UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

X1 DISCOVERY, INC.,
Patent Owner

IPR2025-00253
U.S. Patent No. 7,370,035

SUPPLEMENTAL DECLARATION OF MARTIN P. HAEBERLI

TABLE OF CONTENTS

I. Introduction, qualifications, and materials considered 1

II. Summary of Opinions 2

III. Claim Construction 3

IV. Background 3

A. Prior Art Search Technologies 3

1. Searching Within a Single Document 5

2. Searching for Documents 6

B. Incremental Searching for Documents Was Not "Well Known" at the Time of the Invention 8

V. GROUND 1: The Challenged Claims are Patentable Over the Combination of Wilcox (EX1059), Londergan (EX1061), Raskin (EX1021), and Wu (EX1019) 12

A. Dr. Terveen Has Not Provided Motivation to Combine the Ground 1 References 12

1. Dr. Terveen Provides No Motivation to Combine the Four Distinct Search Functionalities Described in the Lotus Notes References 12

2. Dr. Terveen Provides No Motivation to Combine the Lotus Notes References with Raskin 16

3. Dr. Terveen's Analysis of Reasonable Expectation of Success is Conclusory and Incorrect 28

B. The Lotus Notes-Raskin-Wu Combination Does not Render Obvious All Limitations of the Challenged Claims 32

1. Elements [1.4] and [10.4] 32

2. Elements [1.3], [1.5], [10.3], [10.5] 34

3. Claims 2-9 37

4. Claims 2-4 38

5. Claims 7 and 9 43

6. Claim 8 46

VI. GROUND 2: The Challenged Claims Are Patentable Over the Combination of Schwartz (EX1004), Sebastian (EX1005), Barber (EX1006), True (EX1009), and Baeza-Yates (EX1023). 47

A. Dr. Terveen Has Not Provided Motivation to Combine the Ground 2 References 47

1. Dr. Terveen Has Not Provided Motivation to Combine the Elements from Different Search Systems in the Entourage References 47

2. True is Not Analogous Art to the ’035 Patent 48

3. Dr. Terveen Has Not Provided Motivation to Combine the Entourage References with True.....55

4. Dr. Terveen’s Analysis of Reasonable Expectation of Success is Conclusory and Incorrect 56

B. The Entourage-True-Baeza-Yates Combination Does Not Render Obvious All Limitations of the Challenged Claims 57

1. Claim 3-4.....57

VII. Conclusion 58

I. INTRODUCTION, QUALIFICATIONS, AND MATERIALS CONSIDERED

1. I have been retained by counsel for Patent Owner X1 Discovery, Inc. ("X1") in the above-captioned proceedings. I understand that Petitioner Microsoft Corporation ("Petitioner" or "Microsoft") seeks *Inter Partes* Review ("IPR") of U.S. Patent No. 7,370,035 (the "'035 Patent").

2. I previously prepared and submitted the Declaration of Martin P. Haeberli ("First Declaration," EX2001), which I fully incorporate here by reference.

3. Since preparing my First Declaration, my qualifications and compensation have not changed. I am being reimbursed at my standard hourly consulting rate. My compensation is not affected by or dependent in any way on the outcome of this matter.

4. In forming my opinions expressed in this Declaration, I relied upon my education and experience, which I described in my First Declaration and in my *curriculum vitae* (EX2002). In addition to the materials listed in my First Declaration and the materials listed in the Appendix to this document, I reviewed and considered the following:

a. Paper 7, Patent Owner's Request for Discretionary Denial

b. Paper 8, Patent Owner's Preliminary Response

c. Paper 9, Petitioner's Opposition to Patent Owner's Request for Discretionary Denial

d. Paper 11, Patent Owner's Reply in Support of Request for Discretionary Denial

e. Paper 12, Petitioner's Sur-Reply in Support of its Opposition to Patent Owner's Request for Discretionary Denial

f. Paper 13, Decision Denying Patent Owner's Request for Discretionary Denial

g. Paper 14, Institution Decision

5. I am over 18 years of age. I have personal knowledge of the facts stated in this Supplemental Declaration and could testify competently to them if asked to do so.

II. SUMMARY OF OPINIONS

6. None of the '035 Patent's challenged claims would have been obvious in light of the prior art cited by Petitioner, or combination of prior art, or under the knowledge of a person of ordinary skill in the art because each asserted prior art reference lacks at least one element of the asserted claims and because a person of ordinary skill in the art ("POSITA") would not have been motivated to modify or combine the prior art at the time of the invention in the ways asserted by Petitioner.

III. CLAIM CONSTRUCTION

7. As I explained in my first declaration, I have applied the plain and ordinary meaning of the term as it would be commonly understood by a POSITA at the time of the invention. (EX2001, ¶37.) I also understand that the panel construed the term "*string separator character*" to mean "a single-character string separator." (Paper 14, 10.) I agree with the panel's preliminary construction.

IV. BACKGROUND

A. Prior Art Search Technologies

8. Prior to 2002, the most common type of search was a delimited search in which a user would type in a full search string and then press Enter or hit a Search button to execute the search. (EX1001, 1:32-2:5.) At the time, incremental searching, in which search results are returned in real-time as the search string is typed into the search field, character-by-character, was much less common. (EX1021, 126 ("almost all interface-building tools make it easy to implement delimited searches and difficult or impossible to implement incremental searches").)

9. Different search systems in the prior art were organized into different categories, each of which had different coding, structure, and user experiences. A search could be conducted by sequentially scanning the words in one or more documents to find characters that match a search string (sometimes called a "brute force" search). (EX1023, 191.) Alternatively, a search could utilize an index to

avoid having to look at every word of every document upon every search. (See EX1023, 191-92.) Searches could be either delimited or incremental. (EX1001, 1:32-42, 2:9-21.) Delimited search was most common and required a user to take an action such as hitting a search button before the search was initiated, whereas a search that was incremental would be conducted as the user typed their query. (*Id.*) Searches could allow for single string (search term) queries where a string might be a single word or multiple words treated together as a single string, or multiple string queries where each string is treated as separate in the search. (*See* EX1021, 125; EX1019, 6:54-62.) And a search system might implement the ability to perform multiple string queries using multiple search fields which each accept a single string (EX1059, 163), or by allowing for multiple strings in a single search field separated by a string separator character or, alternatively, by a Boolean operator (EX1059, 162-67). Finally, searches could be characterized as occurring in a single open document (EX1022, 55), or across a large distributed collection of documents (EX1059, 161-64; EX1061, 139-42).

10. As I explain below, none of the prior art Dr. Terveen relies on as background art or in the Grounds discloses incrementally searching for and locating documents. For the rare instances in the art in which incremental search is disclosed, it is limited to searching (with only a single search string) the content of a single

open document (EX1022, 125) or filtering lists (with only a single search) string over a single field (EX1004, 8; EX1005, 183; EX1006, 358, 384.)

1. Searching Within a Single Document

11. As one example, Raskin describes the search functionality in the EMACS text editor. (EX1021, 125-27.) To perform a search in EMACS, the user presses Control-S to search forward through the text and then enters a series of characters. (EX1022, 55.) As the characters are entered, the system sequentially scans through the text of the document and "positions the cursor at the first occurrence of the characters [the user] typed." (EX1022, 55.) This is able to be done quickly because the single, open document is loaded into the computer's memory (i.e. the active buffer). Once the user locates the desired text within the document, he or she can press the Escape key to stop searching; then the cursor remains at the last found location. (EX1022, 55.) If the first instance of the search string found within the document is not what the user is looking for, the user can press Control-S again to advance the cursor to the next instance of the search string in the document. (EX1022, 56.)

12. Raskin's description of the EMACS text editor thus discloses a form of incremental, sequential scanning for a single search term in the text within a single document that was common at the time of invention and is still in use today for searching for strings within a single document. This type of search functionality

is also referred to as "brute-force". The brute-force algorithm is the simplest approach for string searching, which involves comparing each and every chracter in the text that is being searched against the text that is being searched for. (EX1023, 209.) In my opinion, Raskin's description of EMACS does not disclose incrementally searching for and locating documents. It does not disclose returning a list of search results. And it does not disclose incrementally searching on multiple search strings.

13. I am not aware of any prior art before 2002 that discloses incrementally searching within a single open document for multiple search strings. None of Dr. Terveen's cited references disclose such a search system.

2. Searching for Documents

14. All of the prior art references Dr. Terveen cites that disclose searching for and locating documents employ delimited searches. In other words, a user would make use of a search bar or call up a special search interface (such as Search Builder or Query Builder in Lotus Notes), and then perform the search by typing in a search query and then hitting Enter or pressing a Search button. (EX1059, 161-64; EX1061, 139-42.) Some delimited search systems allowed for a user to input multiple search strings in a single search field separated by a character such as a comma or by using Boolean operators to connect them. (EX1059, 162; EX1061, 140-41.)

15. The type of sequential scanning typically employed to search a single open document is very time and resource intensive if employed to search a large set of documents. Instead, many systems for searching for documents employ index data structures to improve the speed and efficiency of searches. (EX1059, 159; EX1061, 144; EX1023, 191-92.) A simple index is a data structure that associates the words in the indexed documents with the occurrences of the words in each document. (EX1023, 192-93.) More complex indexed systems may involve multiple index data structures such as suffix trees, suffix arrays, and signature files. (EX1023, 199-207.)

16. As I explain in more detail in my discussion of Ground 1, the size of an index has a significant impact on the performance of the search system. A larger index will generally result in a slower search and will also consume more system resources. (*Infra* §IV.A.2; EX2022, 6 ("To improve performance, we built a reduced version of the index"); EX2021, 38:1-39:17 (Dr. Terveen testifying that a larger index "could increase the time it took to search it" and that "making the index smaller would be more efficient in general."); EX2021, 39:15-16 ("if you add more information, you will be taking more memory"); EX1059. 169 (explaining that including case-sensitive entries in the index increases the size by 25%); EX1023, 193 ("occurrences demand much more space" in the index).)

17. I am not aware of any prior art before 2002 which discloses incrementally searching for and locating documents. None of Dr. Terveen's cited references disclose such a search system.

B. Incremental Searching for Documents Was Not "Well Known" at the Time of the Invention

18. In the years leading to the invention of the '035 patent, a variety of methods and systems were proposed to address some of the limitations of prior art systems, many of which were described in the alleged "State of the Art" by Dr. Terveen. (EX1003, ¶¶32-50.) For example, recognizing that it would be too resource and time consuming to do full text searches of a person's data, some approaches involved having a search query that does not actually search the underlying documents themselves, but searched pre-determined lists of search strategies that the developer determined would be the most likely to be used. (*See* EX1011, Abstract, Figs. 7B, 8; EX2001, ¶42.) Thus, a user was not selecting from among documents, but instead was searching a list of pre-populated search strategies. (*Id.*) Only once a user hit enter would all the underlying documents that had been characterized as matching that pre-determined search strategy appear as results. (*Id.*) Systems such as this did not incrementally search across multiple documents as a query was entered. Indeed, none of the disclosures Dr. Terveen points to discloses a method of incrementally searching for and locating documents using multiple search terms.

