

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SLACK TECHNOLOGIES, LLC. and
SALESFORCE, INC.,
Petitioners,

v.

WRINKL, INC.,
Patent Owner

IPR2024-01490
U.S. Patent No. 11,973,731

Reply Declaration of Andrew Cockburn
For *Inter Partes* Review of U.S. Patent No. 11,973,731

Table of Contents

	<u>Page</u>
I. Introduction.....	1
A. Engagement.....	1
B. Background and Qualifications.....	1
C. Compensation and Prior Testimony.....	1
D. Information Considered.....	1
E. Legal Standards for Patentability.....	2
II. Specific Rebuttals to the Almeroth Declaration	2
A. Claim Construction.....	2
B. Level of Ordinary Skill in the Art.....	2
C. Invalidity Ground 1	3
1. Claim 1	3
a. Limitation 1[c].....	3
b. Limitation 1[h].....	5
c. Limitation 1[i].....	7
d. Limitations 1[n] and 1[o].....	7
2. Claim 2.....	10
3. Claim 3.....	12
4. Claim 4.....	13
5. Claim 5.....	14
6. Claim 6.....	15
7. Claim 7.....	15

8.	Claim 9.....	15
9.	Claim 10.....	17
10.	Claims 11 and 12.....	17
11.	Claims 15–19, 21–22, and 25–27	19
12.	Claim 30.....	19
D.	Invalidity Ground 2	20
1.	Claim 1	20
a.	Limitations 1[c], 15[b], and 25[b]	20
b.	Limitations 1[d], 15[c], and 25[c]	22
c.	Limitations 1[o], 15[n], and 25[n].....	23
2.	Limitations 1[i], 1[p], 3[a], 13[a], 15[h], 15[o], 17[a], 23[a], 25[h], 25[o], 27[a], and 28[a].....	24
3.	Claim 4.....	27
4.	Claims 9, 12, 21	28
5.	Claim 13.....	29
6.	Claim 14.....	31
7.	Claims 23, 24, 28, and 29	33
8.	Claim 30.....	33
E.	Invalidity Ground 3	34
1.	Limitations 1[c], 15[b] and 25[b]	34
2.	Claims 13–14, 23–24, 28–29	35
3.	Motivation to Combine Kakuta in view of Matsumoto.....	37
F.	Invalidity Ground 4	38

1.	Claims 1, 15, and 25	38
a.	Chang Discloses a “Second User-Selectable Control”...	38
2.	The Proposed Combination Would Result in The Claimed Invention	39
3.	Claim 4	40
4.	Claims 8 & 20	42
5.	Claim 12	43
6.	Motivation to Combine Kakuta in view of Matsumoto and Chang	46
G.	Invalidity Ground 5	48
1.	Claim 30	48
2.	Motivation to Combine	50
III.	Conclusion	50
	APPENDIX A: Information Considered	52

I. Introduction

A. Engagement

1. I have been retained by Slack Technologies, LLC and Salesforce, Inc. (collectively, “Petitioners”) as an expert in connection with Petitioners’ Petition for *Inter Partes* Review of U.S. Patent Number 11,973,731 (the “’731 Patent”). I have been asked to analyze the ’731 Patent and its claims, provide a description of the state of the art of the technology described in the ’731 Patent, and compare the claims of the ’731 Patent to certain references discussed below, which I understand are prior art to the ’731 Patent. I submitted my original Declaration explaining the above analysis in this proceeding on September 26, 2024, and I understand that Declaration was designated as Exh-1003 (“Original Declaration”).
2. I have been asked to review the Declaration of Dr. Kevin C. Almeroth Regarding Petitioners’ Petition for *Inter Partes* Review of U.S. Patent No. 11,973,731, dated August 22, 2025, and submitted as Ex-2003 (“Almeroth Declaration”). I have been asked to analyze whether the Almeroth Declaration changes the opinions that I set forth in my Original Declaration. It does not.
3. I provide herein rebuttal opinions to certain statements or arguments made in the Almeroth Declaration with which I disagree.

B. Background and Qualifications

4. My background and qualifications were provided in my Original Declaration (¶¶6–19) and in Exh-1004.

C. Compensation and Prior Testimony

5. I am being compensated at an hourly rate of \$600/hour for my work on this declaration. My compensation is not contingent on the outcome of this case or the testimony I provide, and the opinions provided here are my own.

D. Information Considered

6. My opinions are based on my years of education, research, and experience, as well as my investigation and study of relevant materials. In forming my opinions, I have considered the materials discussed herein as well as those listed in **Appendix A** to this Declaration.

7. My analysis of the materials produced in this investigation is ongoing and I will continue to review any new material as it becomes relevant to this case.
8. I may supplement and/or amend the opinions expressed here in response to positions taken by the Patent Owner or experts retained on Patent Owner's behalf. To amplify what is stated here, where necessary, and especially in view of information not presently known to me or new information presented by the Patent Owner prior to, or during trial for *Inter Partes* Review in this matter, I reserve the right to supplement and/or amend this declaration should additional information be brought to my attention during the course of this proceeding.

E. Legal Standards for Patentability

9. I have been informed of the general legal principles for determining whether patent claims are patentable over the prior art, as discussed in my Original Declaration (¶¶26–29).

II. Specific Rebuttals to the Almeroth Declaration

A. Claim Construction

10. As I explained in my Original Declaration, I have interpreted the claims of the '731 Patent based on their plain and ordinary meaning from the point of view of a person having ordinary skill in the art at the time of the alleged invention ("PHOSITA").

B. Level of Ordinary Skill in the Art

11. The Almeroth Declaration sets forth opinions on the level of ordinary skill that I believe misrepresent the expertise that would have been typical in the relevant art. Dr. Almeroth characterizes a PHOSITA as someone with only a bachelor's degree in computer science and two years of experience in *general communications software*. Ex-2003, ¶¶40–41. In my view, that description is far too narrow.
12. The '731 Patent concerns the design and implementation of multi-pane, thread-based messaging interfaces, a field rooted in human-computer interaction, user-interface engineering, and collaborative communication systems, not merely in basic networking or backend software. By 2015, professionals working in this area routinely possessed specialized experience developing and evaluating interactive graphical user interfaces for real-time

messaging and collaboration platforms. Accordingly, a PHOSITA would have had both formal training and several years of hands-on experience in graphical interface design and usability—precisely the background that enables understanding of the claimed invention. Dr. Almeroth’s narrower definition artificially lowers the standard and leads him to underestimate what such an artisan would have recognized as conventional or obvious in 2015.

13. The Almeroth Declaration sets forth opinions on how a PHOSITA would understand the claim terms based on their alleged plain and ordinary meaning, as well as on the prior art references cited in Petitioner’s Petition that, in Dr. Almeroth’s view, do not render the ’731 Patent unpatentable. I disagree with the Almeroth Declaration as set forth below.

C. Invalidity Ground 1

1. Claim 1

a. Limitation 1[c]

14. I disagree with Dr. Almeroth’s assertion that Kakuta does not disclose a drop-down menu, as well as his further contention that a drop-down menu is not a “pane” and would not have been understood by a PHOSITA to constitute a “pane.” Ex-2003, ¶88. In fact, the term “menu pane” pane was well known in the art of graphical user interfaces. *See* Exh-1024, 12.
15. Kakuta discloses a drop-down menu in Figure 8. *See* Exh-1005, Fig. 8. It was well known in the art that the arrow symbol associated with a drop-down menu indicates the presence of multiple list items, visually signaling to the user that additional selectable options are available in the menu list, including by scrolling. For example, the figures below show a drop-down menu associated with the “Position” field in a graphical user interface. When the down arrow (similar to that in Kakuta’s Figure 12) is activated, the drop-down menu pane displays a scrollable list for selecting alternative “Position” values. Exh-1025, 365–66.

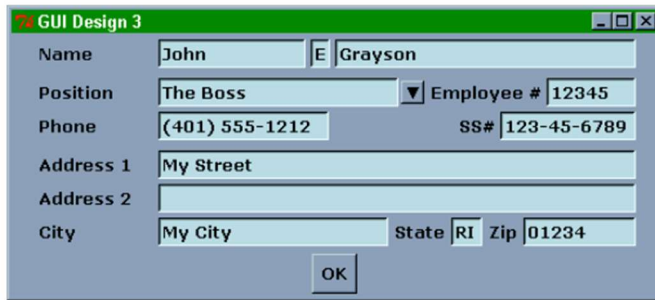


Figure 16.3 A GUI showing logical field grouping



Figure 16.4 ComboBox widget

16. Modern chat clients (e.g., IRC derivatives) contemporaneous with Kakuta routinely employed collapsible lists. *See* Exh1023, 5; *see also* Ex-2003, ¶61 (acknowledging that “Kakuta builds on traditional Internet Relay Chat (“IRC”) technology”). For instance, IRC systems at the time depicted configurable pop-up menus capable of listing channels, allowing users to create menus displaying a designated set of channels. Exh-1023 7, 44, 72. Thus, I disagree with Dr. Almeroth’s assertion that “[t]here is no evidence that IRC systems around the time of the invention used drop-down menus from the main chat screen to handle channel selection.” Ex-2003, ¶92.
17. It was also well known in the art that a drop-down would expand to display a larger list of items/options for the user to scroll through; otherwise, those items would remain hidden from the user. A PHOSITA would have understood that such expandable menu panes therefore would fall within the scope of a “pane.”
18. Moreover, I disagree with Dr. Almeroth’s assertion that a PHOSITA would not employ a drop-down menu in Kakuta for a channel listing in the hundreds or thousands. *Id.*, ¶92. IRC systems at the time were capable of managing varying numbers of channels depending on the server (Exh-1023 8), which is why a down arrow appears to the right of “#CH 1” in Figure 12, allowing users to select channels, including by scrolling. Exh-1005, Fig. 12.

Accordingly, I also disagree with Dr. Almeroth's contention that such an approach would consume excessive screen real estate.

19. It was also well known in the art that graphical interfaces may provide multiple redundant controls to perform the same action in order to enhance the user experience. It is obvious to a PHOSITA to design a graphical interface such that a user could initiate the same command either by selecting a menu item or by clicking a toolbar icon.
20. For example, IRC systems at the time explained that the Channels List dialog could be accessed by clicking its toolbar button, selecting the option from the Tools menu, or pressing Alt + L. Exh-1023, 8. Kakuta itself clearly demonstrates two redundant mechanisms by which a user can navigate among channels: using the down-arrow button on the right to browse an array of channels, and the two arrow buttons on the left to move through channels one by one. Exh-1005, Fig. 12.
21. Patent Owner also argues that in Figure 8, “[w]hile the name of a selected channel (‘#CH1’) appears above the chat window, that channel identifier area is not a pane as that term is used in the art, and regardless only a single channel name is displayed.” Paper 16, 20. However, Kakuta already depicts a title field displaying “#CH1” at the top left of Figure 8. Exh-1005, Fig. 8; *see also id.*, 5–7, 9, 12. A PHOSITA would view it as an unconventional and inefficient design choice to include a redundant, *passive* title display in two separate locations of the interface—yet that is precisely what Patent Owner's interpretation implies.
22. Accordingly, Dr. Almeroth's assertion that Kakuta's “Channel List” would render a drop-down menu redundant is misplaced, as a PHOSITA could have regarded Kakuta's “Channel” button as simply one of several possible access points.

b. Limitation 1[h]

23. I disagree with Dr. Almeroth's assertion that Kakuta does not disclose an “association with” the first user-selectable control (i.e., the “Create” control on floating toolbar 305) and a specific message. Ex-2003, ¶ 97. A PHOSITA would understand that Kakuta establishes a *functional association* between the toolbar and the selected message(s) by allowing the user to select one or more messages in the chat window and then invoke a control from the toolbar to operate on those selections.

