the second step of the claim, namely the search.

Thus, Miller compares patterns (*i.e.*, strings, grammars) against the text of a document to determine if a pattern can be found therein. In this manner, Miller searches in the document for a pattern as a part of the analysis for structures. Thus,

Miller fails to disclose or suggest "performing a search using at least part of the first information as a search term … in an information source external to the document", as required by claim 1. For this additional reason, Ground 2 fails to make a *prima facie* case that claims 1-44 would have been obvious based on Miller.

G. Because the user informs the Luciw apparatus of the input's type of information, Luciw fails to disclose "analyzing, in a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information", and therefore Ground 3 fails to establish a *prima facie* case of obviousness.

Independent claim 1 requires "analyzing, in a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information". Indeed, this limitation is found in all the independent claims and hence is required by all of the claims challenged in Ground 3.

In their application of Luciw to this limitation in section 1c, the Petitioners argue that Luciw analyzes a user entry in a smart field: "While the document is being displayed, the device in Luciw analyzes a user's entry (first information from the document) to determine if implicit assistance is possible and the kind of implicit assist indicated (determine whether first information can be used to find

second information). See, e.g., Figs. 3 and 4a; 10:15-20 (`If the entry in the smart field has been made by the user, the assistance process takes action to identify or recognize the kind of implicit assistance indicated at a step 154.'); 8:7-13 (`At step 104, the process recognizes whether or not an implicit assistance function is to be provided by computer system 10. … If a user does enter information into a `smart field,' the computer database will be queried at step 106 to determine whether assistance is possible given the user input.')". See Petition, page 38.

Although the claim limitation requires "analyzing…to determine if the first information is at least one of a plurality of types of information", the cited portions of the Luciw patent fail to disclose any such analyzing whatsoever. In fact, as we demonstrate in detail below, the cited portions of the Luciw patent show that the user must tell the computing device of Luciw what type of information – *e.g.*, a name – is being entered. Therefore, whatever is entered by the user into the specified field is used without any analysis.

Petitioners apply the "implicit" assistance of Luciw to the "analyzing" of the subject patent's claims. During "implicit" assistance, a user uses a smart field to enter a word that can be looked up in a database, or executes predefined events that result in a database query. Exhibit 1008, col. 8, lines 14-33. Fig. 6a depicts the smart fields in question. Regarding this interface, the Luciw patent states that "The phone slip window 170 in FIG. 6a is shown with a smart name field 175". *Id.*, col.

11, lines 46-47. As explained earlier in Luciw, in order to use a smart field, the user must select a name or phone field depending on whether the textual item that the user wants to be searched is a name or a phone number:

> Once the particular window 170 is presented to the user, the name ISAAC can be handwritten into the particular smart field 175.… As will readily be recognized, window 170 in FIG. 4b may contain several smart fields, in this case for example definable for either the "name" field 175 or a "phone" field shown at step 177.

Id., col. 10, lines 28-39.

This passage makes clear that in order to retrieve information from the database, the user is expected to enter a name into the name field 175 or a phone number into the phone field 177. Note that in Figs. 6a, 6b, and 6c the name field and the phone field are given the same item numbers, 175 and 177 respectively, as in Figs. 4b and 4c discussed above. Thus, by using a smart field, a user tells the computing device what type of information the user is entering. In fact, no analysis to identify the type of text is performed or needed: the system simply assumes a type (*e.g.*, a name) because the user entered text into the corresponding smart field. Since smart fields are designed for user characterization of the type of information that is being entered, the user tells the computer system what type of information is being provided as an input by virtue of the field that receives the text. Since the user already provides the type of information, Luciw need not and

does not perform any "analyzing...to determine if the first information is at least one of a plurality of types of information" (emphasis added).