19. Dr. Terveen cites numerous references he alleges show that incremental search was "well known" at the time of invention. (EX1003, ¶¶33-39.) In this portion of his report, Dr. Terveen characterizes incremental searching as "incrementally locating and displaying search results to a user as they type their query." (EX1003, ¶33.) Similarly, in characterizing the '035 patent, Dr. Terveen describes incremental search as "searching where search results are displayed as the user types their search query." (EX1003, ¶25.) These characterizations of incremental search are premised on the idea that incremental search is a one-size-fits-all solution. I disagree. In my opinion, a POSITA would understand that the implementation of incremental search functionalities varies based on the search targets and underlying data structure. Specifically, I do not agree with Dr. Terveen's characterizations of incremental search in the context of the '035 patent, which requires incrementally searching for and locating *documents* (among a collection or multitude of documents). None of Dr. Terveen's examples disclose this type of incremental search.

20. In my First Declaration, I analyzed the numerous references Dr. Terveen cited that he claimed show that incremental search was "well known" at the time of invention. (EX2001, ¶¶39-47.) As I previously explained, in my opinion, none of those references disclose incrementally searching for and locating documents including two strings entered into the same search field. (EX2001, ¶¶39-

47.) Below I provide additional analysis of some of these references to further explain why none of the references establish that incrementally searching for and locating documents was "well known" as Petitioner alleges.

21. Dr. Terveen alleges that Gabbe "described using incremental searching to provide feedback on the volume of search results as the user typed their query." (EX1003, ¶¶33-34.) I disagree. Gabbe describes searching a list of superimposed code words representing attribute values. (EX1014, 7:64-70, 8-1:7.) Gabbe's search does not return a list of search results; rather it provides the user with feedback as each character is entered: a single number representing the number of records having a specific attribute that includes a word that begins with the character string. (EX1014, 7:11-38, Fig. 1.) Thus, Gabbe does not describe incrementally searching for and locating documents.

22. Dr. Terveen alleges that "in 1994, the background section of [Chew] described 'the incremental search method' as being known." (EX1003, ¶36.) I do not agree that Chew's reference to the incremental search method is relevant to the invention claimed in the '035 patent. Chew does not disclose incrementally searching for and locating documents. Rather, Chew describes searching a list of elements, "where each list element is a sequence of characters" (EX1015, Abstract). Chew might be applicable, for instance, to rapidly selecting a starting location in an alphabetized list of fonts in a pop-up menu or other menu in a user interface.

23. Dr. Terveen alleges that Siitonen describes using incremental search to "locate PDA contacts." (EX1003. ¶37.) Again, this is not incrementally searching for and locating documents as contemplated by the '035 patent. Siitonen describes searching a directory of contact information that is "similar in appearance to a telephone book listing." (EX1016, Abstract.) In other words, Siitonen discloses searching within or filtering a list. Siitonen does not disclose incrementally searching for and locating documents.

24. Dr. Terveen alleges that Colwell describes using incremental searching to "locate locally-saved files on a hard drive." (EX1003, ¶37.) I disagree. Colwell describes a two-step search process. First, a user "establishes a view … by issuing [a] primary search request." (EX1017, 9:50-53.) The primary search—e.g., "Explore all files concerning 'Apple'"—"locate[s] the given data in the index file 6, the interface module 12 displays a list of data files 50 containing the words in the request." (EX1017, 4:53-57.) The primary search is a traditional delimited search. (EX1017, 5:23-34.) Only after the primary search results are returned and the user "establishe[s] a view" of one of the resulting files can the user perform an incremental "secondary search request within a displayed data file." (EX1017, 4:47-49, 9:50-55.) Thus, the incremental search disclosed in Colwell is a search within a single document open in the viewer; Colwell does not disclose incrementally searching for and locating documents.

25. Dr. Terveen alleges that Tafoya shows that "incremental searching … was commonly used in … email clients." (EX1003, ¶33.) However, like most of Dr. Terveen's cited references, Tafoya describes incremental list searching. Specifically, Tafoya describes searching a list of email addresses that is "preferably constrained to ensure that the user does not notice a lag time while the software searches for probable matches." (EX1029, 11:42-44.) As with Dr. Terveen's other references, Tafoya does not disclose incrementally searching for and locating documents, as required by the Challenged Claims.

V. GROUND 1: THE CHALLENGED CLAIMS ARE PATENTABLE OVER THE COMBINATION OF WILCOX (EX1059), LONDERGAN (EX1061), RASKIN (EX1021), AND WU (EX1019)

A. Dr. Terveen Has Not Provided Motivation to Combine the Ground 1 References

1. Dr. Terveen Provides No Motivation to Combine the Four Distinct Search Functionalities Described in the Lotus Notes References

26. Dr. Terveen proposes a combination that draws from four distinct Lotus Notes search functionalities, but it is my opinion that his declaration fails to explain why a POSITA would have been motivated to combine these functionalities in the first instance. I discussed three of the four Lotus Notes search functionalities in detail in my First Declaration, including (1) Search bar, (2) Search Builder, and (3) Query Builder. (EX2001, ¶¶79-81.) The Lotus Notes references also disclose, and Dr. Terveen relies upon, a fourth search functionality—Form-based searching.

27. The Lotus Notes references describe searching by forms. A "form" in the Lotus Notes System is a predefined document template that specifies the fields and structure of documents created using that form. (EX1059, 58 ("A form is a blank version of a document, and you turn it into an actual document by filling in its blanks and then saving it in a database."); EX1061, 143.) Forms are created by the database designer and are part of the design of a Lotus Notes database. (EX1059, 58.) The form defines which "fields" appear in the resulting document, and each field "can contain a particular kind of data." (EX1059, 58; *see also* EX1061, 143.) When a user creates a document, the user selects a form, and the document inherits the fields and layout associated with that form.

28. The Lotus Notes references also describe a "Searching Using a Form" or "Query by Form" feature that allows a user to restrict a search to documents created using a specific form and to search for text within one or more specified fields of that form. (EX1059, 167 ("You can use the forms in a database to create queries that search for text in a particular field."); EX1061, 143.) In other words, this search functionality is limited to a subset of documents—specifically, those created using the selected form. Documents created using different forms are not searched in this functionality. (EX1061, 143 (explaining that "Query by form" should be used when "you only want one type of document.").)

29. Based on my review, the searching by form functionality is a field-constrained delimited search tool that depends on the predefined structure of documents associated with the chosen form. There is no indication that searching by form performs incremental searching where search results dynamically update character-by-character as the user types. Nor does the searching by forms functionality disclosed in the Lotus Notes references teach entering multiple partial search strings into a single search field separated by a string-separator character and having the system treat each partial string independently as the user types.

30. In sum, Dr. Terveen's opinions rely on drawing together several distinct search functionalities within Lotus Notes, each of which operates differently and was intended for different use cases within the system: (1) the Search Bar, where a user enters a query as a single text string using Boolean operators; (2) the Search Builder interface, which requires the user to enter separate search terms into different field-specific input boxes (Londergan, EX1061); (3) the Query Builder interface, which enables the construction of comma-delimited structured queries within a single text field (Wilcox, EX1059); and (4) the Form-based search mechanism, which restricts searching to documents created using a particular form and to fields defined on that form. All of these comprise delimited searches. In my opinion, these functionalities represent alternative search modes rather than modular components of a unified search workflow, and a POSITA would have understood

them to be different ways of performing a search, chosen based on context, rather than features to be merged together.

31. Further, Dr. Terveen does not explain why a POSITA would have chosen to combine these separate search modes in the manner required by the Challenged Claims. His declaration states only that a POSITA "would have been motivated to combine the complementary teachings on Lotus Notes from Londergan (EX1061) and Wilcox (EX1059) to obtain a more complete and comprehensive understanding of the Lotus Notes system, its features, and its capabilities." (EX1003, ¶73.) In my view, a POSITA would have recognized that the Search Bar's single-field Boolean querying, the Search Builder's multiple field-specific input boxes, the Query Builder's comma-delimited structured queries, and the Form-based field-restricted searching each reflect different assumptions about how a user selects search terms, how results are filtered, what subset of documents is subject to the search, and how precisely a delimited search is initiated. Yet Dr. Terveen does not identify any problem, inefficiency, or design objective in Lotus Notes that would have prompted a POSITA to replace these discrete functionalities with a single search field that accepts multiple partial search strings separated by a string-separator character, with the ability to limit the search to specific fields. All Dr. Terveen appears to do in his declaration is identify disclosures in the Lotus Notes references of each of these separate and independent features, but he does not

articulate why a POSITA would have modified Lotus Notes to combine them in the way the claims require.

2. Dr. Terveen Provides No Motivation to Combine the Lotus Notes References with Raskin

a. The Search Technologies Disclosed in the Lotus Notes References and Raskin Are Fundamentally Different and Incompatible

32. The Lotus Notes references describe a system that is built around a database architecture that requires an index to enable full-text searching of documents stored in the database. (EX1059, 159; EX1061, 144.) These references provide detailed explanations of the various configuration options available when creating an index, such as whether to enable case-sensitive searches, and importantly, they explain how each of these options affects the size of the resulting index. (EX1059, 169; EX1061, 138-39.)

33. From a technical standpoint, the size of the index has a direct and significant impact on search performance—specifically, larger indexes result in slower search operations. (*See* EX2022, 6 ("To improve performance, we built a reduced version of the index"); EX2021, 38:1-39:17.) This relationship exists because larger indexes consume more processing power and require greater amounts of available RAM to operate efficiently. (*See* EX2021, 39:15-16 ("if you add more information, you will be taking more memory"); EX1059. 169 (explaining that including case-sensitive entries in the index increases the size by 25%);

EX1023, 193 ("occurrences demand much more space" in the index).) Dr. Terveen himself acknowledged this fundamental principle during his deposition, testifying that a larger index "could increase the time it took to search it" and that "making the index smaller would be more efficient in general." (EX2021, 38:1-39:17.)

34. As I explained herein, the architecture of the Lotus Notes search system is designed around what are known as delimited searches. Dr. Terveen admits as much in his declaration. (EX1003, ¶63, 67 ("the Lotus Notes references do not expressly disclose performing an incremental search").) In a delimited search interface, the user must complete their query entry and then explicitly click a "Search" button of some kind to initiate the search operation and receive results. (EX1003, ¶67.) This is a fundamentally different interaction model from incremental search.

35. In contrast, the portion of Raskin that Dr. Terveen relies upon to allegedly demonstrate incremental searching consists of a high-level conceptual discussion of the search functionality implemented in the GNU EMACS text editor. (EX1003, ¶¶65-67; EX1021, 125-27.) Only by reviewing the EMACS manual itself can one get a sense of how EMACS worked in practice. The search capability in EMACS operates within the confines of a single text document that is currently open in the editor (i.e., loaded into the active buffer)—it is not a system for searching

across multiple documents or for locating documents within a file system. (EX1022, 125.)

36. As I explained above, the EMACS search mechanism is activated by the user pressing the "Control-s" key combination to initiate a forward search through the document, followed by typing the characters that form the search string. (*Supra* §IV.A.1; EX1022, 55.) EMACS then performs what is known as a sequential search, which involves scanning through the text character-by-character within the currently open document to find matches for those characters. (EX1022, 55.) This is a brute force approach—Raskin's disclosure concerns non-indexed, sequential scanning within a single document (in EMACS terminology, the "buffer") using a single search term.