24. A PHOSITA would understand that a ***visual association*** exists when a control is graphically or spatially linked to a particular display element, such that its placement or alignment visually conveys a relationship between the two within the user interface. For instance, a “Join Channel” button positioned directly beneath a channel list visually indicates that it operates on that list by virtue of its spatial placement.
25. Likewise, a PHOSITA would understand that a ***functional association*** exists when a control is linked to another interface component by its operational behavior, regardless of where it is displayed. For example, a “Refresh” button located in a toolbar may functionally update a channel list elsewhere on the screen, even though it is not visually adjacent to it.
26. Kakuta discloses a floating toolbar (element 305) containing selectable controls—including “Create”—that act directly on messages selected in the chat window. Exh-1005, 8:30–47. A PHOSITA would understand that when a user highlights one or more messages and presses “Create,” the control is ***functionally*** linked to those selected communications, satisfying the claimed “association with” requirement. This is precisely the type of operational relationship (i.e., a functional association) between control and content contemplated by the claim limitation.
27. Dr. Almeroth’s characterization of toolbar 305 as merely “generic” and incapable of associating with a specific message (Ex-2003, ¶96) misreads Kakuta. Figure 8 expressly depicts use of the toolbar following message selection in the chat window. When the user highlights a message and presses “Create,” the system generates a thread based on that selected communication—demonstrating the contextual linkage between the control and the message. Exh-1005, 8:38–47. Moreover, contrary to Dr. Almeroth’s suggestion, Kakuta describes multiple mechanisms for thread creation, including selection-based invocation of the “Create” control, not just the sequence derived from Figures 4–8.
28. The system’s ability to highlight selected messages in conjunction with activation of the toolbar control demonstrates a clear ***functional association*** between the user-selectable control and the corresponding message content. This interactive linkage—well understood by a PHOSITA as standard GUI behavior—confirms that Kakuta discloses the claimed “association with” element. The selection and highlighting of messages provide the necessary association, both visual and functional.

c. Limitation 1[i]

29. I disagree with Dr. Almeroth that Kakuta does not display a third pane “in response to” a user selection of said first user-selectable control. Ex-2003, ¶¶99. Dr. Almeroth’s analysis incorrectly assumes that the limitation requires an instantaneous or single-step process for displaying the third pane following a user selection. *Id.*, ¶¶100–02. I disagree.
30. In my opinion, the limitation does not require the third pane to appear immediately or through a single-step action. The phrase “in response to” merely requires that the display of the third pane be triggered by the user’s interaction with the first control. A PHOSITA would understand that, in Kakuta, the third pane is displayed as a result of the user’s activation of the “Create” control, even if an intermediate confirmation or dialog window appears before the pane is shown. *See* Exh-1005, Fig. 6. A PHOSITA would understand that the “Create” control remains the initiating event that causes the system to generate and display the third pane.
31. For example, when a user selects “Save” from a program’s “File” menu, the system may present a confirmation dialog asking the user to provide a file name or confirm overwriting an existing file. Even though the user must click “OK” within that intermediate dialog, the resulting save operation still occurs in response to the original “Save” command. Likewise, in Kakuta, the display of the third pane remains “in response to” the user’s selection of the “Create” control, even if an intervening dialog or confirmation step occurs before the pane appears.

d. Limitations 1[n] and 1[o]

32. I disagree with Dr. Almeroth’s assertion that Kakuta lacks an “anchor message.” Ex-2003, ¶106. Kakuta expressly discloses that when a user selects one or more messages in the chat window and activates the Create control, those selected messages are carried forward into a new thread window (element 307). Exh-1005, 8:38–42; Fig. 7. The first message brought into that thread window serves as the starting point of the subsidiary communication—i.e., the anchor message. A PHOSITA would understand that this carried-over message functions as the reference point from which subsequent messages in the subsidiary thread originate.
33. A PHOSITA would further understand that an explicit label or icon is not required for a message to function as an anchor message. The claim

language imposes no such requirement; what matters is the message's function—serving as the contextual foundation or starting point for a subsidiary conversation. In IRC-based systems like Kakuta, the message carried into the new thread window performs exactly this function, consistent with conventional thread anchoring practices.

34. Dr. Almeroth's characterization of this message as merely the "top-most" message in the thread window (Ex-2003, ¶107) fails to appreciate that this is precisely how anchor messages are represented in typical GUI and IRC threading systems. The first message appears at the top of the thread window and provides the conversational context for subsequent responses. Exh-1005, Fig. 7. Nothing in the claims requires that the anchor message be specially formatted or explicitly labeled. A PHOSITA would recognize that IRC interfaces often rely on positional context—displaying the originating message at the top—rather than an explicit anchor label.
35. Dr. Almeroth's suggestion that the "top-most" message position in Kakuta is arbitrary because it might reflect a "past" or "new" message overlooks the fundamental principle of message-thread continuity. Even when "Quote Messages" is selected, the messages appearing at the top of the thread window are intentionally carried over from the main chat to preserve conversational context. Exh-1005, 8:20–29, 14:51–58. A PHOSITA would understand that this carry-over creates a *functional* link between the selected message(s) and the new thread, thereby defining the first displayed message as the anchor for that thread.
36. Moreover, the argument that the "top-most" message may not be the earliest message in the thread is irrelevant. The claims do not require that the anchor message be chronologically first—only that it serves as the reference communication to which the subsidiary thread relates. Kakuta's threading process inherently designates such a reference through its creation operation, where selected messages or users define the set of communications forming the thread. Exh-1005, 8:38–47.
37. Kakuta's data structure further supports this interpretation. Figure 2 shows the relationship between message IDs and thread IDs—e.g., messages 1, 5, and 6 associated with thread ID "Th1." Exh. 1005, 4:57–65. This correlation demonstrates that Kakuta's system tracks the originating message and its associated thread. A PHOSITA would understand that this correlation ensures the anchor relationship is preserved in both storage and display. Exh-1005, 6:1–21, Fig. 2.

38. Dr. Almeroth’s claim that thread titles in Kakuta (e.g., “Program,” “Meeting”) are “arbitrary” and “signify nothing about any particular message” misreads the reference. Ex-2003, ¶¶104–06, 114. The thread titles correspond to the topical content of the messages that initiated the threads. Kakuta states that a thread may be created by “selecting messages as a whole, or words included in messages.” Exh-1005, 8:45–47. A PHOSITA would recognize that assigning a thread title such as “Program” directly reflects the subject matter of the originating message (“It’s about programming”). This linkage functions as an indication that the corresponding message serves as the anchor for that thread.
39. Accordingly, Kakuta discloses both an anchor message and its indication through its user flow and data structure. The thread creation process selects messages from the main chat and populates the thread window around them. The administration module stores these messages with unique message and thread IDs, ensuring that the anchor relationship is maintained even if not explicitly labeled on-screen. A PHOSITA would understand that Kakuta inherently designates and preserves anchor-message associations in its architecture.
40. I also disagree with Dr. Almeroth’s assertion that Kakuta lacks a “second user-selectable control.” Ex-2003, ¶111. Kakuta’s “Display” control on toolbar 305 operates precisely as the claimed second user-selectable control. Once a user selects a message (or set of messages) in the chat window, pressing “Display” opens the corresponding thread window containing those messages. Exh-1005, 9:23–31. The toolbar’s floating position does not break its *functional association* with the selected message; a PHOSITA would recognize that association is defined functionally, not necessarily spatially.
41. Dr. Almeroth misreads the claim phrase “display a second user-selectable control associated with said first messaging communication *in said second pane*.” Ex-2003, ¶111 (emphasis added). The phrase “in said second pane” modifies the message, not the control. The claim merely requires that the first messaging communication appear in the second pane and that the second control be associated with it—it does not require that the control physically reside within the same graphical boundaries. A PHOSITA would understand that such an association can be established through selection or command linkage, consistent with standard GUI design principles. Regardless, even under Dr. Almeroth’s interpretation, Kakuta still meets the limitation because the floating toolbar is movable and shown within the boundaries of the second pane. See Exh-1005, Figs. 5, 8.

42. Moreover, the “Display” control, at the very least, inherently identifies the anchor message and satisfies the “indication” requirement. Its operation causes the retrieval of the thread data linked to a specific anchor message (e.g., “Program”), displaying it in the thread window. Exh-1005, Figs. 7, 12. A PHOSITA would understand that by triggering display of a thread containing that specific message, the control functions as an implicit indicator that the corresponding message is the thread’s anchor.
43. In sum, Kakuta discloses (1) an anchor message—the first message initiating a thread, (2) an indication—through thread titles and message carry-over establishing association, and (3) a second user-selectable control—the “Display” control that opens the thread window containing the anchor message. A PHOSITA would recognize that these features collectively satisfy Limitations 1[n] and 1[o]. Dr. Almeroth’s contrary interpretation misapplies the claim language, ignores standard GUI conventions, and overlooks how IRC-based systems naturally convey anchor-message relationships through both structural and functional association.

2. Claim 2

44. I disagree with Dr. Almeroth’s assertion that Kakuta fails to disclose “additional messaging communications” and that Kakuta displays chat messages exclusively in either the chat window or the thread window, **but never in both**. Ex-2003, ¶¶118–19. Dr. Almeroth’s conclusion rests on a misreading of Kakuta—particularly a narrow interpretation of both Figures 7 and 14—and overlooks the system’s explicit disclosure that messages can be concurrently displayed in both panes. A PHOSITA would understand that Kakuta teaches the ability for a user to input a message into the side-bar text entry field (input field 309), which can then be displayed in both the messaging pane (chat window 303) and the thread pane (thread window 307).
45. Figure 7 depicts an example of an active thread interface showing the concurrent display of the messages “It’s about programming” and “It’s about programing,” illustrating that Kakuta teaches simultaneous display. See Exh-1005, Fig. 7. The figure further shows thread window 307 with message window 308 and input field 309, confirming that a user may input additional messages within the thread window. See Exh-1005, 8:30–34, Fig. 7.
46. Dr. Almeroth improperly limits his analysis to the sequence preceding Figure 7, treating it as a single-use example rather than a representative

embodiment. Ex-2003, ¶121. However, Kakuta expressly describes Figure 7 as an example of a thread window “when a new thread has been created,” not the *only* instance in which simultaneous display occurs. *See* Exh-1005, 8:30–31. A PHOSITA would recognize that the sequence leading to Figure 7 demonstrates one example of the display process and not an exclusion of other configurations.

47. Kakuta unambiguously discloses that messages can be shown in both the chat and thread windows. Exh-1005, 15:11–22 (“In aforementioned FIG. 7, messages that have been displayed in thread window 307 are also displayed in chat window 303.”); *see id.*, 15:14–17 (“The setting, however, can be such that messages displayed in the thread window are not displayed in the chat window. Preferably, the settings are such that the user can make them.”); *see id.*, Cls. 15, 17. This language confirms that simultaneous display is not only possible but configurable by the user.
48. Accordingly, a PHOSITA would understand that when a user inputs a message into input field 309, that message may be displayed as an “additional messaging communication” in thread window 307 while also being simultaneously displayed in the chat window 303—precisely as depicted in Figure 7.
49. Dr. Almeroth’s reliance on Figure 14—particularly step S15—to argue that Kakuta prohibits concurrent display also reflects a misunderstanding of the disclosure. *See id.*, Fig. 14. Step S15 merely depicts the program’s decision logic for determining where to render a message at a given time; it is not a rigid restriction against dual display. *See id.* The logic reflects a conditional flow: when a thread window is open, messages are routed there; when it is not, messages are displayed in the main chat window. *See id.*, 10:65–11:3. A PHOSITA would readily recognize this as a dynamic routing process, not an exclusionary limitation.
50. Further, Kakuta’s administration module 23 explicitly allows messages to be transmitted to both the control module 32 (which manages the main chat display) and the third input/output module 24 (which manages the thread window) “in accordance with user instructions.” *Id.*, 7:48–60. This architecture confirms that Kakuta supports concurrent routing of the same message to multiple display destinations, consistent with user preferences or system configuration.

51. In any event, nothing in Kakuta limits or precludes such dual-display functionality, and a PHOSITA would have readily understood the system to accommodate it.
52. Accordingly, Dr. Almeroth's interpretation imposes an artificial restriction that is unsupported by the claim language, and inconsistent with the teachings of Kakuta.