Further, since each smart field corresponds to a predefined type, the computer system knows the type of information the user entered simply by the identity of the smart field that receives the entered text. The computer system does not analyze the user input itself. For example, when a user enters the word "Isaac" into the name field 175, the computer system assumes that it has received a name simply because the name field 175 received the text. *Id.*, Fig. 6b. Nowhere does Luciw teach that the computer system has analyzed the word "Isaac" to determine that it is a name. Because the computer system does not consider the content of the input at all, the content of the "first information" is irrelevant for determining its type. Rather, the computer system just assumes that the type of information matches the type for the smart field. Therefore, Luciw fails to disclose or suggest "analyzing...first information from the document to determine if the first information is at least one of a plurality of types of information" (emphasis added).

Luciw indicates that other methods can be used to initiate a search based on text entered by a user. Luciw indicates that other forms of implicit assists "can be triggered by the happening of any of a number of predefined allowable events". *Id.*, col. 8, lines 30-41. However, the sole example that Luciw describes is writing a particular word or indication outside of a particular smart field: "Certain kinds of

events on screen 52, for example, such as the writing of a particular indication or word on screen 52 outside of a particular smart field may trigger an implicit assist". *Id*., col. 8, lines 30-41.

The assumption appears to be that whatever "particular indication or word" is entered will be used by the device for "a query of the database at step 106" of Fig. 3. Indeed, an inspection of Fig. 3, which "is a flow diagram of a process according to the invention for providing controlled computer-assisted user assistance", fails to uncover any step of analyzing text in a document to determine whether text "is at least one of a plurality of types of information that can be searched for in order to find second information". *Id*., Fig. 3; col. 2, lines 65-67, and col. 8, line 1-col. 10, line 5. An inspection of Fig. 3 shows that the only instance wherein the database is queried is in step 106, and that step is preceded simply by a determination, in step 104, whether an "implicit assist" has been invoked, and the database query follows if the determination is that an "implicit assist" has been invoked. There is no analyzing step.

In summary, Luciw fails to disclose any mechanism for "analyzing, in a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information". Rather, the user alerts the device as to the type of information that the user is inputting via selection of the appropriate smart field. Thus, the user informs the Luciw device of the type

of information, such as a name or phone number. Since Luciw fails to disclose each and every limitation of the subject patent's claims, Ground 3 fails to make a prima facie case for obviousness of claims 1-44.

H. Because Pandit's nouns and verbs are not the "types of information" contemplated by the claims, and, at best, the type of second information is decided by the user, and not dependent at least in part on the type or types of first information, Pandit fails to disclose or suggest "performing a search … wherein the specific type or types of second information [found] is dependent at least in part on the type or types of the first information [used as the search term]", therefore Ground 4 fails to establish a prima facie case of obviousness.

Independent claim 1 recites "performing a search … wherein the specific type or types of second information [found] is dependent at least in part on the type or types of the first information [used as a search term]". Indeed, this limitation is found in all the independent claims and hence is required by all of the claims challenged in Ground 4.

Petitioners attempt to apply two examples in Pandit to this limitation of the claim: searching a dictionary for a meaning of a word, and adding an identified telephone number to an address book. See Petition, pages 50-51. We will first address how searching a dictionary fails to meet this limitation, and in Section III(I), we will address how adding a telephone number to an address book also fails to meet the limitation.

The recited "search" of the subject patent's claims includes a number of requirements that are not met by obtaining a word's meaning. First, because the claim refers to the types of information for the first and second information, the "first information" and "second information" each must be of a specific type or types of information. We will show that the types of information used in the dictionary search as not the "type or types of information" contemplated by the claims. Second, the latter half of the limitation requires the type of the "second information" to depend upon the type of the "first information". We will show that even if Pandit's dictionary were to involve the types of information required by claim 1, which it does not, the type of word input into the dictionary program is irrelevant to the type of information obtained. Since dictionaries always obtain meanings of words, the dictionary necessarily retrieves the same type of information.