37. The technical differences between these two search methodologies are substantial and represent fundamentally different approaches to the problem of finding information. The Lotus Notes system uses pre-computed indexes to enable rapid searching across large document collections, while the EMACS system described by Raskin uses real-time sequential scanning of text that is already loaded into the computer's active memory.

b. Dr. Terveen's Stated Motivation to Combine Fails on Technical Grounds

38. Dr. Terveen's stated motivation for combining Raskin with the Lotus Notes references is that doing so will allegedly result in improvements in "speed,

effectiveness, and ease of use." (EX1003, ¶75.) However, this motivation is contradicted by the technical realities of how these systems actually work.

39. During his deposition, Dr. Terveen testified that for his proposed combination, he was simply extracting from Raskin the abstract "idea of incremental search" and, as his motivation to combine, the alleged benefits of performing incremental searches. (EX2021, 114:10-19, 116:20-118:6.) Critically, Dr. Terveen testified that he did not consider the specific context in which Raskin was using incremental search—namely, sequentially scanning, or brute force search, within a single open document—and stated that considering such implementation details was not required for his obviousness opinion. (EX2021, 116:10-117:17.)

40. This approach is technically problematic. In my opinion, a POSITA would not simply extract an abstract concept from a reference without understanding how that concept is actually implemented and whether that implementation is compatible with the target system. The technical context matters because the feasibility and benefits of a feature depend entirely on how it is implemented.

41. The first critical incompatibility between Lotus Notes and Raskin is that each describes fundamentally different search methodologies. Raskin discloses a brute force search system that operates by sequentially scanning every character

in the text of a single open document to find a single matching character string. (*Supra* §IV.A.1.) Raskin does not even disclose how such a system could be adapted to handle multi-word searches, nor to multi-document searches.

42. The Lotus Notes references, by contrast, disclose an indexed database system in which the full text of all documents in the database has been pre-processed and indexed, and where users can employ multiple search strings either in separate search fields, or combined using Boolean operators or with a string separator character. (*Supra* §V.A.1.) Dr. Terveen's analysis fails to acknowledge these fundamental and meaningful differences in search functionality.

43. From a technical implementation perspective, these are not merely different features—they represent entirely different architectural approaches to information retrieval. A system that searches by sequential scanning performs no preprocessing but must examine every character of every document at query time. (EX1023, 191 ("An obvious option in searching for a basic query is to scan the text sequentially. Sequential or online text searching involves finding the occurrences of a pattern in a text when the text is not preprocessed. Online searching is appropriate when the text is small (i.e., a few megabytes), and it is the only choice if the text collection is very volatile (i.e., undergoes modifications very frequently) or the index space overhead cannot be afforded.").) An indexed search system pre-processes documents to create data structures that enable rapid lookup, trading

storage space and preprocessing time for query speed. (EX1023, 191 ("A second option is to build data structures over the text (called indices) to speed up the search. It is worthwhile building and maintaining an index when the text collection is large and semi-static. Semi-static collections can be updated at reasonably regular intervals (e.g., daily) but they are not deemed to support thousands of insertions of single words per second, say. This is the case for most real text databases, not only dictionaries or other slow growing literary works. For instance, it is the case for Web search engines or journal archives. Nowadays, the most successful techniques for medium size databases (say up to 200Mb) combine online and indexed searching.").)

44. A POSITA would not be motivated to combine these references because he or she would understand that implementing Raskin's incremental search approach in any embodiment of the Lotus Notes system would not result in improved search functionality. Dr. Terveen's stated motivation to combine Raskin with the Lotus Notes references is "speed, effectiveness, and ease of use," and he testified that increased speed means "the search being faster for the user." (EX2021, 106:7-109:6.)

45. However, Dr. Terveen's analysis fails to explain how Raskin's incremental search would actually be implemented in the Lotus Notes search functionality to increase speed, effectiveness, or ease of use. In fact, Dr. Terveen

stated during his deposition that he was not required to consider how incremental search would be implemented, but only that a POSITA would be motivated to use the abstract "idea" of incremental search in Lotus Notes. (EX2021, 116:20-118:6.)

46. This represents a fundamental gap in the technical analysis. You cannot evaluate whether a modification will improve a system without understanding how that modification would actually be implemented and what effects that implementation would have on system performance.

47. Even if one were to ignore all other teachings in Raskin and rely solely on the abstract "idea" of incremental search to be implemented into one of the search tools in Lotus Notes, a POSITA would not be motivated to do so because of the technical limitations in the computing power available to most consumers at the time of the invention. The average computer specifications in the period leading up to and at the time of the invention (2002) were substantially lower than what is available today.[1] As Dr. Terveen himself admitted during his deposition, implementing incremental searching of a user's files with multiple search terms would require a much larger index, which would consume a large proportion of the

[1] For example, just using a back-of-the-envelope calculation based on Moore's law, assuming roughly doubling of performance every 18 months, current computers are over 30,000 times faster today than 23 years ago.

average consumer computer's RAM and hard disk space in 2002. (EX2021, 37:7-39:17.) In conservative estimates, such an index could consume half of the available RAM on a typical consumer computer of that era.

48. To arrive at this rough 50% estimate, we can look at, for example, the minimum system requirements for Microsoft Windows XP, which was released in 2001. In order for a computer to support Windows XP, it needed to have 128 MB RAM and 1.5 GB available hard disk space with 233 MHz CPU. (EX2024, 1.) Now, many users' computers of the time likely had less than 128 MB RAM, given that devices were not required to support Windows XP, especially devices sold before the 2001 release. Because this estimate is intended to be conservative, I will use 128 MB as a generous average. While not strictly necessary, in practice, keeping the index in RAM is desirable for performance for incremental search. So the question is how much RAM the index would occupy. For the purposes of this exercise, I will assume that the amount of disk that held documents was 1 GB. Then, we can use Baeza-Yates's estimate that an index of the time might be around 5% of that 1 GB (EX1023, 193), giving us a rough index size of 50 MB. This is almost half of the (generous) average 128 MB RAM on a typical user's computer at the time of the invention.

49. This level of resource consumption could lead to general performance degradation across the entire computer system, and could even result in search

operations lagging to such an extent that the system would not be able to provide search results as the user types, meaning it would no longer function as an incremental search at all.

50. When I consider the perspective of a POSITA, I assume they would not work in a theoretical vacuum. They would consider the practical constraints of the real-world computing environment in which their system will operate. The hardware limitations of consumer computers in the 2002 timeframe would have been a significant barrier to implementing the type of system Dr. Terveen describes.

51. A POSITA would also understand that implementing Raskin's brute-force incremental search by sequential scanning in the Lotus Notes indexed database search functionality would require significant modifications to the Lotus Notes system. For example, one would need to perform a Raskin-style brute force sequential search of all of the characters in every file in the entire database.

52. The technical problem with this approach is one of scale. A POSITA applying the "incremental search" methodology of Raskin to Lotus Notes' search systems would understand that the sheer volume of material that would need to be sequentially searched would be so large that the search would slow down dramatically, to the point that it would cease to be incremental (providing search results as the user types) at all. Dr. Terveen himself agrees with this assessment, explaining in his declaration for the related '490 patent proceeding (IPR2025-

00254) that "scanning every word of every file sequentially … takes more time and consumes more computing resources" than searching using an index. (EX2023, ¶148.)

53. Alternatively, one might attempt to increase the size of the index in the Lotus Notes system to accommodate all of the potential partial strings included within each word in every document in the database. However, this approach faces its own technical challenges.

54. As I explained in this Section and above, and as Dr. Terveen's own testimony confirms, increasing the size of the index reduces efficiency and increases search times. (*Supra* §IV.A.2; EX2021, 37:7-39:17.) This creates a fundamental problem: not only does an increased search time directly contradict Dr. Terveen's stated motivation of improved speed, but if search times increase to the point where a user can type characters faster than the system can complete the search and return results, the search is no longer incremental by definition.

55. For a search to be truly "incremental" in the sense that Raskin describes (and according to Dr. Terveen's description of incremental search occurring "as the[user] type[s] their query" (EX1003, ¶33)), the system must be able to return updated results faster than the user can type the next character. If the system cannot keep pace with the user's typing speed, the benefit of incremental search—immediate feedback—is lost.

56. Some of the prior art references that Dr. Terveen cites as alleged evidence that incremental search was "well known" at the time of invention also explicitly note the technical drawbacks and challenges of implementing incremental search. (*See* EX1052, 541; EX1044, 3.) Notably, none of these references actually disclose incremental search across a set of documents in the manner claimed. (*Supra* §IV.A.)

57. For example, one reference explicitly states: "As the database grows, it is more difficult to update the display fast enough, and specialized data structures or parallel computation is required." (EX1052, 541.) Another reference states: "This approach requires adequate screen space and high bandwidth, plus, for a large database, very rapid processing: whether it is feasible depends very much on the situation." (EX1044, 3.) These statements from the prior art demonstrate that POSITAs at the time were well aware that incremental search faced significant technical challenges when applied to large datasets. The feasibility of incremental search was understood to be highly dependent on the specific implementation context.

58. Even Dr. Terveen himself recognized these performance considerations in his own published research. In a journal article published in 2007, Dr. Terveen acknowledged that reducing the size of an index would improve performance. (EX2022, 6; EX2021, 103:14-104:6.)

59. Based on the above analysis, in my opinion, a POSITA as of the priority date in 2002 would not have been motivated to implement Raskin's incremental search in the Lotus Notes search functionality because doing so would have required a larger index and made searches slower rather than faster, directly contradicting Dr. Terveen's stated motivation to combine.

60. Perhaps most tellingly, Raskin itself explicitly teaches that despite any potential benefits of incremental searching, "almost all interface-building tools make it easy to implement delimited searches and difficult or impossible to implement incremental searches." (EX1021, 126.) This statement directly undermines Dr. Terveen's suggestion that a POSITA would have been motivated to implement Raskin's incremental search approach in the Lotus Notes system. A POSITA would not be motivated to undertake a modification that the very reference teaching that modification describes as "difficult or impossible to implement." (EX1021, 126.) This teaching in Raskin reflects the technical reality that incremental search, while conceptually appealing, presented significant implementation challenges—particularly when applied to systems designed around different architectural principles, such as the indexed database system disclosed in the Lotus Notes references.

61. Dr. Terveen has failed to identify any prior art reference that actually suggests or discloses incremental search across a database of documents in the

manner claimed. This absence is revealing. If the combination Dr. Terveen proposes was truly obvious and would provide the benefits he claims, one would expect to find at least some examples of such systems in the prior art. The fact that no such systems existed suggests that persons of ordinary skill in the art at the time did not view this combination as obvious or beneficial.

62. Ultimately, it is my opinion that Dr. Terveen's combination does not pass technical muster because it essentially extracts the abstract "idea" of incremental search from Raskin while ignoring everything else Raskin teaches about the context, implementation, and limitations of that approach, and then attempts to apply that decontextualized idea in an unspecified fashion to the Lotus Notes systems. This is simply not how a POSITA would have approached the problem that the '035 patent addresses, and I therefore disagree that a POSITA would have been motivated to combine the Lotus Notes references with Raskin to arrive at the claimed invention.