3. Claim 3

53. I disagree with Dr. Almeroth's assertion that Kakuta fails to disclose any third user-selectable control associated with "additional" thread messages. Ex-2003, ¶126. Dr. Almeroth's interpretation overlooks the **functional association** between Kakuta's toolbar controls—such as "Link" and "Display"—and the thread messages displayed in the main chat window 30 (the purported "second pane"). A PHOSITA would understand that these controls are designed to operate directly on selected messages or threads, thereby satisfying the "third user-selectable control" limitation.
54. As discussed above, Kakuta teaches that a user may input additional messaging communications through input field 309, which can then be displayed in both the chat and thread panes simultaneously. *See* Section II.C.2. This functionality establishes that thread-related messages remain accessible and operable from within the second pane, consistent with the claim's requirement that the third control be associated with "additional" messaging communications.
55. Kakuta expressly discloses that toolbar 305, which includes the "Display" control, operates on the user's selected message or thread. When the user selects a message and activates "Display," the system retrieves the associated thread data and opens the subsidiary thread window (the "third pane") containing at least a portion of that thread. Exh-1005, 9:23–31, 14:46–58; Figs. 7, 12, 19. A PHOSITA would understand that although toolbar 305 is a floating and movable interface element, it maintains a **functional association** with the selected message because it acts directly upon the user's current selection within the second pane.
56. Dr. Almeroth also mischaracterizes the functionality of the "Link" button as performing merely a "single function" of opening a thread list. Ex-2003, ¶127. However, Kakuta explicitly describes in the "Link Process Subroutine" that "[t]he creation module ... displays the rearranged messages

in the new thread window.” Exh-1005, 13:28–30. This process does more than open a list—it displays the linked thread content in a separate thread window. That is precisely what Claim 3 requires: a user-selectable control that, when activated, displays at least a portion of a thread containing the associated messages.

57. Accordingly, Kakuta discloses a third user-selectable control that performs the claimed operation—triggering the opening of a thread (third pane) corresponding to the selected “additional” thread message in the second pane.

4. Claim 4

58. I disagree with Dr. Almeroth’s assertion that Kakuta provides “no such visual relationship” and therefore fails to meet the proximate limitation. Ex-2003, ¶134. Kakuta clearly discloses a third user-selectable control—specifically, the “Link” control—displayed proximate to the additional messaging communication. As illustrated in Figure 7, Kakuta positions the toolbar 305, which contains the third user-selectable control, near to the displayed message elements, thereby satisfying the claimed “proximate” relationship. Exh-1005, Fig. 7. This also takes into account the ambiguity of Claim 4 because the “said additional messaging communication” is displayed in both the third pane (*see* Limitation 1[n]) and second pane (*see* Claim 2).
59. Moreover, Kakuta explicitly identifies toolbar 305 as a floating toolbar, which can be repositioned within the user interface of the display control device. *See id.*, Figs. 1, 12; *see also id.*, Figs. 4 and 5 (illustrating toolbar 305 in multiple locations). This flexibility allows the toolbar—and consequently the “Link” control—to be displayed in close spatial proximity to the relevant message communications as determined by user configuration or interaction.

Fig. 8

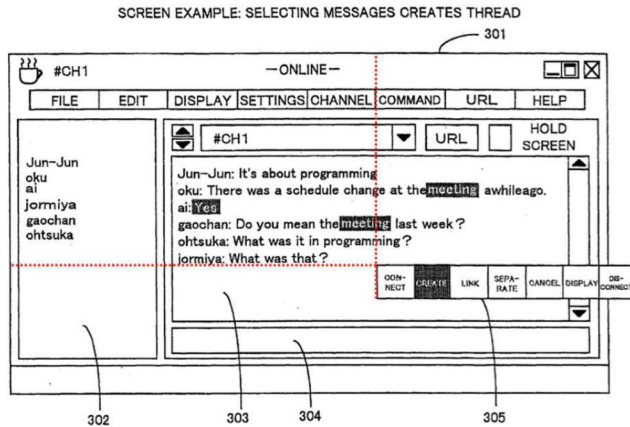
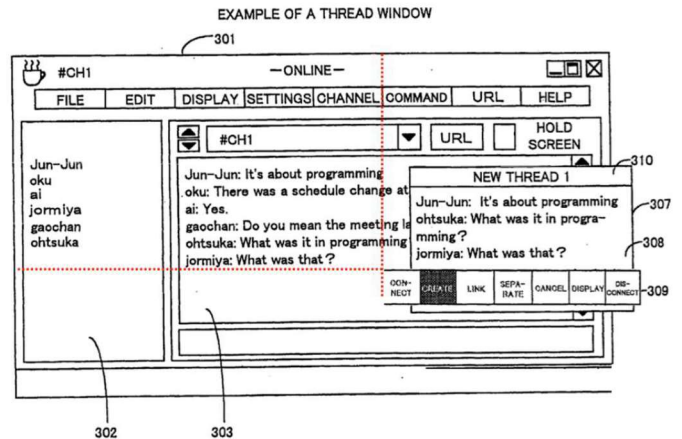


Fig. 7



Exh-1005, Figs. 7 and 8 (superimposed).

60. Contrary to Dr. Almeroth's position, a PHOSITA would understand that the very purpose of a floating toolbar is to enhance usability by allowing the user to position frequently used controls—such as “Link”—in visual proximity to the content or message being acted upon. The ability to move or dock toolbar 305 near a specific message establishes a visual association between the control and the displayed communication. The precise pixel distance or static placement is immaterial; what matters is that the interface design permits the control to appear near the message it operates on, satisfying the “proximate” requirement. In the context of user-interface design, a PHOSITA would recognize that such a movable toolbar provides an intentional and functional visual linkage between the control and the associated message, consistent with ordinary GUI conventions.

5. Claim 5

61. Dr. Almeroth merely reiterates his earlier Claim 2 argument that Figure 7 lacks an “additional messaging communication.” Ex-2003, ¶ 136. This conclusion is incorrect. Kakuta expressly discloses the simultaneous display of messages in both the chat and thread panes, and nothing in the record supports Dr. Almeroth's attempt to exclude messages carried into the thread window. *See* Section II.C.2.
62. A PHOSITA would understand that the messages shown within the thread window are the *subsidiary communications* generated in response to the user's selection of a message in the chat window.

63. Moreover, Kakuta explicitly states that message display order “can be ... in a time sequence, [or] a user-and-time sequence,” confirming that the system dynamically presents additional messages as part of the same thread. Exh-1005, 9:1–4.

6. Claim 6

64. Dr. Almeroth merely reiterates the same arguments he raised with respect to Claim 3. Ex-2003, ¶138. For the same reasons discussed above (*see* Section II.C.3), Kakuta discloses a sidebar text-entry field (input field 309) within the third pane (thread window 307) that is operable to receive additional messaging communications. Exh-1005, 8:30–34. A PHOSITA would recognize that this input field enables users to compose and transmit further messages within the subsidiary thread, thereby fully satisfying the functional requirement of Claim 6.

7. Claim 7

65. I disagree with Dr. Almeroth that Kakuta lacks identifying information associated with an anchor message. Ex-2003, ¶141. Kakuta discloses that each message appears with the author’s name at the top of a thread, indicating that a subsidiary thread originates from the anchor message. See Exh-1005, Fig. 7. In my opinion, the author’s name constitutes identifying information associated with the anchor message.
66. Moreover, Dr. Almeroth misinterprets Claim 7, which does not require the identifying information to be a separate label or indicator. A PHOSITA would understand that the author’s name sufficiently identifies the anchor message displayed in a subsidiary thread, and therefore constitutes identifying information. The claim limits neither form nor placement, only requiring information “associated with said anchor message.” Kakuta discloses message-identifying information, and the ’731 Patent’s specification provides no further restriction; a PHOSITA would understand this includes Kakuta’s displayed usernames. Exh-1005, 3:39–47.

8. Claim 9

67. I disagree with Dr. Almeroth’s assertion that Kakuta lacks a “fourth user-selectable control” because, according to him, the selection of users or messages is not a control whose selection causes the display of those messages in the third pane, as required by the claim. Ex-2003, ¶¶143–45. The claims do not require identifying information to appear *only after*

sidebar creation. Kakuta shows the same identifying data (e.g., username) in both the main chat and thread windows, displaying it with additional messaging communications. Exh-1005, 10:65–11:3, 15:11–22, 4:3–5; Fig. 7. Even if “Create” precedes the sidebar field, Kakuta still discloses this behavior, as selected messages and their identifying information can appear in both panes depending on user settings. *Id.* Dr. Almeroth’s position mischaracterizes both Kakuta’s teachings and how a PHOSITA would understand user-selectable controls in GUI-based chat systems built on IRC conventions. Kakuta, consistent with its IRC foundation, discloses multiple user interactions—such as selecting a username, message, or message-identifying information—that operate as controls triggering the creation or display of a new thread window (the “third pane”).

68. For instance, Kakuta explicitly teaches a *first control path* where “the creation means creates, based on the selection of the messages from a user, message groups containing the selected messages and message-identifying information for the selected messages.” Exh-1005, 3:43–47. A PHOSITA would understand this to mean that a user can select either a message itself or its identifying information (e.g., the sender’s name) before initiating thread creation. This functionality mirrors standard IRC practices, where selecting a participant’s name—through a click, double-click, or context menu—initiates a private conversation or subsidiary communication thread. Consistent with these conventions, a PHOSITA would recognize that selecting the author’s name in Kakuta constitutes a user-selectable control based on identifying information that, when selected, causes a new thread to display in the third pane.
69. This understanding is reinforced by contemporaneous IRC clients. For example, IRC systems at the time explained that double-clicking a nickname opens a private “query” window for direct communication with that user. *See* Exh-1023, 28. Likewise, demonstrates that since at least the late 1980s, GUI interfaces supported triple-clicking to select an entire line of text. *See* Exh-1013, 13. A PHOSITA would recognize that selecting a user’s name or triple-clicking a message line in Kakuta similarly selects the message and its identifying information—thereby serving as the initiating user action (control) that opens a new thread window. Thus, the first control path—user/message selection based on identifying information—meets the “fourth user-selectable control” requirement.
70. Kakuta also provides a *second control path* through toolbar 305. As illustrated in Figure 8, the user may highlight one or more messages (or their

identifying information) and then click “Create” to initiate thread creation. Kakuta expressly states that “a thread is created by selecting messages as a whole, or words included in messages.” Exh-1005, 8:45–47. Accordingly, when a user selects the sender’s name or other identifying information and then activates “Create,” that combination of actions constitutes a *user-selectable control based on identifying information* that causes the associated thread to display in the third pane.

71. A PHOSITA would understand that either of these control paths—the (1) direct selection of a user’s name or message line or (2) the selection of identifying information followed by activation of the “Create” control—constitutes the requirements of a fourth-user selectable control from a functional standpoint based on the specific claim limitations. In both cases, the control is *based on identifying information* and triggers display of the related thread in the third pane. Kakuta’s disclosures, along with an understanding of well-known behavior of IRC-based chat systems, would make this equivalence readily apparent to a PHOSITA.
72. Moreover, the intrinsic record reinforces that user identifiers function as actionable elements. Kakuta’s “creation means” explicitly ties the selection of messages or their identifying information to thread creation, confirming that those identifiers operate as user-selectable controls. *See* Exh-1005, 3:43–47; 8:45–47; Fig. 8.

9. Claim 10

73. Dr. Almeroth contends that Claim 10 fails for the same reasons as Claim 1. Ex-2003, ¶146. I disagree. As discussed in Section C.1, Kakuta discloses all of the elements of Claim 1, including a thread window (307) with its own input field (309) displayed concurrently with the main chat input field (304). Exh-1005, Fig. 7. A PHOSITA would understand that both input fields are operable simultaneously, enabling a user to compose and transmit messages in the main chat and thread windows at the same time—fully satisfying the additional limitations of Claim 10.

10. Claims 11 and 12

74. I disagree with Dr. Almeroth’s assertion that Kakuta fails to disclose a request to permit a non-member user to receive messages. Ex-2003, ¶146. Kakuta plainly discloses controls for managing user participation and message visibility within public and private threads—functionalities that

align with well-established user management features in standard IRC-based systems. A PHOSITA would understand that such management controls accommodate user add/remove operations, including those involving non-member users. In particular, IRC conventions contemporaneous with Kakuta provided a standard “/invite” command, which allows a user to request that another user join a private channel—functionally equivalent to “receiving a request to permit a non-member user to receive messages.” Exh-1023, 211–212.