The first embodiment of Pandit that the Petitioners use detects nouns or verbs in a document. See Petition, page 51. To the best of Patent Owner's understanding, Petitioners appear to equate the detected noun or verb with the "first information". Since Petitioners cite a dictionary function for the "search", Petitioners equate the dictionary entry of the noun or verb with the "second information". Also, because the Petitioners reference executable programs in Pandit beyond the dictionary, other potential types of "second information" would

include synonyms of the word, the singular or plural version of a noun, or the conjugation of a verb. See Petition, page 51. However, the subject patent describes telephone numbers, fax numbers, and e-mail addresses as exemplary types of information. Exhibit 1001, col. 4, lines 12-14. In light of the specification, one of ordinary skill in the art would recognize that mere nouns and verbs are non-analogous to entities such as telephone numbers, fax numbers, and e-mail addresses. As such, one of ordinary skill in the art would not recognize nouns and verbs as the "types of information" contemplated by the claims. Therefore, the first embodiment of Pandit that Petitioners rely upon is inapplicable to the subject patent's claims.

Further, even if the entities in this embodiment of Pandit were "types of information", Pandit still fails to describe or suggest that the "specific type or types of second information is dependent at least in part on the type or types of the first information". Instead, the type of second information depends on the selection that a user makes from a pull-down menu of programs: "Where the invention is capable of recognizing nouns or verbs, pull-down menus can, for example, identify executable programs which provide the meaning of the highlighted word, appropriate synonyms and the singular or plural version of the noun or conjugation of the verb". Exhibit 1009, col. 3, lines 12-16. We note that Pandit fails to teach

how these identified programs operate. Nowhere does Pandit indicate that any of these programs "performing a search", as required by claim 1.

Regardless, the particular program that the user selects from the pull-down menu determines the type of information that Pandit will retrieve. Once the user selects a program, the system of Pandit obtains the type of information associated with that program, regardless of the type of information associated with the input. For example, if a user selects the dictionary program, the dictionary will retrieve the dictionary entry of a word, regardless of whether that word is a noun or verb. Likewise, if a user selects the thesaurus program, the thesaurus will retrieve the synonym entry of a word, regardless of whether that word is a noun or verb. Thus, the type of information that Pandit retrieves (e.g., the type of the "second information") is actually dependent on the user request for a program that obtains that particular type of information, not on the type of the information that is input to the program (e.g., the type of the "first information", such as a noun or verb).

For at least the forgoing reasons, Petitioners have failed to demonstrate that Pandit discloses or suggests "performing a search … wherein the specific type or types of second information [found] is dependent at least in part on the type or types of the first information [used as the search term]". Therefore, Ground 4 fails to make a prima facie case that claims 1-44 of the subject patent would have been obvious.

I. Because Pandit's does not disclose searching in the address book , Pandit fails to disclose or suggest "performing a search using at least part of the first information as a search term in order to find the second information" and "causing a search for the search term", and therefore Ground 4 fails to establish a *prima facie* case of obviousness.

We now turn to the second example of Pandit that Petitioners attempt to apply to the claimed "search:" adding an identified number to an address book. See Petition, page 51. Petitioners attempt to equate a telephone number with the "first information" and contact information associated with the telephone number with the "second information" based on Pandit's disclosure of "adding an identified number to an address book". See Petition, page 51. However, and as admitted by the Petitioners, Pandit only discloses adding a telephone number to an address book which does not require a search in the address book, and indeed Pandit does not disclose any such search. Pandit does not, contrary to the statements by Menascé, disclose ensuring that there are no multiple entries of the same address in the address book.

As with Miller, Petitioners cannot rely on text or figures within the four corners of Pandit to disclose all of the limitations of the subject patent's claims. Again, they resort to expert testimony to interpret Pandit in their favor: "[I]t would have been obvious to a person of ordinary skill in the art that the first step in adding to an address book is searching the address book to determine if an entry

already exists with this information and displaying any associated information which is located. (Menascé Decl. ¶99.) This would have been a matter of common sense to one of ordinary skill, in order to avoid multiple entries of the same address". See Petition, page 51. Therefore, Petitioners admit that Pandit fails to disclose "performing a search", but attempt to argue that this search "would have been obvious".