3. Dr. Terveen's Analysis of Reasonable Expectation of Success is Conclusory and Incorrect

63. Dr. Terveen does not explain why a POSITA would have a reasonable expectation of success in combining Raskin's sequential scanning approach with Lotus Notes' indexed search system. Instead, Dr. Terveen simply asserts that the combination would require "simple and straightforward software modifications." (EX1003, ¶77.)

64. This characterization is technically inadequate and does not reflect the actual complexity of the proposed modification. The search functionalities in the Lotus Notes references and Raskin are materially different in ways that make their combination anything but simple and straightforward. Dr. Terveen relies on Raskin's abstract discussion of the advantages of incremental search, but Raskin provides no meaningful technical details about how incremental search actually works as implemented in EMACS, nor does it provide guidance on how to implement such functionality in other types of systems. (*Supra* §V.A.2.a.)

65. Moreover, Dr. Terveen's analysis ignores Raskin's own admission that it is "difficult or impossible to implement incremental searches." (EX1021, 126.) During his deposition, Dr. Terveen testified that he "did not have to talk about implementation level details or incorporating a particular implementation," adding that this "wasn't necessary for [his] opinion." (EX2021, 116:20-117:17.) Dr. Terveen's combination simply takes the abstract "idea of incremental search" from Raskin without grappling with the implementation hurdles that Raskin itself identified.[2]

---

[2] As the apocryphal story about the absent-minded genius math professor goes: "During a lecture, G. H. Hardy wrote a theorem on the board and declared, "The proof is obvious." He paused, frowned, and muttered, "Or is it?" He then left the

66. As I explained above, Raskin discloses a search system within the EMACS text editor that searches for character strings within a single document that has been loaded into the active buffer. (*Supra* §IV.A.1; *see* EX1021, 125-26.) This is not a system for searching for and locating documents, as required by the claims. In contrast, the Lotus Notes references describe an indexed search system designed to search for strings across a collection of documents. (*See* EX1059, 169-70; EX1061, 138-39; *supra* §V.A.2.b.) Dr. Terveen's declaration provides no explanation of how a system designed for primitive, brute force searching (i.e., sequential scanning of characters in an active buffer) could be modified to search for and locate documents within an indexed file system. (*Supra* §V.A.2.) These represent fundamentally different use cases and system architectures:

67. The incremental search functionality in Raskin as implemented in GNU EMACS does not use an index but instead sequentially scans the text of an open document. (EX1022, 55.) As I explain above, such a search implementation makes no technical sense in a file system containing large numbers of documents because it would be far too slow to be usable. (*Supra* §V.A.2.)

---

room. After about fifteen minutes, he returned, looked at the board, and announced with satisfaction, "Yes, it is obvious!" before continuing the lecture." If even such a luminary as Hardy had to think for 15 minutes, it was likely -not- obvious.

68. This performance problem is precisely the issue with prior art search systems that the ’035 patent identifies and that the inventors sought to address. (*See* EX1001, 1:32-2:5.) In my opinion, implementing Raskin’s incremental search functionality using real-time sequential scanning into Lotus Notes’ indexed database system would not be straightforward because the two search systems are fundamentally different and not amenable to combination in the manner Dr. Terveen proposes.

69. Finally, if coding the desired software functionality were as straightforward as Dr. Terveen suggests (*e.g.*, EX1003, ¶89), one would expect that incremental search across multiple documents to locate documents using two search strings in a single search field would have been implemented prior to the priority date. This is especially confusing given the publication dates of the references: Raskin was published in 2000, Wilcox was published in 1993, Londergan was published in 1999, and Wu was filed in 1997 and granted in 1999. All these references existed almost three years before the September 3, 2002 priority date of the ’035 patent.

70. Raskin would not characterize implementing incremental search as “difficult or impossible” if it were truly straightforward. (EX1021, 126.) Dr. Terveen has not identified any references disclosing such search functionality. Indeed, both of Dr. Terveen’s proposed obviousness grounds rely on a combination

of a minimum of four different references to allegedly achieve this "simple and straightforward" functionality. (EX1003, ¶89.)

71. The need to combine four or more references to arrive at the claimed invention is itself an indicator that the combination is not as straightforward as Dr. Terveen suggests. Each additional reference required in the combination represents another conceptual and technical leap that a POSITA would need to make, and another potential source of incompatibility or implementation difficulty. Therefore, I disagree with Dr. Terveen that a POSITA would have a reasonable expectation of success in combining the Lotus Notes references with Raskin and Wu.

B. The Lotus Notes-Raskin-Wu Combination Does not Render Obvious All Limitations of the Challenged Claims

1. Elements [1.4] and [10.4]

72. Claims 1 and 10 each require "*receiving a second string in the first search field, wherein the first and second strings are separated by a string separator character in the first search field*." Dr. Terveen's proposed combination of Lotus Notes, Raskin, and Wu does not adequately account for this claim limitation when the term "string separator character" is properly understood.

73. Dr. Terveen initially points to the Lotus Notes Search bar functionality, which he contends meets this limitation. (EX1003, ¶81.) However, the Search bar functionality requires that multiple search strings be separated by a Boolean AND operator. (EX1059, 162; EX1061, 140-41.) I understand the panel determined in the

Institution Decision that a Boolean AND operator is not a string separator character. (Paper 14, 28-29.) As I explained in my First Declaration, Boolean operators like AND are logical operators that define a relationship between search terms and instruct the search engine on how to process those terms, i.e., a search syntax instruction. (EX2001, ¶109; *see* EX1059, 162 (discussing the use of "logical operators AND, OR, NOT, and ACCRUE to find documents").) A string separator character, by contrast, is a delimiter that simply separates distinct character strings without imposing any logical relationship or search instruction. The Boolean AND operator in the Lotus Notes Search bar performs a fundamentally different function than a string separator character.

74. Dr. Terveen alternatively points to two other Lotus Notes functionalities: the Query Builder and the Search Builder. (EX1003, ¶82.) He asserts that these functionalities receive two strings separated by a comma, which he contends is a string separator character. (EX1003, ¶82.) However, this characterization is not accurate for the Search Builder functionality. The Search Builder interface only accepts a single search string in each individual field. (EX1061, 142.) The interface provides multiple separate fields, but each field is designed to receive only one search string. (EX1061, 142.) Because each field accepts only a single search string, there is no separation of multiple strings within

a single field, and therefore no string separator character is present in the Search Builder functionality as Dr. Terveen suggests.

75. With respect to the Query Builder functionality, Dr. Terveen's reliance on this feature is, in my opinion, insufficient to demonstrate that the claimed limitation would have been obvious. As I explained above, Dr. Terveen has not provided an adequate technical explanation of how the various different search functionalities disclosed across the Lotus Notes references—including the Search bar, the Query Builder, the Search Builder, and Form-based search—would be combined together in a single system that meets all of the limitations of the challenged claims. (*Supra* §V.A.1.) Each of these search functionalities operates differently and serves different purposes within the Lotus Notes system. Simply pointing to individual features across multiple different search tools, without explaining how those features would be integrated into a cohesive system, does not establish that the claimed invention would have been obvious to a POSITA.

2. Elements [1.3], [1.5], [10.3], [10.5]

76. Claims 1 and 10 each require "*incrementally locating…a group of documents*." In my opinion, Dr. Terveen's proposed combination of Lotus Notes, Raskin, and Wu does not render this limitation obvious.

77. Dr. Terveen acknowledges in his declaration that the Lotus Notes references do not disclose any incremental locating functionality. (EX1003, ¶83.)

To address this gap, Dr. Terveen relies on Raskin. (EX1003, ¶¶83-86.) However, this reliance is problematic for several reasons.

78. The specific portion of Raskin that Dr. Terveen cites describes the incremental search functionality implemented in the GNU EMACS text editor. (EX1003, ¶¶83-86 (citing EX1021, 125-27).) As I explained above, this functionality operates by sequentially scanning through the text of a single document that is currently open in the editor's active buffer. (EX1022, 55.) The user initiates the search by pressing Control-s and then typing characters, and EMACS scans through the open document character-by-character to find matches. (EX1022, 55.)

79. This is different from locating "a group of documents" to locate a "first group of documents" and a "second group of documents" as required by the claims. Raskin's disclosure concerns searching within a single open text document, not searching across multiple documents to identify and locate specific documents that match search criteria. These are distinct operations that serve different purposes and require different system architectures. Searching within a single open document involves scanning text that is already loaded into the computer's active memory (the EMACS "buffer"). (EX1022, 55.) The system knows the boundaries of the search space (the beginning and end of the document), and the search operation is simply a matter of examining the characters in sequence until a match is found or the end

of the document is reached. (*See* EX1022, 55-57; EX1023, 191 ("An obvious option in searching for a basic query is to scan the text sequentially. Sequential or online text searching involves finding the occurrences of a pattern in a text when the text is not preprocessed.").)

80. In contrast, searching across a set of documents to locate specific documents requires the system to first identify which documents exist in the set, access those documents (which may not be loaded into active memory), search their contents, determine which documents contain matches to the search criteria, and present those matching documents to the user. (EX2001, ¶58; *see also* EX1023, 9, 192.) This is a more complex operation that involves interacting with the file system to identify available documents, retrieving those documents from storage, and collecting the results together, none of which are present in Raskin's disclosure of searching within a single open document.

81. As I explained above, Dr. Terveen has not provided an adequate technical explanation of why a POSITA would have been motivated to combine Raskin's teaching about sequential scanning of a single text document with the Lotus Notes indexed search system that is designed to search across multiple documents. (*Supra* §V.A.2.) These represent meaningfully different search architectures—one based on real-time sequential scanning of text in active memory, and the other based on pre-computed indexes of document collections.

82. Furthermore, as I explained above, Dr. Terveen has not explained how such a combination would actually be implemented from a technical standpoint. (*Supra* §V.A.2.) How would one adapt a system designed for character-by-character scanning of an open document to instead search across a collection of documents in a file system or database? What modifications would be required to the Lotus Notes architecture? How would the index be used, if at all? These implementation questions are not addressed in Dr. Terveen's analysis.

83. Dr. Terveen also has not established that a POSITA would have had a reasonable expectation of success in implementing such a combination. As I discussed above, the technical challenges involved in adapting Raskin's sequential scanning approach to work across multiple documents would be substantial. (*Supra* §V.A.3.) The performance issues alone would make such an approach impractical for the intended use case. A POSITA would recognize these challenges and would not have a reasonable expectation that the combination would successfully achieve the desired functionality.

84. For these reasons, it is my opinion that Dr. Terveen has not established that the prior art renders obvious "*incrementally locating…a group of documents*."

3. Claims 2-9

85. In my opinion, Petitioner has not met its burden to establish obviousness of claims 2-9 for at least the same reasons I explained above for claim

1, including a failure to establish motivation to combine and reasonable expectation of success. (*Supra* §§V.B.1-2.)

4. Claims 2-4

86. Claims 2-4 each require "*displaying a portion of a selected document*." In my opinion, the Lotus Notes-Raskin-Wu combination does not render this claim obvious.