75. Additionally, the ’731 Patent itself discloses that “non-members may be notified by [sic] through an electronic transmission, such as email, that a task is waiting.” Exh-1001, 16:8–20; *see also id.*, 13:38–42, 17:44–18:61. A PHOSITA would therefore understand that the claimed “apparatus” in Claim 1 is broad enough to encompass other software systems on the device, including web-based services such as email (e.g., Hotmail). Likewise, Kakuta discloses that its chat system operates as a “Web chat” carried out with a “Web server providing Web chat services.” Exh-1005, 11:34–36. Accordingly, the availability of other web services—such as email functionality—is inherent in Kakuta’s system. Indeed, Kakuta’s user interface includes a “URL” button (*see, e.g., id.*, Fig. 7), and one of the well-known URL protocols at the time of the alleged priority date was the “mailto:” scheme, which opens a new email message addressed to a recipient. A PHOSITA would thus recognize that a user operating Kakuta could readily copy a message (or messages) and send it to a non-member via email, consistent with the disclosures of the ’731 Patent.
76. Kakuta further discloses mechanisms for handling user participation within chat channels, including managing the joining and departure of users. *See* Exh-1005, 11:6–10. A PHOSITA would understand that the same underlying mechanism that governs participation in private threads could also process a join request from a non-member user. Functionally, this operation is equivalent to “receiving a request to permit a non-member user” and enabling that user to receive communications within the thread. The implementation of such a process would have been straightforward and entirely consistent with standard IRC user-management logic.
77. Moreover, Kakuta explicitly describes administrative functions that manage user participation and message visibility across both public and private threads, with private threads restricted to selected users. Exh-1005, 3:56–63, 11:6–10, Fig. 6. These disclosures confirm that Kakuta’s architecture supports the selective inclusion of users in private discussions,

encompassing scenarios where a non-member's access request is received and approved.

11. Claims 15–19, 21–22, and 25–27

78. I disagree with Dr. Almeroth's conclusions regarding Claims 15–19, 21–22, and 25–27. He offers no new arguments or evidence beyond those already addressed for Claims 1–12. For the same reasons discussed above, a PHOSITA would understand that Kakuta discloses and anticipates the subject matter of these claims.

12. Claim 30

79. I disagree with Dr. Almeroth's assertion that Kakuta's "Display" control does not create a subsidiary thread communication. Ex-2003, ¶149. To the contrary, Kakuta discloses that activating the "Display" control initiates retrieval of message data and causes the system to display a subsidiary thread window containing those messages, thereby instantiating a subsidiary thread communication.
80. A PHOSITA would understand that the term "create" in this context encompasses instantiating or initiating a thread communication process—i.e., performing an action that causes a subsidiary thread window to be generated or displayed, within which the user can later compose or exchange additional thread messages using the input field 309.
81. Kakuta plainly supports this interpretation. When the user selects "Display" from toolbar 305, the system opens a thread selection window listing available thread titles. Exh-1005, 9:23–31; Fig. 12. Upon user selection, the system invokes creation module 21 to execute the display process subroutine shown in Figure 19. *Id.*, 14:46–50. Steps S431–S435 depict a multi-step creation routine in which the system retrieves the selected thread's identifiers from the administration table, acquires the associated message data from storage, and renders the subsidiary thread window populated with those messages. *Id.*, 14:51–58. In other words, the system receives and acts upon the user's request to instantiate—or create—a thread communication environment where messages are acquired, reconstructed, and made available for continued interaction.
82. Even under a narrow construction of "create," Kakuta still satisfies this limitation. Figure 19 depicts a process that does more than merely reopen an existing window; it rebuilds the thread dynamically by retrieving and

populating messages (“acquires messages in the selected threads from the message list using the thread IDs”) and transmitting those messages and display instructions to the I/O module. Exh-1005, Fig. 19 (S433–S434). The system thus performs a functional recreation of the thread—assembling its contents, rendering it in a new interface, and updating its display status (S435). These operations constitute the claimed “creation of a subsidiary thread communication,” as the user’s request initiates a new thread instance in which additional messages can be composed through input field 309.

83. I disagree with Dr. Almeroth’s assertion that Kakuta’s “PROGRAM” label is not a thread summary indicator or is not renderable proximate to any particular message. Ex-2003, ¶151. In Kakuta, the “PROGRAM” label identifies and summarizes the subject matter of a corresponding subsidiary thread. The label appears in a pop-up window adjacent to the main chat interface, and each title within that window—such as “Program” or “Meeting”—opens the related thread conversation. Exh-1005, Fig. 12. A PHOSITA would readily recognize that these titles function as thread summary indicators that identify and summarize the content of their respective subsidiary threads. Nothing in Claim 30[d] limits the indicator’s format (e.g., textual or graphical).
84. The “Program” indicator is *renderable proximate* to the first messaging communication. Figure 12 illustrates the Display tool being invoked to show several thread summary indicators, including “Program,” which represents the subsidiary thread. Moreover, Kakuta discloses that the thread selection window (307) is movable and is therefore renderable in different positions within the interface. *See* Exh-1005, Figs. 10 and 12. Even if the “Program” label is not permanently fixed beside the first message, its placement within a movable window satisfies the proximity requirement because the window can be positioned near the relevant message, consistent with standard GUI design practices. The “Program” label is therefore “renderable ... proximate the first messaging communication.”

D. Invalidity Ground 2

1. Claim 1

a. Limitations 1[c], 15[b], and 25[b]

85. I disagree with Dr. Almeroth’s assertion that Kakuta does not disclose a drop-down list. Ex-2003, ¶160. In Figure 8, Kakuta’s channel identifier

element would have been readily recognized by a PHOSITA as a ComboBox-type drop-down control capable of displaying multiple channel identifiers. Exh-1005, Fig. 8. Graphical user interface (GUI) frameworks available at the time commonly implemented ComboBox widgets that combined a display field with a collapsible list of selectable options. Exh-1012, 52–53, Exh-1025, 365–66.

86. A ComboBox is specifically designed to accommodate a number of selectable items—precisely the function performed by Kakuta’s channel selection feature. *Id.* (“[A ComboBox] allows the user to select from a list of options ... [and] may be scrolled to accommodate a large number of selections.”). A PHOSITA would understand that such controls were routinely used to present extensive lists, such as channel identifiers, in a compact and efficient format. Indeed, ComboBox widgets were well known to allow scrolling and provide access to effectively unlimited selectable values. *See id.*, Exh-1012, 52–53 (e.g., “scrolledlist_items=asply” option in the ComboBox implementation).
87. GUI design references of the time emphasized the flexibility and scalability of ComboBoxes, noting that they “occupy little space, even though they may give the user access to an unlimited number of selectable values.” *Id.*, Exh-1025, 153-154. A PHOSITA would therefore understand that Kakuta’s drop-down channel selection mechanism could readily accommodate either small or large channel lists using a standard ComboBox implementation.
88. Dr. Almeroth further contends that Exhibit 1012’s example of a ComboBox includes a prompt (“Choose a play”) and a subject (“Play:”) followed by a list of options, and asserts that Kakuta lacks such elements. Ex-2003, ¶160 (citing Exh-1012, p. 016–17). However, this argument misreads the cited example.
89. A closer examination of the source code (*see* Exh-1012, p. 016) shows that the example creates **two separate widgets**: a Label widget (around line 9) with the text “Choose play,” and a ComboBox widget with “label_text=‘Play:’” and “dropdown=1” (around lines 19–20). The Label widget’s text (“Choose play”) is not part of the ComboBox itself but rather a separate GUI element. A PHOSITA reviewing this example would recognize that ComboBoxes typically do not require or include a separate Label widget immediately above them. Furthermore, a PHOSITA would recognize that the text “Play:” displayed at the left of the ComboBox widget in this example could be replaced by any text, including no text by setting

“label_text=””. Indeed, the example of Figure 2.2 of Exh-1025 reproduced below, shows a set of ComboBox widgets, two of which have no associated “label_text.” Thus, Dr. Almeroth’s reliance on the presence of a label as a structural feature of a ComboBox is misplaced. When the ComboBox example of Exh-1012 on p. 016 is properly understood, it is consistent with Kakuta’s disclosure of a ComboBox with no associated label.

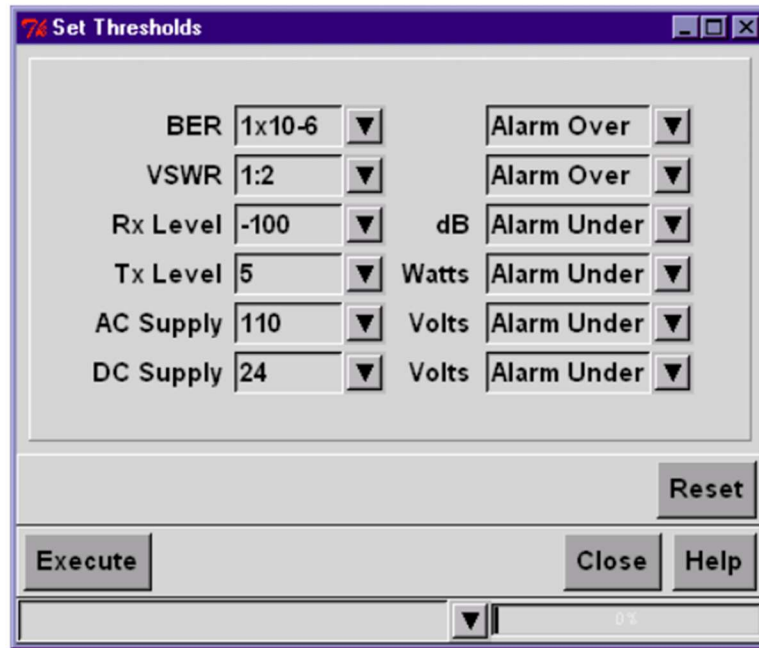


Figure 2.2 Tkinter and Pmw on win32

90. Moreover, a PHOSITA would have found it both obvious and natural to implement Kakuta’s user-selectable channels as a drop-down list. Listing available channels in a collapsible ComboBox-type interface was a well-known and intuitive enhancement for managing numerous selectable options within limited screen space. The existence of a “Channel” button in Kakuta does not preclude the use of a drop-down list; rather, the button could easily serve as one of multiple entry points for accessing the same channel list. A PHOSITA would thus recognize that Kakuta’s channel identifier control inherently or, at minimum, obviously corresponds to a ComboBox-type drop-down list consistent with the claim limitations.

b. Limitations 1[d], 15[c], and 25[c]

91. Dr. Almeroth reiterates his prior arguments concerning the ComboBox and Channel button, contending that Kakuta does not inherently disclose a pane configured to display a plurality of channel identifiers. Ex-2003, ¶¶166–67. I

disagree for the same reasons discussed above. *See* Sections II.C.1.a, II.D.1.a, *supra*. A PHOSITA would understand that such functionality is inherent in IRC-based systems, which routinely presented users with lists of active channels and their corresponding identifiers. *See* Exh-1014, 4, Fig. 3, Exh-1023, 7–8.

92. Kakuta’s design plainly anticipates users switching among multiple channels, and displaying multiple channel identifiers with corresponding message windows would have been a routine and predictable design choice in the context of chat systems at the time. Thus, even if not expressly depicted, a PHOSITA would have found it obvious to implement this feature, as it represents a standard user-interface convention in multi-channel communication platforms derived from IRC technology.

c. Limitations 1[o], 15[n], and 25[n]

93. I disagree with Dr. Almeroth’s assertion that Kakuta’s “Display” control does not indicate an anchor message. Ex-2003, ¶168. Dr. Almeroth argues that pressing “Display” does not inform the user which message the thread originated from, and that there is no visible label showing that association. *Id.* This mischaracterizes the requirements of the limitation, Kakuta’s functionality, and how a PHOSITA would understand user-interface cues in IRC-derived systems. When a user clicks “Display,” Kakuta opens a selection window listing existing threads—such as “Program,” “Meeting,” and others—and upon user confirmation, the system opens the corresponding thread window (the third pane). *See* Exh-1005, Fig. 12; 9:23–31. That thread window displays the thread beginning with the first message of the discussion—the message that initiated the thread, shown at the top of the pane in Figure 7. Functionally, the “Display” control thus identifies and retrieves the thread anchored to that first message. Even though the main chat pane does not show a separate icon next to the anchor message, the “Display” control nonetheless acts directly on that thread and its originating communication, making the association inherent in the system’s operation.
94. In the background, when a user selects “Display,” the command executes the “Displaying Process Subroutine,” which acts on a specific thread ID stored in the administration table and opens the corresponding thread beginning with its first message. *See id.*, Fig. 2, 14:46–58, 15:11–22. A PHOSITA would understand that this operation necessarily references the anchor message of the thread—the first message recorded under that thread ID—because it defines the context for all subsequent messages in the subsidiary

communication. The absence of a visible label identifying that message is inconsequential; the association exists by design within Kakuta's data structures and retrieval process. If a designer wished to make that association visually explicit, adding a small indicator (e.g., a highlight, arrow, or icon next to the corresponding message) would be a trivial display-layer modification, not a substantive technical change.