Petitioners cannot point to any teachings in Pandit that might lead one of ordinary skill in the art to this conclusion. Since Pandit does not describe any process by which its system adds e-mail addresses or telephone numbers to an address book, Pandit lacks any teachings from which one of ordinary skill in the art could deduce its operation. Instead, Petitioners rely on the subjective, and amorphous, basis of "common sense". However, in light of Pandit's silence, one of ordinary skill could just as readily use "common sense" to conclude that selecting the "Add to address book" option would cause the computer system to open the address book itself and create a new entry. Since nothing in Pandit teaches a search through the address book, Petitioners' argument is based on their importation of the subject patent's limitation into their understanding of the text.

Even assuming for the sake of argument that Pandit were to contemplate searching for duplicate entries, which he does not, such a search would still fail to meet the requirements of the claim. Suppose Pandit received a request to add a

telephone number to the address book. To determine if an entry for this telephone number already existed, Pandit would search its entries for the telephone number. Further, Pandit would be concerned only with finding the telephone number in its records, not with any other information. Therefore, a search for duplicate entries would be a search for "first information", not a search "in order to find the second information, of a specific type or types", as required by claim 1.

For at least the forgoing reasons, Petitioners have failed to demonstrate that Pandit discloses or suggests "performing a search using at least part of the first information as a search term in order to find the second information" and "causing a search for the search term". Therefore, Ground 4 fails to make a prima facie case of demonstrating that claims 1-44 of the subject patent would have been obvious.

CONCLUSION

For the foregoing reasons, Petitioners have failed to establish a reasonable likelihood of prevailing as to any claim of the '843 Patent, and *inter partes* review of claims 1-44 of U.S. Patent No. 7,917,843 should be denied.

Dated: March 12, 2014

Respectfully submitted,

/Robert M. Asher, #30,445 /

Robert M. Asher
Registration No. 30,445
Bruce D. Sunstein
Registration No. 27,234
Dorothy Wu
Registration No. 69,535
Sunstein Kann Murphy & Timbers LLP
125 Summer Street
Boston, MA 02110
Tel: (617) 443-9292
Fax: (617) 443-0004

CERTIFICATE OF SERVICE

It is certified that on March 12, 2014, copies of the Preliminary Response of the Patent Owner under 35 U.S.C. § 313 and 37 C.F.R. § 42.107 has been served on Petitioners as provided in 37 C.F.R. § 42.6(e) via electronic mail transmission addressed to the persons at the following addresses:

LEAD COUNSEL FOR PETITIONER APPLE DAVID L. FEHRMAN dfehrman@mofo.com Registration No. 28,600 MORRISON & FOERSTER LLP 707 Wilshire Blvd., Suite 6000 Los Angeles, CA 90017-3543 Tel: (213) 892-5200 Fax: (213) 892-5454	BACK-UP COUNSEL FOR PETITIONER APPLE MEHRAN ARJOMAND marjomand@mofo.com Registration No. 48,231 MORRISON & FOERSTER LLP 707 Wilshire Blvd., Suite 6000 Los Angeles, CA 90017-3543 Tel: (213) 892-5200 Fax: (213) 892-5454
LEAD COUNSEL FOR PETITIONERS GOOGLE AND MOTOROLA MOBILITY MATTHEW A. SMITH smith@turnerboyd.com Registration No. 49,003 TURNER BOYD LLP 2570 W. El Camino Real, Suite 380 Mountain View, CA 94040 Tel: (650) 265-6109 Fax: (650) 521-5931	BACK-UP COUNSEL FOR PETITIONERS GOOGLE AND MOTOROLA MOBILITY ZHUANJIA GU gu@turnerboyd.com Registration No. 51,758 TURNER BOYD LLP 2570 W. El Camino Real, Suite 380 Mountain View, CA 94040 Tel: (650) 265-6109 Fax: (650) 521-5931 and kent@turnerboyd.com docketing@turnerboyd.com

Date: March 12, 2014

/Robert M. Asher, #30,445 /

Robert M. Asher
Registration No. 30,445
Sunstein Kann Murphy & Timbers LLP
125 Summer Street
Boston, MA 02110
Tel: (617) 443-9292
Fax: (617) 443-0004