87. The '035 patent teaches that the user interface includes "a list area or pane 324A, which includes a list of the search results" and "a view area or pane 326A is provided that displays the content of the first item or user-selected item from the list pane or area 324A." (EX1001, 14:21-36.) This is illustrated in Figure 3A:

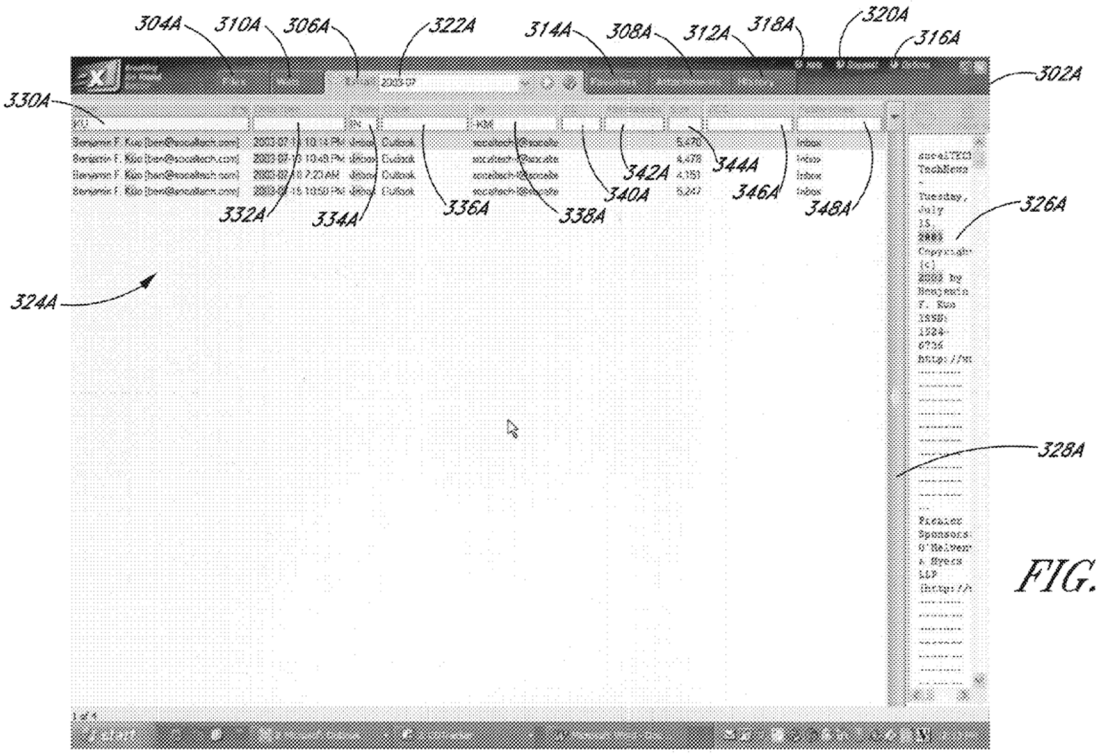


FIG. 3A

’035 Patent Figure 3A (EX1001)

88. The ’035 patent explains that clicking on a list pane entry causes the content of the document to be displayed in the view pane. This is illustrated in, for example, Figures 3C and 3D:

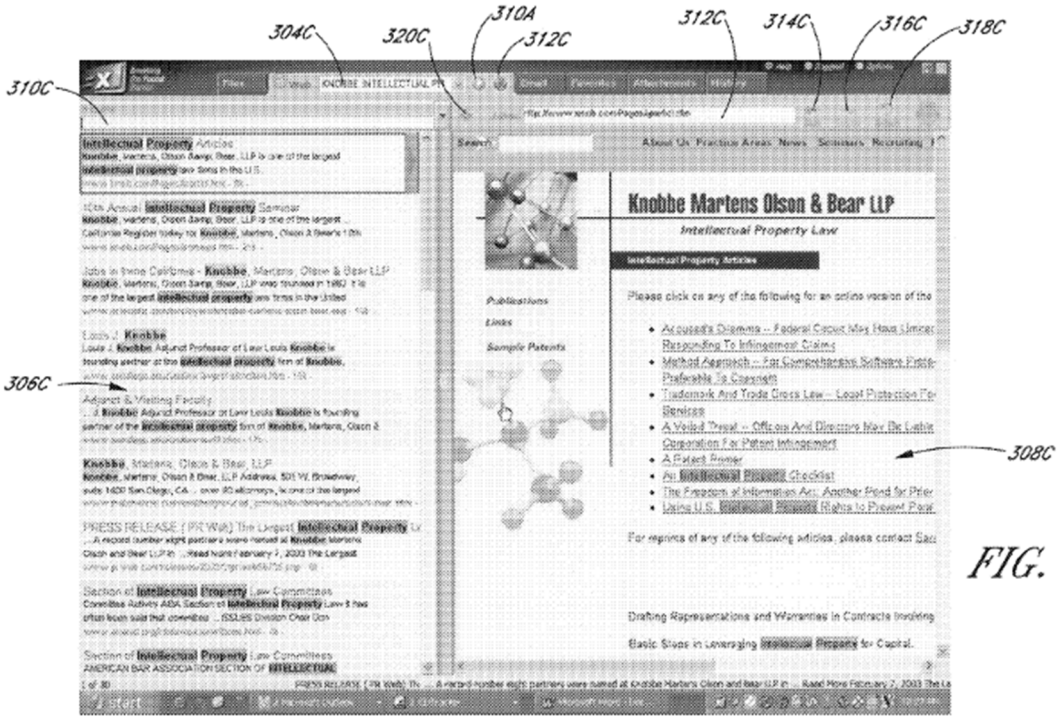


FIG. 3C

’035 Patent Figure 3C

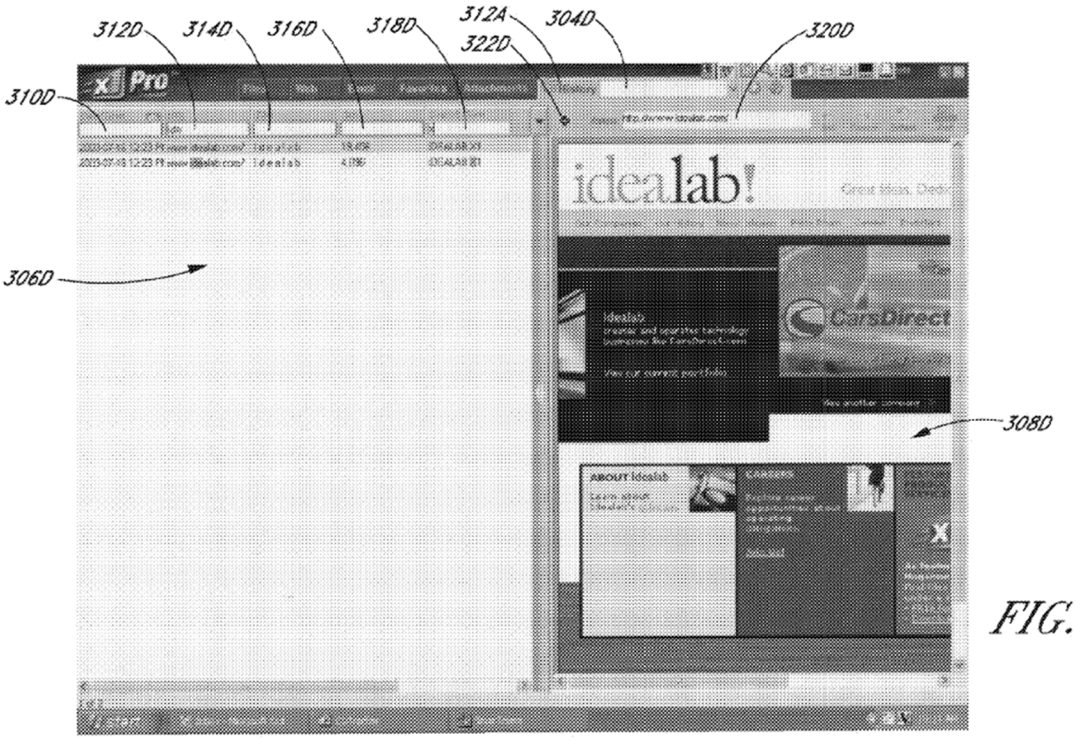


FIG. 3D

<u>'035 Patent Figure 3D (EX1001)</u>

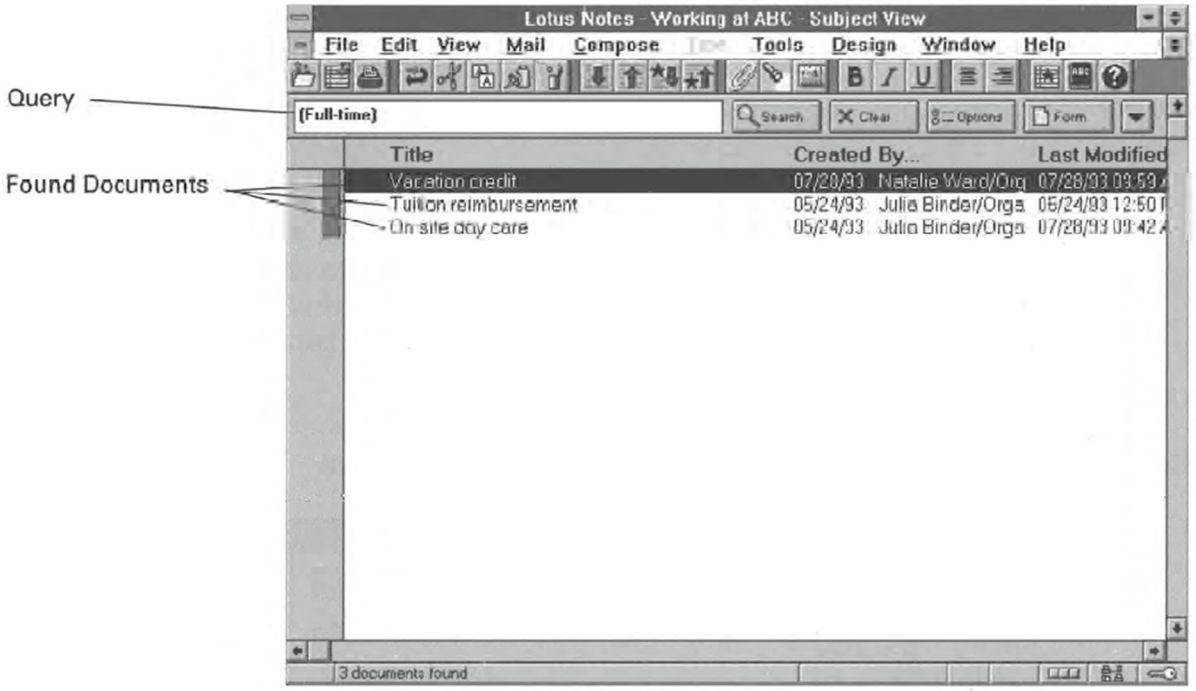
(*See also* EX1001, Fig. 3E; 13:60-14:9;16:40-58; 17:20-30; 19:48-64.) The '035 patent further teaches that a user can choose to open the document itself "in its native application" by double-clicking on it. (EX1001, 14:37-48.)