95. In any event, adding such a visible indication of the anchor message would have been a straightforward and predictable enhancement that a PHOSITA could readily implement using standard GUI conventions. Interface designers routinely added context markers or highlights to indicate message associations, and Kakuta's architecture already maintains the necessary data correlation between message IDs and thread IDs (*see* Exh-1005, 6:1–21, 4:57–65). Thus, the claimed "indication that said first messaging communication is an anchor message" would have been an obvious, routine visualization of information already inherent in Kakuta's operation.

2. Limitations 1[i], 1[p], 3[a], 13[a], 15[h], 15[o], 17[a], 23[a], 25[h], 25[o], 27[a], and 28[a]

96. I disagree with Dr. Almeroth's contention that a PHOSITA would not have found it obvious to modify Kakuta to satisfy the "in response to" limitation. Ex-2003, ¶¶171–74. Dr. Almeroth argues that because Kakuta includes an intermediate confirmation window (Fig. 6), the thread window is not displayed "in response to" the selection of the "Create" control. A PHOSITA, however, would readily understand that skipping or bypassing an optional confirmation step to display a thread immediately is a routine and predictable user-interface optimization, not a substantive design change. Eliminating unnecessary confirmation dialogs to reduce user actions was a well-known refinement in GUI design that would have been obvious to apply to Kakuta's interface.
97. Kakuta itself already demonstrates both forms of interaction—multi-step and immediate. Specifically, Kakuta discloses that the "Create" command can be used independently, such that clicking "Create" without selecting any users or messages immediately opens a new, empty thread window. Exh-1005, 8:52–56. This is a direct, one-step response to a user command. A PHOSITA would therefore find it obvious to extend that immediate-response functionality to cases where a user has preselected one or more messages or participants, viewing it as an expected efficiency improvement rather than an inventive leap.

98. At the time of Kakuta’s disclosure, streamlining multi-step interactions into single-step operations was a well-recognized design trend in interface development. For example, providing “Shortcuts” was one of ten “basic characteristics of usable interfaces” taught in standard industry and academic texts, such as Nielsen’s 1994 book—“Usability Engineering.” Exh-1027, 139. Designers consistently sought to minimize intermediate windows when user intent was already clear, especially for routine or non-destructive operations. A PHOSITA implementing Kakuta’s system would have viewed replacing the two-step “Create” → OK sequence with a single-step thread creation as a straightforward usability enhancement—an ordinary design choice motivated by efficiency, not by inventive insight.
99. Dr. Almeroth asserts that Kakuta’s “Create” control cannot be modified to provide an immediate response when a user has made selections, arguing that the immediate and multi-step options are mutually exclusive. Ex-2003, ¶172. This interpretation mischaracterizes Kakuta. A PHOSITA would recognize that Kakuta’s intermediate confirmation window merely provides configuration options—such as “Quote Messages”—that could easily be automated or defaulted. *See* Exh-1005, 15:18–22. The existence of the intermediate window does not restrict system architecture but instead reflects a usability preference. Modifying or bypassing this optional dialog to streamline user flow would have been an expected optimization consistent with GUI principles emphasizing flexibility and user control.
100. Nothing in Kakuta discourages extending this same direct or automated behavior to cases involving message selection. A PHOSITA would understand that merging or skipping the confirmation window would not “break” the system—it would simply automate a step the user could otherwise confirm manually. Such modifications were commonplace and would have been an obvious refinement in pursuit of a smoother, more efficient workflow. Consistent with this understanding, Nielsen’s text states “[s]ystem-provided default values constitute a shortcut since it is faster to recognize a default and accept it than having to specify a value or an option. In many cases, users do not even need to see the default value, which can remain hidden on an option screen that is only accessed in the rare case where it needs to be changed.” Exh-1027, 142.
101. I disagree with Dr. Almeroth’s characterization of my “inherent” statement regarding Kakuta’s “confirmation window.” Ex-2003, ¶174. My opinion does not rely on the premise that the confirmation window *inevitably* results

in the display of the third pane. Rather, my point is that the confirmation window forms part of the *process* initiated by the user's selection of the "Create" control—a process that, when the user proceeds, leads to the display of the thread (third) window. The fact that a user could cancel does not break this causal relationship; optional dialogs are a routine component of user-initiated operations in graphical interfaces. A PHOSITA would recognize that such intermediate windows are transitional steps within a single functional sequence. Moreover, even if the confirmation window provides certain configuration options (such as the "Quote Messages" checkbox), those options could easily be automated, defaulted, or integrated into the primary interface without altering Kakuta's overall functionality.

102. Moreover, Dr. Almeroth's suggestion that the presence of an intermediate dialog renders the claimed functionality non-obvious is unsupported. Kakuta's confirmation window serves practical functions—configuring disclosure levels (public or private) and message quoting—but those tasks could easily be automated. A PHOSITA would have recognized that a simplified workflow removing the redundant dialog, or merging thread selection and display into a single step, would maintain Kakuta's underlying behavior while reducing user effort. These are routine refinements aimed at improving efficiency, not fundamental architectural changes.
103. Patent Owner's related argument that a PHOSITA would not have integrated Kakuta's "Display" control with its thread-selection list into a single action also fails. Paper 16, 67–68. The Petition does not allege that Kakuta's process was deficient, only that a PHOSITA would have recognized an obvious opportunity to streamline it. Combining the "Display" control and thread-selection step—so that clicking a thread title (e.g., "Program") directly opens the corresponding thread—is a predictable usability enhancement that reduces user interaction without altering system functionality.
104. A PHOSITA would further appreciate that this integration reflects how IRC-based interfaces evolved. For instance, clients allowed users to double-click a thread or username to open a related window, merging selection and activation into a single gesture. Exh-1023, 8–9. The same concept applies naturally to Kakuta's "Display" control: enabling users to click a thread title to open it in one step. Such improvements were standard practice in graphical design and require no inventive insight.

105. Finally, Dr. Almeroth’s suggestion that a PHOSITA would “avoid improving interface efficiency” is contrary to established GUI design principles. Combining sequential actions—such as thread selection and display—does not change system function; it simply improves user experience. Because Kakuta already links these actions (the “Display” control opens a list and “OK” confirms thread selection), a PHOSITA would have viewed merging them as a predictable optimization achieving the same result with fewer actions.

3. Claim 4

106. I disagree with Dr. Almeroth’s assertion that the “third user-selectable control is not identified in the Petition.” Ex-2003, ¶176. As explained above, Kakuta expressly discloses a floating toolbar (305) containing multiple user-selectable controls—including “Link,” “Display,” and “Create”—that operate on user-selected messages or threads. Exh. 1005, Figs. 7–8, 10, 12; 8:30–47; 9:23–31. A PHOSITA would understand that these toolbar controls correspond to the claimed third user-selectable control because they are invoked in conjunction with selected message content in the chat window (the purported “second pane”) to display or create subsidiary thread windows (the “third pane”).
107. I also disagree with Dr. Almeroth’s assertion that Kakuta “is designed and functions in such a way that proximity is irrelevant.” Ex-2003, ¶178. While Kakuta’s toolbar is described as floating—that is, capable of being repositioned or docked anywhere within the interface—a PHOSITA would understand that the very purpose of a floating toolbar is to permit contextual proximity and flexibility, not to eliminate it. Movable toolbars were widely used in GUI design to allow users to position controls near the content or function they most frequently act upon, enhancing efficiency and usability. A PHOSITA would therefore recognize that Kakuta’s toolbar 305 could readily be repositioned or docked near active messages, providing visual and functional association consistent with the “proximate to” limitation of Claim 4.
108. Contrary to Dr. Almeroth’s suggestion, allowing a toolbar to be docked or contextually anchored near a message would not constitute a “fundamental” change to Kakuta’s operation. It would instead represent a minor and predictable design refinement that preserves the system’s underlying architecture. A PHOSITA would have found it obvious to permit Kakuta’s toolbar to snap to or align with a message area—particularly when the

toolbar's controls (e.g., "Link" or "Display") operate directly on selected message content. Such interface flexibility was a well-established convention in GUI development long before Kakuta, enabling users to optimize workflow through spatial alignment of controls and their corresponding targets.

109. Accordingly, even if Kakuta's toolbar were initially implemented as fully floating, a PHOSITA would find it obvious to modify its behavior so that it could be positioned near the relevant messages or dynamically appear adjacent to user selections. This design pattern—contextual control positioning—was a common and predictable usability enhancement, motivated by efficiency and clarity rather than invention. Therefore, Kakuta either discloses or, at a minimum, renders obvious the claimed requirement that the third user-selectable control be displayed "proximate to said additional messaging communication."

4. Claims 9, 12, 21

110. I disagree with Dr. Almeroth's assertion that Kakuta lacks the ability to process a "request to permit a non-member user." Ex-2003, ¶181. Kakuta expressly manages user-join, departure, and channel-participation events through its administration and control modules. Exh-1005, 11:6–10; 3:56–63. A PHOSITA would recognize that these same modules inherently handle user-access requests, including those from users not yet in a channel, because that capability was fundamental to IRC-based architectures on which Kakuta builds. In standard IRC systems, the /invite command allowed an existing channel member to request that a non-member user join a private conversation. Exh-1023, 8–9, 211. A PHOSITA would therefore view Kakuta's join/leave mechanisms as naturally encompassing this same invitation-based workflow—i.e., receiving a request to permit a non-member user to receive thread messages—because it represents a straightforward implementation of standard IRC user-management logic.
111. I also disagree with Dr. Almeroth's contention that Kakuta provides no selectable user ID or nickname element. Ex-2003, ¶180. As a derivative of IRC systems, Kakuta's interface includes user identifiers displayed within user lists and message headers that are functionally linked to user-management actions such as private messaging, query initiation, and invitations. *See, e.g.*, Exh-1005, Fig. 8. IRC clients routinely allowed users to double-click or right-click a username to open a private chat or send an /invite request. Exh-1023, 203. A PHOSITA would understand that adapting

these textual commands into GUI-based clickable identifiers—as Kakuta’s system plainly could (*see* Exh-1005, 6:47–50)—was a trivial and predictable enhancement that preserves the same underlying behavior while improving accessibility within a windowed interface. Thus, implementing a clickable username in Kakuta to open a private thread or extend an invitation to a non-member would not introduce new functionality but would simply modernize an existing IRC interaction model.

112. Moreover, Kakuta’s architecture could directly reuse IRC’s invitation mechanism. Its administration module (23) and control module (32) already handle message routing, user state changes, and channel permissions “in accordance with user instructions.” Exh. 1005, 7:56–60. A PHOSITA would recognize that the same underlying process controlling channel membership could easily accommodate a non-member request—by sending a permission message or notification through the existing modules. Incorporating an explicit invitation or confirmation step would therefore have been an expected and obvious refinement, consistent with the collaborative, access-controlled structure of IRC-derived messaging systems. Nothing in Kakuta discourages or teaches away from such an implementation.
113. Finally, to the extent Claims 9, 12, and 21 are not anticipated by Kakuta (*see* Sections II.C.8, II.C.10–11, *supra*), each concern routine, predictable interface refinements—association of controls with identifying information (Claim 9), permission management for non-members (Claim 12), and thread duplication or synchronization (Claim 21)—that reflect long-established GUI and messaging design principles well known to a PHOSITA before the alleged priority date. Each limitation represents the natural evolution of familiar IRC features into graphical form.

5. Claim 13

114. I disagree with Dr. Almeroth’s assertion that there is no fifth user-selectable control that is associated with any specific type of messaging communication. Ex-2003, ¶184. Dr. Almeroth misreads both the claim language and the Petition’s mapping. Claim 13 does not restrict the number or order of user-selectable controls; it merely requires that one such control be operable to selectively display messaging communications of a specific type. Kakuta’s toolbar (305) includes multiple user-selectable controls—such as “Link,” “Display,” and “Create”—each capable of filtering or displaying different types of communications, threads, or message groupings. A PHOSITA would readily understand that these controls satisfy

the functional requirement of a fifth user-selectable control associated with a specific messaging type.

115. I also disagree with Dr. Almeroth's contention that Kakuta lacks a "specific type" filter for displaying messages. Ex-2003, ¶184. Filtering mechanisms were well-known in the art and commonly implemented in graphical messaging interfaces. It would have been obvious to a PHOSITA to incorporate standard filtering techniques—such as user-based, thread-based, or message-type filtering—into Kakuta's existing framework. These enhancements require no inventive concept; they represent routine design adaptations that allow a user to select and display messages meeting particular criteria within a communication window.
116. Kakuta's own disclosure supports this functionality. The "Link" and "Display" controls both enable the user to select and display groups of messages from specific threads or users. Exh-1005, 8:30–47; 9:23–31; Figs. 7, 10, 12. The "Link" control, for example, allows combining messages from multiple threads into a new display window, while "Display" allows a user to select which existing thread to open. A PHOSITA would understand that extending these same controls to display messages filtered by message type, author, or category—such as all subsidiary threads created by a particular user—would be a predictable and straightforward design choice, fully consistent with Kakuta's existing architecture.
117. Furthermore, Dr. Almeroth does not dispute that Kakuta's toolbar controls already operate independently and provide thread-specific functionality; he merely notes that Kakuta does not explicitly disclose a "specific type" filter. Ex-2003, ¶¶182–85. That omission does not negate obviousness. At the time of the alleged priority date, selectable filters and message-type displays were common, expected user-interface enhancements, particularly in IRC-derived systems that supported message categorization and user-based thread organization. A PHOSITA would recognize that incorporating a selectable filter into Kakuta's toolbar—such as an option to display only messages from a specific user or thread class—would have been a predictable extension of its existing control set.
118. It was also well understood in the art that good user-interface design emphasizes reducing the number of steps or clicks required to perform an operation. A PHOSITA would understand that minimizing input actions increases efficiency, improves usability, and reduces user error. Consistent with these principles, it would have been *obvious* to combine Kakuta's

“Display” control and its subsequent thread-selection step into a single, integrated operation—allowing a user to click directly on a desired thread or filter type to open the corresponding message view. This would have been an expected evolution of Kakuta’s interface, improving user experience while maintaining the same underlying functionality.

119. I also disagree with Dr. Almeroth’s assertion that showing all channels in a chat window would be “unwieldy.” Dr. Almeroth attempts to justify this claim by stating that “IRC chat systems had thousands of channels covering disparate topics.” Ex-2003, ¶¶182–85. However, a PHOSITA would understand that while some IRC servers might have thousands of channels, it was also common for IRC servers to be small and privately hosted, with very few channels. In such cases, displaying messages from all channels would not be unwieldy, and would have the advantage of allowing a user to quickly and easily peruse all messages posted across channels without the burden of having to discretely select a series of individual channels to examine recent activity. Furthermore, a PHOSITA would understand that providing an overview of messages available across all channels would be desirable and consistent with the “information seeking mantra” that has been exceptionally well known for enabling efficient information access in user interfaces since the mid 1990s. Exh-1026, 337. This “mantra”— “overview, zoom and filter, then details on demand”—advocates for user interface designs that first present an overview of all information, allowing the user to subsequently zoom and filter the information (e.g., selecting a specific channel or user), then select information for detailed scrutiny (e.g., scrolling to particular messages).

6. Claim 14

120. I disagree with Dr. Almeroth’s assertion that Kakuta cannot meet Claim 14 because it allegedly lacks a mechanism for displaying “a class of messaging communications ... generated by a particular user.” Ex-2003, ¶187. Dr. Almeroth’s argument misreads both Kakuta’s disclosure and the scope of the claim. Kakuta explicitly teaches creating threads based on messages, words within messages, or messages chosen by a user. Exh-1005, 3:43–47; 8:45–47. A PHOSITA would understand that selecting multiple messages authored by the same user inherently results in a thread composed of a class of communications generated by that user. When a user highlights messages authored by a particular participant and activates “Create” or “Link” controls, Kakuta generates a subsidiary thread containing only those selected

messages—precisely the type of class-based message grouping recited in the claim. *See* Exh-1005, 8:38–47; Fig. 7.

121. Moreover, Kakuta’s system architecture already supports classification by sender identity. Its administration tables correlate message-identifying information, user nicknames, and thread identifiers. Exh-1005, 6:23–34. A PHOSITA would understand that leveraging this existing metadata to group or filter messages by user is a straightforward and obvious extension of Kakuta’s design. Indeed, IRC systems contemporaneous with the alleged priority date routinely offered “view by user” or “filter by participant” features that allowed users to isolate messages from specific senders within a channel. Exh-1026, 337. Kakuta already teaches the ability to select messages from a specific user by selecting and threading all messages from a specific user, which can then be linked with the “Link” tool to form a subsidiary thread encompassing all of that user’s communications across threads. *See* Exh-1005, 8:38–9:4. Applying such well-known filtering logic to Kakuta’s thread architecture—so that a user could display all messages authored by a specific participant—would have been a predictable and expected improvement consistent with established GUI design practices.
122. Dr. Almeroth’s suggestion that Kakuta’s thread-creation process would always include messages from multiple users misrepresents the system’s flexibility. Kakuta allows a user to select any subset of messages prior to pressing “Create,” including all messages from a single user. Exh-1005, 8:38–47. Nothing in Kakuta limits the selection process to multiple authors or prevents user-specific filtering. A PHOSITA would recognize that threads based on message selection inherently permit user-specific groupings and that extending this mechanism to aggregate “a class of messages generated by a specific user” would require no more than reusing Kakuta’s existing thread-generation logic with a user-based selection criterion.
123. Furthermore, Kakuta’s “Link” and “Display” controls already support combining and presenting messages across multiple threads or users. Exh-1005, 8:59–9:4; 13:22–30. A PHOSITA would understand that these controls could easily be extended to include user-based filters—allowing selection of a participant’s name or ID to view all messages authored by that participant across threads. Such an enhancement would be a predictable GUI design refinement, analogous to standard filtering and aggregation tools widely used in contemporaneous messaging clients.

124. Finally, Claim 14 does not require a novel interface element or limit the “class” to an entire channel. The claim simply requires that messages belonging to a specific type—here, those generated by a particular user—can be displayed via a user-selectable control. Kakuta’s existing toolbar controls (“Link,” “Display,” “Create”) already provide user-initiated filtering and aggregation of message content. Adding an option to view all messages authored by a selected user—across one or more threads—would be a routine, predictable enhancement well within the ordinary skill of the art. As GUI designers routinely minimized redundant steps, a PHOSITA would have been motivated to incorporate this capability into Kakuta’s framework for usability and efficiency reasons.

7. Claims 23, 24, 28, and 29

125. Dr. Almeroth offers no new substantive analysis for Claims 23, 24, 28, and 29, instead incorporating his prior arguments. Ex-2003, ¶189. For the same reasons discussed above with respect to Claims 1–22, a PHOSITA would have found it obvious to implement the claimed features in Kakuta.

8. Claim 30

126. I disagree with Dr. Almeroth’s assertion that it would not have been obvious to modify Kakuta to include a “thread summary indicator” by integrating the “Display” control with the thread-selection step into a single operation. Ex-2003, ¶190. Dr. Almeroth’s analysis mischaracterizes both Kakuta’s teachings and how a PHOSITA would have approached interface optimization at the time. Kakuta’s Display process (Exh-1005, Fig. 12, 9:23–31) already links the user’s input directly to the display of a thread window based on the selected thread title. Consolidating these steps—so that a user could click directly on a thread indicator (e.g., “Program,” “Meeting”) to open the associated thread—would have been a predictable usability refinement consistent with standard GUI evolution.
127. As mentioned prior, streamlining existing controls to reduce user actions was a well-known and routine design motivation in GUI development. A PHOSITA would recognize that eliminating redundant steps—such as pressing “Display” followed by “OK”—improves efficiency without altering Kakuta’s underlying functionality. In fact, Kakuta itself demonstrates a similar principle: pressing “Create” without making a selection immediately opens an empty thread window (*id.*, 8:52–56), showing that the system already supports direct one-step display behavior. Extending this existing

framework to integrate “Display” and thread selection would therefore have been obvious, as it merely simplifies an already linked sequence of actions.

128. Dr. Almeroth’s suggestion that combining these steps would “complicate” the toolbar or fundamentally alter Kakuta’s operation is incorrect. A PHOSITA would understand that integrating selection and display functionality could be accomplished through ordinary interface design methods—such as making each thread title in the listing directly selectable or clickable—without changing the system’s architecture. This enhancement would streamline the user workflow, reduce click count, and make thread navigation more intuitive, all of which were recognized goals within the field at the time of the alleged priority date.

E. Invalidity Ground 3

1. Limitations 1[c], 15[b] and 25[b]

129. I disagree with Dr. Almeroth’s assertion that there was no reason for a PHOSITA to incorporate Matsumoto’s “channel window 36” into Kakuta. Ex-2003, ¶192. In fact, Kakuta already discloses a channel-selection mechanism—specifically, a “Channel” button and a down-arrow indicator that allow users to switch between IRC channels (Exh-1005, 6:28–33; Fig. 8). A PHOSITA would have recognized that incorporating Matsumoto’s list-based or scrollable channel display into Kakuta would represent an expected refinement to improve navigation efficiency. Doing so would not change Kakuta’s operation but would simply allow users to view and switch channels directly within the main interface, rather than toggling through a separate dialog. Integrating such a list or scrollable pane of channels is consistent with GUI conventions of the time, which emphasized minimizing window changes and improving real-time awareness of available channels.
130. Dr. Almeroth further asserts that Matsumoto’s “channel window 36” merely identifies channels already displayed and therefore cannot serve as a “channel selection pane.” Ex-2003, ¶193. This interpretation overlooks the functional equivalence between Matsumoto’s and Kakuta’s systems. Matsumoto expressly depicts a main chat window (32), a sub-chat window (34), and a channel window (36) listing multiple channels (e.g., “#CHOCOA”), each representing an active conversation area. Exh. 1006, 9:31–46; Fig. 3. A PHOSITA would have understood Matsumoto’s channel window as an organizational interface for selecting or monitoring active channels—a function materially similar to the “pane configured to display a

plurality of channel identifiers” recited in the challenged claims. Both systems use the same logical model of IRC channels as distinct conversation containers. *See* Exh-1005, 5:36–43; Exh-1006, 7:35–56. Adding Matsumoto’s channel list to Kakuta’s existing channel-management scheme would therefore have been straightforward and beneficial, enabling quick switching between channels while maintaining context within the same main display.

131. Dr. Almeroth’s suggestion that Matsumoto’s features would have “no utility” within Kakuta’s framework is incorrect. The ability to view and manage multiple channels from one interface directly supports Kakuta’s stated goal of simplifying conversation display and improving readability across concurrent communications. A PHOSITA would appreciate that incorporating Matsumoto’s scrollable list would align with Kakuta’s objective “to make conversational display easy to view” (Exh-1005, 1:9–19) by allowing users to stay within one screen when navigating between channels. Indeed, contemporary IRC clients—such as XChat and mIRC—used nearly identical channel lists to streamline navigation (Ex-2008; Exh-1023 8). Thus, integrating Matsumoto’s “channel window 36” into Kakuta’s architecture would have been an obvious and beneficial design refinement, not a fundamental alteration.