89. Petitioner and Dr. Terveen rely only on the disclosures of the Lotus Notes references plus the alleged knowledge of a POSITA to meet this limitation. In fact, Dr. Terveen misleadingly states that "the Lotus Notes+Raskin+Wu combination renders this claim obvious at least because the Lotus Notes references expressly disclose displaying a portion of a selected document of the documents containing the first and second search strings. (EX1003, ¶92.) I disagree. In my

opinion, the Lotus Notes do not disclose or render obvious "*displaying via the user interface a portion of a selected identified document*."

90. The Lotus Notes references make clear that in order to view any portion of an identified document, the user must first open the document itself. For example, Wilcox instructs the user to "[o]pen the first document listed in the view." (EX1059, 164.) Similarly, Londergan explains that "[y]ou open a document uncovered by a search the same way you would open any other document. ***Double-click it or select it with your mouse and press Enter. Notes then shows you the document*** and highlights the text that you wanted to find." (EX1061, 145-46.)

Figure 5.17 The results of a query



7. Open the first document listed in the view. The found text *Full time* is indicated by a red box. The hyphen wildcard also finds text strings separated by a space.

Wilcox, Figure 5.17 (EX1059, 164)

91. Dr. Terveen admitted this during his deposition, testifying that "the express disclosure in Wilcox" is that "in order to view the document and see the highlighting or the red box you have to actually open the document." (EX2021, 61:18-62:2.) He also admitted that Londergan discloses that the user must open the document in order to view any portion of it. (EX2021, 67:19-68:2 ("that is the director [sic] express disclosure in Londergan.").)

92. In my opinion, opening a document in a separate window does not disclose or render obvious "*displaying a portion of a selected document*" as required

by claims 2-4. As I explained above, the ’035 patent contemplates a unified user interface that includes a list pane, where search results are displayed, and a view pane, where a portion of the selected located document is displayed. (The list and view panes are both part of the user interface itself. Therefore, it is my opinion that this limitation is not expressly disclosed in the Lotus Notes references.

93. I also disagree with Dr. Terveen’s opinions regarding claims 3 and 4. These claims require “*highlighting a first occurrence of the first word in the selected document*” and “*highlighting at the same time a plurality of occurrences of the first word in the selected document while not highlighting portions of text between occurrences of the first word*,” respectively. (EX1001, claims 3-4.) Because they depend from claim 2, the highlighting must occur in the portion “*display[ed]*.” But as I explained above, Lotus Notes requires opening documents to view highlighting, it is “*display[ed]*.”

5. Claims 7 and 9

94. Claims 7 and 9 specifically requires the ability to search on particular email fields, such as the “To,” “From,” or “Subject” fields of an email message. In my opinion, the Lotus Notes-Raskin-Wu combination does not render this claim obvious.

95. Dr. Terveen’s analysis of this limitation relies on combining separate and disconnected disclosures from the Lotus Notes references without adequately

explaining how these disclosures would be integrated or why a POSITA would make such a combination. Dr. Terveen first points to the fact that some documents stored in the Lotus Notes environment may be email messages. (EX1003, ¶100.) This is a general statement about the types of documents that can exist within Lotus Notes, but it does not disclose any specific functionality for searching email fields.

96. Dr. Terveen then separately points to a different feature in Lotus Notes called the "form" functionality. (EX1003, ¶100.) This feature allows users to search within specific fields of documents that were created using a particular form template. (EX1059, 167-68; EX1061, 142-44, 146-47.) However, the Lotus Notes references contain no teaching or disclosure that this searching by "form" functionality can be used to search specific email fields.

97. These are two separate disclosures: (1) that emails can exist in Lotus Notes, and (2) that there is a forms-based search feature. Dr. Terveen appears to be combining these two separate pieces of information to suggest that one could use the forms functionality to search email fields, but this combination is not disclosed or suggested anywhere in the references. If Dr. Terveen intends to propose such a combination, he has not explained the technical details of how it would work, why a POSITA would be motivated to make this combination, or why such a person would have a reasonable expectation of success in implementing it.

98. From a technical perspective, the "searching by Forms" functionality is an entirely separate search tool from the various search functionalities that Dr. Terveen relies upon in his analysis of claim 1—namely, the Search bar, Search Builder, and Query Builder functionalities. The searching by Forms feature is specifically limited to searching for documents that were created using a specific form template. (EX1059, 167; EX 1061, 143.)

99. Importantly, there is no disclosure in the Lotus Notes references of any ability to search on multiple strings entered into a single search field using the searching by Forms functionality. There is also no disclosure of incrementally searching for documents using this functionality. These are critical limitations required by the claims that are simply absent from the Forms-based search feature.

100. Because Dr. Terveen did not rely on the searching by Forms functionality in his analysis of claim 1, he never explains how this functionality would be incorporated into his proposed Wilcox-Londergan combination, or into the broader Lotus Notes-Raskin-Wu combination. This creates a significant gap in the analysis. If Dr. Terveen is proposing a combination that includes the searching by Forms functionality along with the other search tools, he has not provided an adequate motivation to combine these disparate features, nor has he established that a POSITA would have a reasonable expectation of success in creating such a combination. As I explained above, combining multiple different search

functionalities from the Lotus Notes references presents significant technical challenges, and simply pointing to individual features without explaining how they would work together does not automatically mean a POSITA would have combined those features. (*Supra* §V.A.1.)

101. Therefore, I disagree with Dr. Terveen that claim 8 is obvious over the Lotus Notes-Raskin-Wu combination.

6. Claim 8

102. This dependent claim requires searching on specific fields. In my opinion, the Lotus Notes-Raskin-Wu combination does not render these claims obvious.

103. Claim 8 requires the ability to search on specific fields within documents. Dr. Terveen relies solely on the Lotus Notes references for these claims but fails to identify any disclosures in these references that adequately demonstrate how these claim limitations would have been obvious. (EX1003, ¶¶101-01.) As with claim 8, Dr. Terveen points only to the searching by Forms functionality as allegedly allowing searching within specific fields of a document. (EX1003, ¶¶101-01.) However, Dr. Terveen's analysis suffers from the same technical deficiencies I identified with respect to claim 8 (*supra* §V.B.5): he does not explain how the "form" functionality would be integrated with the other search functionalities he relies upon for claim 1, he does not provide a motivation for combining these

separate features, and he does not establish that a person of ordinary skill in the art would have had a reasonable expectation of success in implementing such a combination.

VI. GROUND 2: THE CHALLENGED CLAIMS ARE PATENTABLE OVER THE COMBINATION OF SCHWARTZ (EX1004), SEBASTIAN (EX1005), BARBER (EX1006), TRUE (EX1009), AND BAEZA-YATES (EX1023).

A. Dr. Terveen Has Not Provided Motivation to Combine the Ground 2 References

1. Dr. Terveen Has Not Provided Motivation to Combine the Elements from Different Search Systems in the Entourage References

104. In addition to my discussion in my First Declaration about Dr. Terveen's alleged motivation to combine the teachings of the Lotus Notes references, I further offer my opinion that the fact that the developers of Entourage chose to keep these functionalities separate—maintaining distinct incremental list filtering and separate delimited Find/Advanced Find tools—would suggest to a POSITA that doing so was for good reason. Software designers make deliberate architectural choices, and when a product maintains two separate search mechanisms rather than combining them, it typically reflects technical constraints or usability considerations that make combination undesirable or impractical.

105. In Entourage's case, the separation appears driven by the performance issues I discussed earlier. (*Supra* §§IV.A.2, V.A.2.a.) The Entourage references

explicitly describe the Find and Advanced Find features as being slower, particularly when searching across broader document sets. (EX1006, 384-86 (if you search "All Items," you should "be prepared to rent a video during the time Entourage will take the search its entire multimegabyte database."); EX1004, 9.) Maintaining the incremental list filtering as a separate, lightweight feature would have allowed users to quickly narrow down visible items without triggering the more resource-intensive indexed search operations required by Find and Advanced Find. Combining these approaches likely would have required either maintaining substantially larger indexes or performing real-time sequential scanning, either of which would exacerbate the existing performance problems and degrade the user experience. (*Supra* §§IV.A.2, V.A.2.a.)

106. This same reasoning applies to another of Dr. Terveen's opinions: that the Entourage references "at least suggest receiving two search strings … in the search field separated by a string separator character." (EX1003, ¶127.) In my opinion, a POSITA would interpret Find's multiple boxes as an indication that the developers intended that each box would receive one string, not multiple.

2. True is Not Analogous Art to the '035 Patent

107. Dr. Terveen relies on True to satisfy the missing element in the Entourage references of "*receiving two partial search strings in a single search*

field." (Pet., 105.) However, in my opinion, a POSITA would not understand True to be analogous to the '035 patent.

a. True Relates to Graphical Representations and User Navigation of Audio File Transcripts

108. True is a system for visualizing the contents of a single underlying file, such as an audio file, using a text transcription of that file that is inherently imprecise due to transcription errors. (*See* EX1009, 1:18-27, 1:38-45, 6:38-44 ("Words whose confidence levels fall below a predetermined threshold may be displayed as ellipses (" ... ") in the transcript window.") In the system, a user can navigate throughout the audio file that corresponds to a given part of the transcribed text, edit the transcription when errors are detected, and locate "material of interest" in the text file and therefore the underlying audio file. (EX1009, 1:38-45.)

109. While True refers to "searching" the transcript file, the specification makes clear that the purpose of True is not to definitively locate where a search string actually appears in the text file. Instead, it uses "confidence levels" to show a user how likely it is that the portion of the "transcript window" of the underlying audio file actually contains a given word in the text file (instead of being an incorrectly recognized word):

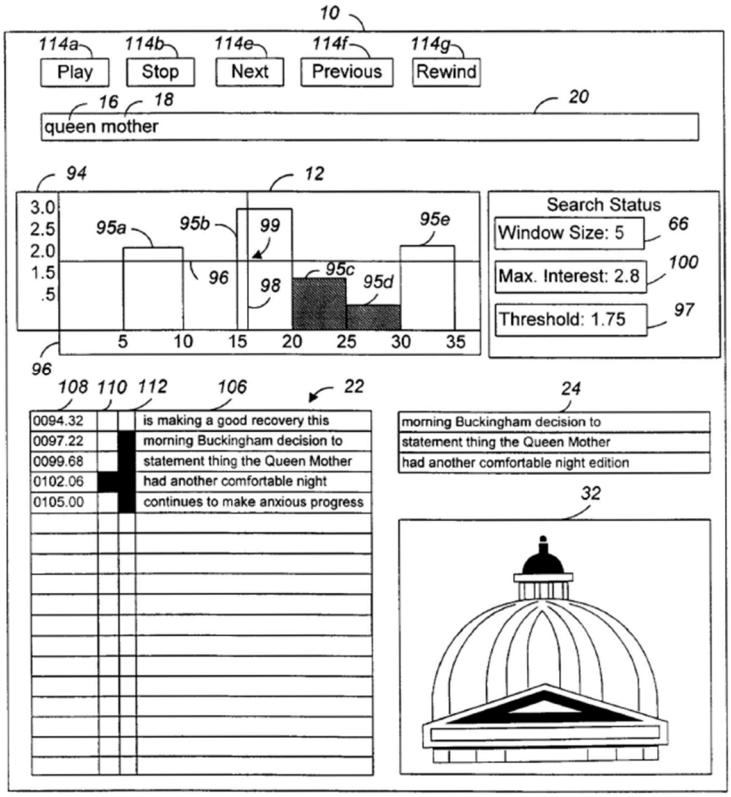
> A recognized word's confidence level is a number between zero and one indicating an estimated likelihood that the recognized word is correct. For example, the recognized word "is" has a confidence level of 0.63, indicating a 63% likelihood that "is" corresponds to the portion

> of the digitized audio 26 with a start time of 3.72 seconds and an end time of 3.77 seconds.