2. Claims 13–14, 23–24, 28–29

132. I disagree with Dr. Almeroth’s assertion that Matsumoto’s “filter destination channel” cannot function as a user-selectable control. Ex-2003, ¶195. A PHOSITA would recognize that Matsumoto’s filtering mechanism allows users to define and select message groups based on predetermined criteria—such as channel, user, or message content—and that such filters inherently serve as controls for displaying specific message classes. Integrating Matsumoto’s filtering capability into Kakuta’s IRC-based framework would have been a straightforward extension, enabling the display of selected threads or user-defined message categories within Kakuta’s thread window. These filters perform exactly the control function contemplated by the claims: displaying communications of a “specific type” or “class” through user selection.
133. Dr. Almeroth further contends that combining Kakuta and Matsumoto would not result in a system that displays messages from “a multitude of different channels” and that a PHOSITA would not deviate from Kakuta’s single-channel design. Ex-2003, ¶195. This argument is misplaced. A PHOSITA

would not view Matsumoto's teachings as requiring Kakuta to intermingle channels; rather, Matsumoto illustrates how correlated or filtered messages could be displayed concurrently, within user-defined parameters. *See* Exh-1006, 2:65–3:4, 3:51–63, 16:51–65. Incorporating such correlation or filtering logic into Kakuta would allow selective presentation of messages—e.g., threads containing communications from particular users or topics—without departing from Kakuta's single-channel architecture. This selective-display concept is fully compatible with Kakuta's thread model and would have been recognized as an obvious enhancement to increase flexibility in organizing related discussions.

134. Dr. Almeroth also criticizes the combination as unspecified, asserting that the Petition “does not explain what the combination[] would look like or how it would meet each limitation[.]” Ex-2003, ¶195. In fact, the combination would look familiar to any PHOSITA: Kakuta's main interface would remain unchanged except for the addition of Matsumoto's channel-list pane along one edge of the screen (or in a separate window), and the addition of Matsumoto's “correlate” control. The requisite filtering options to control which thread or user classes appear in the display are already present in Kakuta, as exemplified by item 302 of Kakuta's Fig. 5 and selection window 307 of Kakuta's Fig. 10. Such a modification would require no structural redesign of Kakuta's underlying modules and would be an intuitive application of known GUI composition principles (e.g., window docking and list-view integration).
135. Dr. Almeroth's critique of Matsumoto's “Correlate” control also misconstrues its function. Ex-2003, ¶196. While Matsumoto's “Correlate” control activates stored “sending conditions” to link or forward messages across channels, its operation demonstrates the exact type of user-initiated correlation control recited in the claims. Exh-1006, 16:51–65. A PHOSITA would recognize that Kakuta's “Link” control (which already merges thread content) and Matsumoto's “Correlate” control perform analogous functions: both act as user-selectable controls to group and display related messages according to logical relationships. Incorporating Matsumoto's correlation logic into Kakuta's thread-linking process would simply extend its capabilities to include broader relationships—such as cross-channel or multi-thread associations—consistent with the predictable use of known message-filtering techniques.

3. Motivation to Combine Kakuta in view of Matsumoto

136. I disagree with Dr. Almeroth's assertion that a PHOSITA would lack motivation to combine Matsumoto with Kakuta because, as he claims, the two systems purportedly address "fundamentally different problems." Ex-2003, ¶¶ 197–201. That conclusion misconstrues both references. Kakuta and Matsumoto operate in the same technological field—networked, IRC-based chat environments—and share the same overarching goal: improving the readability, organization, and navigation of simultaneous conversations. Exh-1005, 1:6–23; Exh-1006, 1:52–54. Each reference builds upon standard IRC architectures and interface conventions, as Dr. Almeroth concedes, employing multiple panes or windows to manage ongoing message streams. *See* Ex-2003, 61, 198. A PHOSITA would naturally consider their interface concepts complementary rather than incompatible.
137. Dr. Almeroth focuses narrowly on Matsumoto's main channel window 31 and sub-chat window 34, but a PHOSITA would also recognize the relevance of Matsumoto's channel window 36. For instance, Matsumoto's channel window 36 allows the user to select channels formed as a combination of multiple channels, and the channel window 36 remains persistently available, without the need to drop down the channel list from Kakuta's ComboBox. Even if window 36 lists sub-channels or correlated channels, it provides a logical, scrollable pane for displaying multiple active channels. Integrating Matsumoto's channel window 36 into Kakuta's thread-based layout would have been a natural, predictable design enhancement, extending Kakuta's single-channel threading model into a multi-channel context while maintaining Kakuta's modular interface organization. Because both systems rely on pane-based GUI structures, a PHOSITA would view such an addition as a straightforward adaptation to improve usability.
138. Nothing in either Kakuta or Matsumoto teaches away from this combination. Both systems are modular and designed for interface scalability. Kakuta's toolbar and windows can be freely repositioned (Exh-1005, 4:42–47), while Matsumoto introduces additional panes to support concurrent channel monitoring. A PHOSITA would have recognized that these modular display elements were compatible and could easily be integrated through routine GUI composition techniques familiar in IRC clients at the time of the alleged priority date. *See* Exh-1023, 8. The result would be a predictable improvement: a unified interface allowing users to navigate channels and threads more efficiently without switching screens.

139. Moreover, a PHOSITA would have been motivated by well-known usability principles driving GUI evolution—specifically, the goals of reducing user navigation steps, increasing contextual awareness, and consolidating related communication areas within a single interface. Integrating Matsumoto’s scrollable list of active channels into Kakuta’s interface would address these same objectives by providing a persistent view of available channels and threads. This integration would have improved workflow and user experience while preserving each system’s original purpose and functionality.
140. Dr. Almeroth’s argument that such a combination would “complicate” Kakuta is unfounded. Adding a small pane or docked list of channels is not a fundamental alteration of Kakuta’s architecture but a minor interface refinement, consistent with contemporaneous IRC clients that routinely displayed channel lists alongside main chat windows. A PHOSITA would have considered this a low-risk, high-value improvement that aligns with the common design trend toward integrated, multi-view chat environments. As a result, a PHOSITA would have been fully motivated to combine Matsumoto’s channel window 36 with Kakuta’s thread-based IRC system to enhance navigation and display management—yielding a predictable and beneficial outcome without requiring inventive insight.

F. Invalidity Ground 4

1. Claims 1, 15, and 25

a. Chang Discloses a “Second User-Selectable Control”

141. I disagree with Dr. Almeroth’s assertion that Chang does not disclose a “second user-selectable control” or that its controls are unrelated to specific messaging communications. Ex-2003, ¶¶202–03. A PHOSITA would recognize that the decision to position user controls relative to messages in IRC-based or chat-style interfaces is primarily a *design choice*, not a technical limitation. Adjusting the position of a control—whether through a toolbar, contextual menu, or touch-triggered overlay—to align with the message being acted upon was a well-known and predictable GUI convention at the time. Such positioning decisions were routinely made to improve usability and message association, not to introduce new functionality.

142. In my opinion, Chang explicitly teaches controls displayed proximate to the communication being acted upon. For instance, Chang discloses that a long-press or touch action on a specific message area brings up a contextual pop-up menu offering options such as displaying or managing a related private chat. Exh-1007, ¶¶ [0270]–[0271]. This functionality is inherently message-specific: the user triggers the control from a defined message location, and the menu appears directly adjacent to that message. Chang also describes proximity-based touch sensing that activates contextual actions when the user interacts with message content or private chat areas. A PHOSITA would therefore interpret Chang as teaching controls that appear in close proximity to, and operate upon, the message under interaction.
143. Similarly, Kakuta’s toolbar 305 provides functionally equivalent message-level operations through a different interface modality. In Kakuta, a user selects one or more messages and then presses “Create” or “Display” to open a related thread or private communication window. Exh-1005, 8:38–47; 9:23–31. Both systems thus implement the same underlying functionality: a control associated with a message that, when selected, initiates a related conversation window. Incorporating Chang’s contextual menu approach into Kakuta—by repositioning Kakuta’s floating toolbar controls into a pop-up or proximity-based menu—would have been an obvious and routine interface optimization to enhance usability, not a change to core functionality.
144. Dr. Almeroth argues that Chang also discloses only a “multi-step process” requiring a user to first touch a private chat area before triggering the pop-up menu. Ex-2003, ¶206. However, as discussed above, a PHOSITA would have recognized that consolidating such sequential actions into a single, more direct interaction would be an obvious and routine interface optimization consistent with established GUI design principles.

2. The Proposed Combination Would Result in The Claimed Invention

145. I also disagree with Dr. Almeroth’s conclusion that combining Chang with Kakuta would not yield the claimed invention. Ex-2003, ¶¶208–11. Chang’s contextual pop-up mechanism and Kakuta’s message-selection workflow are complementary, not contradictory. Kakuta already discloses a system where selecting a message or group of messages activates controls (e.g., “Create,” “Display,” or “Link”) that generate or display related thread windows. Incorporating Chang’s contextual display of those controls directly adjacent

to a selected message would have been a predictable, routine enhancement aimed at improving clarity and reducing user clicks.

146. A PHOSITA would recognize that the combination preserves both systems' functionality: Kakuta provides the threading architecture and control operations, while Chang offers a more direct, context-sensitive presentation of those controls. The result is an interface that allows a user to invoke Kakuta's "Display" or "Create" actions through a proximity-based contextual menu—precisely the type of user experience optimization common in GUI evolution. Such integration would simply unify the two well-known interaction models (toolbar buttons and pop-up menus) without altering Kakuta's underlying processes for thread creation and display.
147. Dr. Almeroth's suggestion that the combination would "complicate" the toolbar or render Kakuta's system inconsistent with the claims is misplaced. Ex-2003, ¶209. Incorporating Chang's pop-up structure does not remove or fundamentally change Kakuta's floating toolbar; rather, it provides an alternative invocation path for the same functions. A PHOSITA would understand that contextual menus were a mature, interchangeable GUI feature by the priority date, routinely used to present message-related commands in chat and messaging systems. Replacing or supplementing a static toolbar with contextual controls would have been an expected refinement—motivated by usability, not invention.
148. Furthermore, the contextual menu of Chang appears "proximate" to the selected message, establishing both a visual and functional association between the message and the control. Combined with Kakuta's controls (e.g., "Display") would then open the related thread window (the "third pane")—the same end result achieved by Kakuta's toolbar, but with improved contextual clarity. This modification is exactly the kind of predictable GUI optimization that a PHOSITA would find obvious.

3. Claim 4

149. I disagree with Dr. Almeroth's assertion that Chang fails to disclose a control displayed proximate to an additional messaging communication. Ex-2003, ¶213. Dr. Almeroth's arguments mischaracterizes both Kakuta's and Chang's teachings. As I explained above, both references disclose controls that are functionally and contextually tied to specific messages being acted upon. *See* Section F.1.a., *supra*. The placement of those controls—whether in a floating toolbar, contextual pop-up, or adjacent to the message—is a

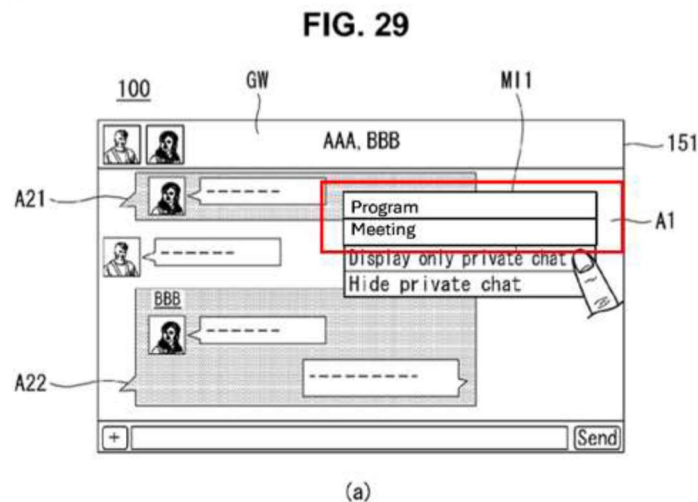
matter of interface design choice, not new functionality. Adjusting the spatial relationship between a message and its associated control would have been a routine, predictable GUI design optimization understood by a PHOSITA.