(EX1009, 4:34-38.) As the above excerpt demonstrates, each recognized word in the transcript includes a confidence level ("a number between zero and one indicating an estimated likelihood that the recognized word is correct." (EX1009, 4:21-38.) Based on those confidence levels, True is able to show a user, based on a specified time length of a segment of the transcript ("transcript window"), such as five-seconds, "which segments of the transcript relate to topics of interest" by summing the confidence levels of each word in a given time period of the audio file. (EX1009, 2:10-16, 3:40-43, 4:55-65.)

110. True's query results consist of a graphical display showing the "interest level" of "transcript windows," as shown in Figure 2, reproduced below.

FIG. 2



True, Figure 2 (EX1009)

111. It is possible for a transcript window to be highlighted as relevant even though the visible transcript text does not show the query word. True explains that all confidence scores in the segment, even low-confidence recognized occurrences of the word, are counted toward the interest score. (EX1009, 5:40-42 ("A [transcript] window's interest level may be calculated, for example, as the sum of the confidences of each word in the transcript window matching a word in the query.").) However, "[w]ords whose confidence levels fall below a predetermined threshold may be displayed as ellipses (' ... ') in the transcript window." (EX1009,

6:38-44.) So, if multiple low-confidence occurrences of a search word appear close together, the transcript window might display a "relevant" section of the transcript that is made entirely of ellipses. True's bar chart in the result window, therefore, provides a visual indication of which segments of the transcript (transcript windows) might contain at least one of the words in the search query. (EX1009, 6:9-11.) True does not provide a list of search results (let alone documents) pointing to where all the instances of the words in the search query appear. (EX2001, ¶130.) Thus, True is not searching and locating documents or groups of documents that correspond to two partial search strings, as required by the claims. True's search is also done sequentially by scanning each character in the text of the transcript for the query words. (*See* EX1009, 4:21-49.)

b. True is Not in the Same Field of Endeavor

112. Dr. Terveen asserts that True is analogous because it is in the same "field of endeavor": "providing a user interface through which a user can enter search strings, and locating documents and/or portions of documents containing the search strings." (EX1003, ¶117.) I disagree. Dr. Terveen's characterization of the field of endeavor is based on a fundamental misunderstanding of the claimed invention. In my opinion, a POSITA would understand that the field of endeavor for the claimed invention is not 'methods and systems for entering search strings in search fields'; it is directed to incrementally searching sets of indexed documents

that correspond to the partial search strings in the search field. All claims of the patent require that a "first set of documents" and a "second set of documents" corresponding to the search strings are identified, and the title of the patent is "Methods and systems for search indexing." (EX1001, Title.)

113. The fact that True and the '035 patent both relate to search at the highest level is largely irrelevant in my opinion. Search is an incredibly broad field that no POSITA, as defined by both myself and Dr. Terveen, could be familiar with in its entirety. (See EX2001, ¶¶29-31; EX1003, ¶¶30-31.) Thus, in my opinion, Dr. Terveen takes an abstracted view of the field of endeavor that is not reflected in the technical reality of the POSITA at the time of the invention. It should be undisputed that True does not search documents, groups of documents, does not search indexed groups of documents, and in fact does not even necessarily return results that would have the query word(s) in them. As such, I do not believe that True is in the same field of endeavor as the '035 patent.

c. True is not Reasonably Pertinent to the Problem to be Solved by the '139 Patent

114. In my opinion, a POSITA would not have looked to True in considering the problem that the '035 patent addresses. The inventors of the '035 patent sought to invent systems and methods for quickly searching groups of documents, solving the lack of speed and efficiency in sequential scan techniques and delimited, "search builder" type user interfaces of the prior art that were too

slow or inaccurate for multi-document search. (EX1001, 1:32-2:5.) The search techniques of True appear to have the exact problems that the ’035 patent seeks to improve upon—these references disclose sequential, slow, and inaccurate search. (*See* EX1009, 4:21-49, 6:37-44.) True requires a sequential scan of the entire transcript (*see* EX1009, 4:21-49), only works in a single document and not across groups of documents (*see* EX1009, 3:19-21), and when it returns a search result, that does not necessarily even mean the term is in the search result (EX1009, 5:40-42, 6:38-44). (*See* EX2001, ¶¶129-40.) Thus, instead of True providing a solution to the problem faced by the inventors, it was, if anything, a step backwards. It also utilized sequential search that would have been unsuitable for multi-document incremental search because it would be too slow. (*See* EX1009, 4:21-49.)

115. Dr. Terveen also overlooks the fact that just because prior art relates in a general sense to the problem sought to be solved by the claimed invention, that does not mean it is reasonably pertinent to that specific problem. In my opinion, a POSITA would not have looked to True to solve the problem addressed by the ’139 patent—slow, sequential searching of large document sets. (*See* EX1001, 1:32-2:5.) From a technical perspective, True addresses an entirely different use case. True is designed to help users navigate and edit imprecise transcriptions of audio files by providing probabilistic indicators of where words might appear in the underlying audio. (*See supra*.) The ’035 patent, by contrast, is designed to help users quickly

and accurately locate specific documents within large collections of indexed documents. (*See, e.g.*, EX1001, 2:15-27.) These are different technical problems requiring different solutions. In my opinion, a POSITA working on the problem of searching large document collections would not look to a system designed for navigating imprecise transcriptions of single audio files, as the technical approaches and constraints are entirely different.

3. Dr. Terveen Has Not Provided Motivation to Combine the Entourage References with True

116. In my First Declaration, I explained that Dr. Terveen has not provided a motivation to combine True's multiple strings with the Entourage references. (EX2001, ¶¶128-140.) Particularly, I explained that:

> Further, a POSITA would not even think about using True's incremental search that requires performing a search every time a character is entered given the fact that searching through all files and text in Entourage is known to take a long time even with the Find and Advanced Find tools. (EX1006, 385 ("you can choose All Items–but be prepared to rent a video during the time Entourage will take to search its entire multimegabyte database.") Dr. Terveen's analysis did not include any suggestions why a POSITA would consider True's incremental search in such conditions or any explanation as to how a POSITA would combine the references to avoid such major obstacle.

(EX2001, ¶140.) To further elaborate on this point, a POSITA would not be motivated to implement True's incremental search in the Find and Advanced Find search tools of the Entourage references for the same technical reasons that they

would not be motivated to combine the teachings of Raskin with the Lotus Notes references. (*See supra* §V.A.2.)

117. As I explained above regarding the Lotus Notes-Raskin combination, implementing incremental search in a multi-document search system would require either an increased index size, which would slow down searching or exceed the available computing power of a typical consumer computer at the time of the invention, or would require sequential search across all documents, which would be so slow as to no longer function as incremental search at all. (*See supra* §V.A.2.)

4. Dr. Terveen's Analysis of Reasonable Expectation of Success is Conclusory and Incorrect

118. As I explained in my First Declaration and above, the specific incremental list filtering functionality from Entourage and the incremental "search" disclosed in True are fundamentally different from, and incompatible with, the delimited searches that can be performed using the Find and Advanced Find tools. (EX2001, ¶¶134-41; *supra* §VI.A.2.) The Entourage references repeatedly characterize the delimited search tools (Find and Advanced Find) as being slower and, if the search is not limited to a narrower set of documents, can take up to hours to receive results. (EX1006, 385 ("you can choose All Items–but be prepared to rent a video during the time Entourage will take to search its entire multimegabyte database.") Both Dr. Terveen and I agree that implementing incremental search requires a much larger index, which can further slow down the search process, even

to the point where the search results are no longer returned in a timely manner and therefore the search is no longer responsively 'incremental.' (*Supra* §V.A.2; EX.2021, 49:13-52:19.) Thus, in my opinion, a POSITA would not expect incremental search in the Find or Advanced Find features of the Entourage references to yield a successful, workable search system due to speed and efficiency concerns.

B. The Entourage-True-Baeza-Yates Combination Does Not Render Obvious All Limitations of the Challenged Claims

1. Claim 3-4

119. Dr. Terveen has not established that the Entourage-True-Baeza-Yates combination discloses "*automatically highlighting a first occurrence of the first word in the selected document*" or "*highlighting at the same time a plurality of occurrences of the first word in the selected document while not highlighting portions of text between occurrences of the first word* as is required by Claims 3 and 4, respectively.

120. Dr. Terveen relies solely on the Entourage references to disclose the highlighting limitation of claims 3-4 (EX1003, ¶¶153-57), citing only to Schwartz's disclosure of using the Find and Advanced Find command functionality to search for a single string within a single Entourage item. (EX1003, ¶171; *see also* EX1004, 10-11 (describing searching the "current item" using the "Find" feature), 162-63 (describing searching the "current message" using the Advanced Find feature).)

However, any highlighting in Entourage occurs only in an open document during a search of that open document. The highlighting is not shown in a portion of the document displayed via the user interface. (EX1004, 11 ("Entourage highlights the first match, scrolling as necessary to display it.").)

121. Furthermore, Dr. Terveen acknowledges that the Entourage-True-Baeza-Yates combination does not disclose "highlighting" of the search strings, and instead relies on what he characterizes as the knowledge of a person of ordinary skill in the art to fill in the missing limitation, stating that "it would have been obvious to a POSITA … since this targeted search ability and its benefits were known in the art." (EX1003, ¶¶172, 191; EX2021, 160:161:9 (Dr. Terveen testifying that he relied on the knowledge of a POSITA to fill this gap).) However, Dr. Terveen has not pointed to any specific prior art reference, technical publication, or other concrete evidence demonstrating that highlighting a plurality of occurrences in the manner claimed was part of the common knowledge in the art at the relevant time. This represents a gap in Dr. Terveen's technical analysis of claims 5 and 26.

VII. CONCLUSION

122. Based on the above, it is my opinion that none of the '035 patent's challenged claims are invalid in view of the prior art asserted by Dr. Terveen and Microsoft

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed in Redwood City, California.

Dated: November 3, 2025

Martin P. Haeberli

PETITIONER'S EXHIBIT LIST

Exhibit No.	Description of Document
1001	U.S. Patent No. 7,370,035 ("'035 patent")
1002	File History of U.S. Patent No. 7,370,035
1003	Declaration of Dr. Loren Terveen, dated December 19, 2024
1004	Steve Schwartz, Visual QuickStart Guide Entourage 2001 for Macintosh, Peachpit Press (2001) ("Entourage-1")
1005	Bonita Sebastian et al., Microsoft Office 2001 A Hands-On Guide for Macintosh, Computer Literacy Press (excerpts) (2000) ("Entourage-2")
1006	Nan Barber & David Reynolds, Office 2001 for Macintosh: The Missing Manual, Pogue Press/O'Reilly (1st ed.) (Apr. 2001) ("Entourage-3")
1007	Eudora Email User Manual for Windows Version 5.0, QUALCOMM (Aug. 2000) ("Eudora")
1008	Alan R. Neibauer, Running Microsoft Outlook 2000, Microsoft Press (excerpts) (1999) ("Outlook")
1009	U.S. Patent No. 6,112,172 to True et al. ("True")
1010	U.S. Patent No. 5,444,823 to Trung D. Nguyen ("Nguyen")
1011	U.S. Patent No. 7,039,635 to Morgan et al. ("Morgan")
1012	U.S. Patent Application Publication No. 2002/0055981 to Spaey et al. ("Spaey")

Exhibit No.	Description of Document
1013	U.S. Patent No. 5,845,278 to Kirsch et al. (“Kirsch”)
1014	U.S. Patent No. 4,255,796 to Gabbe et al. (“Gabbe”)
1015	U.S. Patent No. 5,692,173 to Chee Heng Chew (“Chew”)
1016	U.S. Patent No. 6,049,796 to Siitonen et al. (“Siitonen”)
1017	U.S. Patent No. 5,303,361 to Colwell et al. (“Colwell”)
1018	U.S. Patent No. 6,741,985 to Robin A. R. Green (“Green”)
1019	U.S. Patent No. 5,991,756 to Jiong Wu (“Wu”)
1020	Internation Patent Application Publication No. WO 00/26762 to Mark Richanbach (“Richanbach”)
1021	Jef Raskin, The Humane Interface: New Direction for Designing Interactive Systems, ACM Press (2000) (“Raskin”)
1022	Richard Stallman, GNU Emacs Manual (1st ed.) (Jun. 1985) (“GNU Emacs”)
1023	Ricardo Baeza-Yates & Berthier Ribeiro-Neto, Modern Information Retrieval, ACM Press (1999) (“Baeza-Yates”)

Exhibit No.	Description of Document
1024	Sunny Lam, The Overview of Web Search Engines (Feb. 9, 2001) ("Lam")
1025	Oscar Vallarino, On The Use Of Bit Maps For Multiple Key Retrieval, Proceedings of the 1976 Conference on Data: Abstraction, Definition And Structure, 108-144 (Mar. 1976) ("Vallarino")
1026	Steve Lawrence & C. Lee Giles, Context And Page Analysis For Improved Web Search, IEEE Internet Computing, 38-46 (Jul.-Aug. 1998) ("Lawrence")
1027	Rawley D. Johnson et al., Technical Reference Document Database System (TDOCS) Requirements Definition (Aug. 1993) ("Johnson")
1028	Lev Finkelstein et al., Placing Search in Context: The Concept Revisited, ACM Transactions on Information Sys., Vol. 20, No. 1, 116-131 (Jan. 2002) ("Finkelstein")
1029	U.S. Patent No. 6,952,805 to Tafoya et al. ("Tafoya")
1030	U.S. Patent No. 6,009,459 to Belfiore et al. ("Belfiore")
1031	U.S. Patent No. 5,220,625 to Hatakeyama et al. ("Hatakeyama")
1032	U.S. Patent No. 6,185,551 to Birrell et al. ("Birrell")
1033	U.S. Patent No. 6,839,702 to Patel et al. ("Patel")
1034	U.S. Patent No. 6,973,450 to Silverbrook et al. ("Silverbrook")
1035	U.S. Patent No. 6,834,276 to Jensen et al. ("Jensen")

Exhibit No.	Description of Document
1036	U.S. Patent No. 5,946,678 to Ijsbrand Jan Aalbersberg ("Aalbersberg")
1037	European Patent Application Publication No. EP 1 209 866 to Baldonado et al. ("Baldonado")
1038	Declaration of Delainey Tanner, dated November 18, 2024
1039	Kaitlin Duck Sherwood, Overcome Email Overload With Eudora 5, World Wide Webfoot Press (excerpts) (1st ed.) (2001)
1040	Claims Listing
1041	Donald Knuth, The Art Of Computer Programming Vol. 3 Sorting And Searching, Addison Wesley Longman (excerpts) (2nd Ed.) (May 1998)
1042	United States District Courts Judicial Caseload Profile (excerpt)
1043	Steven Schwartz & Robert Correll, Running Microsoft Office 2001 for Mac, Microsoft Press (excerpts) (2001)
1044	Ben Shneiderman, Clarifying Search: A User-Interface Framework For Text Searches, D-Lib Magazine (Jan. 1997)
1045	N/A
1046	N/A
1047	N/A

Exhibit No.	Description of Document
1048	N/A
1049	N/A
1050	N/A
1051	N/A
1052	Ben Shneiderman, Designing The User Interface: Strategies For Effective Human-Computer Interaction, Addison Wesley Longman (excerpts) (3rd ed.) (1998)
1053	Claudine Badue et al., Distributed Query Processing Using Partitioned Inverted Files, Proc. Eighth Symp. on String Processing and Info. Retrieval, IEEE, 10-20 (2001)
1054	U.S. Patent No. 5,685,003 to Peltonen et al. ("Peltonen")
1055	Dick Pountain, The New Penguin Dictionary of Computing, Penguin Group (excerpts) (1st ed.) (2001)
1056	N/A
1057	Alfred & Emily Glossbrenner, Search Engines For The World Wide Web, PeachPit Press (3rd ed.) (2001)
1058	N/A
1059	Adam A. Wilcox, PC Learning Labs Teaches Lotus Notes 3.0, Ziff-Davis Press (1993)

Exhibit No.	Description of Document
1060	Declaration of Dr. Sylvia Hall-Ellis Dated December 16, 2024
1061	Stephen Londergan & Pat Freeland, Lotus Notes R5 For Dummies Quick Reference, IDG Books Worldwide, Inc. (1999)
1062	Declaration of Dr. Ingrid Hsieh-Yee Dated December 16, 2024
1063	Exhibit CX-0196 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1064	Exhibit CX-0198 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1065	Exhibit CX-0199 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1066	Exhibit CX-0203 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1067	Exhibit CX-0204 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1068	Exhibit CX-0206 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1069	Exhibit CX-0208 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1070	Exhibit CX-0209 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1071	Exhibit CX-0211 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)

Exhibit No.	Description of Document
1072	Exhibit CX-0213 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1073	Exhibit CX-0214 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1074	Exhibit CX-0215 to Complainant's Pre-Trial Brief in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (August 29, 2024)
1075	U.S. Patent No. 9,663,139 ("'139 patent")
1076	File History of U.S. Patent No. 9,663,139
1077	U.S. Patent No. 10,552,490 to Gross et al. ("'490 patent")
1078	File History of U.S. Patent No. 10,552,490
1079	Transcript, Evidentiary Hearing (Vol. IV) in the Matter of Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof, ITC-337-TA-1389 (excerpts) (October 9, 2024)
1080	N/A
1081	First Amended Complaint in *X1 Discovery, Inc. v. Microsoft Corporation* 8:23-cv-02415-GW-JDE (C.D. Cal), ECF No. 61 (May 9, 2025)
1082	Scheduling Order in *X1 Discovery, Inc. v. Microsoft Corporation* 8:23-cv-02415-GW-JDE (C.D. Cal), ECF No. 62 (May 15, 2025)
1083	*C.D. Cal. Time to Milestones*, Docket Navigator (last visited May 15, 2025)
1084	*Judge George H. Wu Time to Milestones*, Docket Navigator (last visited May 15, 2025)

Exhibit No.	Description of Document
1085	Initial Determination Granting Summary Determination of Non-infringement, *Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof*, 337-TA-1389, Order No. 18 (Sep. 3, 2024)
1086	Declaration of Arie Pellikaan, dated May 20, 2025
1087	Windows 10 available as a free upgrade on July 29, Microsoft Corporation (June 1, 2015)
1088	Windows 10 available as a free upgrade on July 29, Microsoft Corporation (June 1, 2015)
1089	Windows Guide, Using Windows Vista, Microsoft Corporation (2008)
1090	Complaint, X1 Discovery, Inc. v. Microsoft Corp.
1091	Screenshots from "Superfast searching in Windows Vista"
1092	Mary Jo Foley, "Microsoft Releases First Public Test Build of Windows Search 4.0"
1093	X1 Products webpages, "Overview" and "FAQ"
1094	David Pogue, Windows Vista: The Missing Manual (2007) (excerpts)

PATENT OWNER'S EXHIBIT LIST

Exhibit No.	Description of Document
2001	Declaration of Mr. Martin Haeberli, dated April 28, 2025
2002	CV of Mr. Martin Haeberli
2003	Revised Stipulation to Stay, *X1 Discovery, Inc. v. Microsoft Corp.*, No. 8:23-cv-02415 (C.D. Cal. Feb 29, 2024), ECF No. 52
2004	Order Granting Revised Stipulation to Stay, *X1 Discovery, Inc. v. Microsoft Corp.*, No. 8:23-cv-02415 (C.D. Cal. Mar. 1, 2024), ECF No. 53
2005	Joint Status Report, *X1 Discovery, Inc. v. Microsoft Corp.*, No. 8:23-cv-02415 (C.D. Cal. Mar. 12, 2025), ECF No. 57
2006	Minutes of Status Conference, *X1 Discovery, Inc. v. Microsoft Corp.*, No. 8:23-cv-02415 (C.D. Cal. Mar. 12, 2025), ECF No. 59
2007	Excerpt of the Initial Determination Granting-In-Part Respondents' Motion for Summary Determination, *Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof*, ITC-337-TA-1389, at 14 (Sept. 3, 2024)
2008	Excerpt of the Initial Determination of Violation of Section 337 and Recommended Determination on Remedy and Bond , *Certain Computing Devices Utilizing Indexed Search Systems and Components Thereof*, ITC-337-TA-1389, at 31-32 (Feb. 26, 2025)
2009	Email from counsel for Microsoft to counsel for X1 regarding *Sotera* stipulation, Apr. 18, 2025
2010	File History of U.S. Patent No. 7,370,035
2011	Information Disclosure Statement for US 11/108,335 dated June 26, 2008
2012	Information Disclosure Statement for US 15/199,928 dated June 30, 2016 (now U.S. Patent No. 10,897,443)

Exhibit No.	Description of Document
2013	Office Action for US 11/128,397 dated May 18, 2007
2014	Office Action for US 11/108,479 dated March 24, 2008
2015	Office Action for US 11/771,554 dated Dec. 23, 2009 (now U.S. Patent No. 8,533,176)
2016	Office Action for US 12/786,246 dated Jan. 10, 2012
2017	Office Action for US 13/918,306 dated June 24, 2015
2018	Information Disclosure Statement for US 15/012,684 dated Feb. 1, 2016 (now U.S. Patent No. 10,546,030) and filing receipt
2019	***Reserved***
2020	Supplemental Declaration of Martin P. Haeberli
2021	Transcript from the Deposition of Dr. Loren Terveen, dated October 21, 2025
2022	Dan Cosley et al., *SuggestBot: Using Intelligent Task Routing to Help People Find Work in Wikipedia*, Proceedings of the 12th International Conference on Intelligent User Interfaces (2007), marked as Exhibit 1 at the Deposition of Dr. Loren Terveen
2023	Declaration of Dr. Loren G. Terveen from *Microsoft Corp. V. X1 Discovery, Inc.*, IPR2025-00255, Ex. 1003
2024	Windows XP Home Edition System Requirements (2001)