150. In particular, Chang explicitly discloses proximity-based controls triggered by user interaction with specific messages. Chang teaches that a long press on a message or message area brings up a contextual pop-up menu directly over or adjacent to that message, offering options such as “Display only private chat” or “Hide private chat.” Exh-1007, ¶¶[0270]; Fig. 29. This is an example of a control displayed proximate to the communication being acted upon. Chang further describes proximity-based touch sensing that triggers controls and corresponding visual indicators based on the sensed proximity touch action and proximity touch position. Exh-1007, ¶¶[0071]. A PHOSITA would interpret these disclosures as teaching controls that appear in close visual proximity to the message or chat element the user interacts with.
151. Functionally, Chang and Kakuta perform the same user operation. In both systems, the user selects a message and triggers a control linked to that specific communication to open or manage a related thread or private chat window. Kakuta achieves this through its “Create” or “Display” buttons on a floating toolbar, which operate on selected messages in the chat window. Exh-1005, 8:38–47; 9:23–31. Chang achieves the same functionality by presenting context-specific controls immediately adjacent to the selected message. Exh-1007, Fig. 29. The difference lies only in presentation: Kakuta uses toolbar-based buttons, while Chang integrates the same message-level functions into a proximity-based pop-up menu. A PHOSITA would view these as equivalent interaction models implementing the same logical association between a control and the corresponding message.
152. Relocating a message-level control from a toolbar to a contextual pop-up is a well-understood GUI refinement, aimed at improving clarity, efficiency, and reducing user effort. Exh-1003, ¶¶[273–74]. By the alleged priority date, contextual menus and right-click/long-press actions were standard mechanisms for invoking commands directly associated with selected items. A PHOSITA would recognize that integrating Kakuta’s “Create” or “Display” controls into Chang’s contextual pop-up interface would yield the same result—allowing the user to open a related thread or private window—while streamlining the workflow into a single, intuitive interaction.

153. Dr. Almeroth’s argument that Chang’s control is not “proximate” to any particular message overlooks the inherent spatial and functional linkage between a message and its contextual pop-up. Ex-2003, ¶213. Chang’s pop-up appears only upon interaction with a specific message or private chat area and is rendered immediately adjacent to that element, thereby creating the required visual and operational association. A PHOSITA would understand this as a classic example of a control displayed “proximate to” the relevant communication.

4. Claims 8 & 20

154. I disagree with Dr. Almeroth’s assertion that the combination of Kakuta and Chang fails to disclose an “indicator identifying an additional messaging communication as associated with a subsidiary communication thread.” Ex-2003, ¶214. Claims 8 and 20 merely require that a user be able to discern when a message is associated with a subsidiary thread; they do not specify any particular visual form or placement for the indicator. A PHOSITA would understand that the claimed indicator may correspond to the menu items displayed within Chang’s pop-up menu—such as selectable thread options—combined with, for example, Kakuta’s “Program” and “Meeting” thread titles that identify the associated subsidiary conversations.



Exh-1007, FIG. 29 (combination image depicting an example selectable menu with a selection of each thread.).

155. These options appear when a user performs a long-press or touch on a specific private-chat area (A21 or A22), which the controller then recognizes as the user’s contextual focus. *See* Exh-1007, Fig. 29. The pop-up menu

itself is the indicator: it signals to the user that the message or region pressed is associated with a subsidiary conversation and provides direct access to that conversation through selectable menu entries. *See id.*

156. Dr. Almeroth’s criticism—that the pop-up appears only “generally” over a shaded region rather than over an individual message—does not undercut this functionality. Each shaded region in Chang (A21 and A22) corresponds to a distinct conversation context, containing the relevant messages that form that subsidiary thread. When the pop-up menu appears above that region, it visually and functionally identifies that set of messages as part of a separate conversation. A PHOSITA would readily understand this interaction as a clear indicator that “said additional messaging communication is associated with a subsidiary communication thread,” exactly as required by the claims.
157. Further, the claim requires only that the indicator be displayed “while said additional messaging communication is displayed.” Thus, the indicator need not be permanent or continuously visible; a temporary display—such as a contextual pop-up menu—is fully sufficient. Chang’s pop-up appears when the relevant communication is displayed and disappears once the interaction ends, matching this requirement precisely. A PHOSITA would recognize that such transient, context-activated indicators are a common and intuitive form of message-level feedback in GUI design, providing just-in-time cues without persistent screen clutter.
158. Kakuta provides the foundation for this approach by providing the underlying thread structure: a main message pane and a subsidiary thread pane that together allow a message and its related conversation to be viewed concurrently. Exh-1005, 8:38–47, Fig. 7. Integrating Chang’s contextual pop-up indicator into this framework would simply supply the user with an additional, direct method for recognizing and invoking subsidiary threads tied to specific messages—an obvious and predictable enhancement of Kakuta’s existing display system.

5. Claim 12

159. I disagree with Dr. Almeroth’s assertion that the combination of Kakuta, Matsumoto, and Chang fails to teach or render obvious a request to permit a non-member user to receive messages from a subsidiary communication thread. Ex-2003, ¶¶215–16. Dr. Almeroth argues that I “failed to explain what the combination would look like or how it would operate,” and that reliance on Chang’s invitation of “another party” is insufficient. *Id.*, ¶215.

This is incorrect. As explained in detail in my prior declaration (Exh-1003, ¶¶282–85), Chang explicitly discloses a process by which a user may request to add another participant to an ongoing chat. For example, Chang teaches that “when a request for adding another party is received, the controller 180 controls to display a list L of another parties that can be added in a partial area of the screen. Thereafter, when another party L1 included in the list L is touched and dragged to a chat window by the user, the controller 180 selects a chat group based on a position at which another party L1 is dragged.” Exh-1003, ¶¶282–85 (citing Exh. 1007, ¶¶[0281]–[0282]; Figs. 31–32). This disclosure describes a user-initiated request to permit a non-member user to join and receive messages in an existing conversation.

160. Claim 12 requires only a mechanism or process by which a user can bring a new participant—someone not already part of the ongoing conversation—into a subsidiary thread or private chat. A PHOSITA would understand this limitation to encompass any user action that extends message-receiving privileges to a non-member participant, whether through an invitation, a permission adjustment, or inclusion in an access list.
161. The proposed combination would incorporate a familiar and predictable graphical element—a dialog box or invitation window—that allows a user to request another participant’s inclusion in an existing thread or chat. A PHOSITA would understand that such an access-request mechanism, consistent with Chang’s “add-another-party” process, represents a straightforward and obvious implementation within Kakuta’s thread-based architecture.
162. Kakuta provides the underlying access-control framework. It expressly distinguishes between public threads (visible to all channel members) and private threads (visible only to selected users). Exh-1005, 3:56–63; Fig. 3. This distinction necessarily requires the system to determine which users can view and receive messages within a thread—precisely the access-management behavior described in Claim 12. Kakuta’s interface allows a user creating a thread to define its “disclosure level,” thereby deciding who may participate. Exh-1005, 15:18–22. A PHOSITA would recognize that this capability inherently supports adjusting permissions to include additional users after thread creation, just as many IRC systems allowed real-time updates to channel membership. *See* Sections II.C.10, II.D.4, *supra*.

163. Chang complements this framework by expressly teaching a user-initiated invitation mechanism for adding participants to a chat. As shown in Figure 31 of Chang and the accompanying description (Exh-1007, [0276]–[0285]), the controller (1) receives a request to add another party (S501), (2) displays a list of available users (S502), and (3) adds the selected participant to the chosen chat area (S505). *Id.*, Fig. 31. The corresponding examples in Figs. 32–33 show the user dragging a contact into a group or private chat area, after which the new participant begins receiving the ongoing messages in that area. This sequence discloses a user-initiated request that permits a non-member to receive messages from a subsidiary communication thread.
164. A PHOSITA would understand that Kakuta’s and Chang’s features are complementary implementations of the same fundamental design principle: managing participation and visibility of thread messages. Kakuta governs access through disclosure settings (public/private), while Chang enables dynamic inclusion through its “add-another-party” operation. Both approaches determine which users may receive thread communications. Combining these well-known techniques would have been an expected refinement to enhance usability, flexibility, and user control.
165. Kakuta’s modular architecture—particularly its administration module 23 and control module 32—already processes user join/leave events and routes messages to authorized participants. Exh. 1005, 11:6–10; 7:57–60. A PHOSITA would recognize that implementing Chang’s invitation feature within this framework would require only a routine software enhancement, such as an “invite” button or drag-and-drop action mapped to Kakuta’s existing user-management logic. This mirrors the well-established IRC “/invite” command, which allows a user to request that a non-member join a private channel. The same functional concept would apply to adding a non-member to a thread within Kakuta’s GUI—a predictable and straightforward adaptation well within ordinary skill.
166. Chang also makes clear that when a new participant is added, that user begins receiving the same messages as existing participants. For example, Figures 32 and 33 show that once the user L1 is dragged into a group or private chat area, the controller “adds the selected another party as chat another party of the selected chat area” and transmits messages accordingly (Exh-1007, ¶¶[0279]–[0285]). A PHOSITA would recognize this as precisely the claimed process—a request to permit a non-member user to receive messages from a subsidiary communication thread.

6. Motivation to Combine Kakuta in view of Matsumoto and Chang

167. I disagree with Dr. Almeroth's assertion that the proposed combination of Kakuta, Matsumoto, and Chang is hindsight-driven or technologically incompatible. Ex-2003, ¶¶219–25. In reality, all three references operate in the same field—real-time, text-based chat systems—and address complementary aspects of the same well-known usability problem: managing multiple, overlapping conversations in an efficient, intuitive, and visually organized interface. Kakuta enhances traditional IRC with threaded subsidiary windows to clarify discussions (Exh-1005, 1:6–23). Matsumoto improves IRC usability by enabling simultaneous monitoring and navigation of multiple channels through its channel window 36 (Exh-1006, 1:47–2:15). Chang, in turn, introduces intuitive, message-specific controls for starting or managing private chats using proximity-based or contextual input mechanisms (Exh-1007, ¶¶[270]–[271]). Each reference contributes an incremental, modular solution toward the same design goal—simplifying navigation and conversation organization in multi-party chat systems.
168. A PHOSITA would have been motivated to combine these features because they address the same underlying design challenges from complementary angles. Integrating Matsumoto's channel window 36 with Kakuta's thread-based IRC framework would naturally extend Kakuta's single-channel design into a multi-channel environment, improving situational awareness without altering its fundamental architecture. Incorporating Chang's contextual controls would then add a familiar, intuitive mechanism for users to initiate, open, or manage threads directly from message interactions. Together, these elements enhance the same dimensions of usability—efficiency, clarity, and organization—that define well-established GUI improvement principles.
169. Each reference's contribution fits seamlessly into the modular, pane-based architecture that characterized IRC clients. Kakuta's interface already comprises distinct panes for channels, messages, and threads. Matsumoto's channel window 36 introduces an additional persistent list-style pane for multi-channel navigation—an element fully compatible with Kakuta's existing framework. Chang's contextual menu or proximity-based controls, meanwhile, operate at the message level and would simply replace or augment Kakuta's floating toolbar with an alternative invocation method. These interface components are visually distinct, independently functional, and easily integrated within a shared display environment—a direct

reflection of the pane-based modularity common to IRC clients of the early 2000s.

170. Dr. Almeroth's claim that the combination is hindsight-driven misinterprets the relationship among these references. *See* Ex-2003, ¶224. A PHOSITA would not need the claims of the '731 Patent to recognize that multi-channel management, message-level controls, and thread-based organization were complementary and convergent design trends in GUI chat systems. Matsumoto and Kakuta both build directly on IRC architectures and aim to improve message readability and navigation; Chang addresses the same problems in the context of mobile or compact-screen interfaces. The proposed combination merely unites these compatible teachings in a predictable way to achieve an improved chat experience.
171. Dr. Almeroth's claim that Chang cannot reasonably be applied because it involves mobile apps is incorrect. Ex-2003, ¶¶221. A PHOSITA would understand that Chang's input method—touch, tap, or long press—is conceptually identical to standard mouse- or keyboard-based operations such as right-clicking or selecting with a cursor. Indeed, “mouse emulation,” which allows mouse actions such as click, double-click, and right-click to be emulated on a touch screen have been routine and conventional since at least the early 2000s. Exh-1028, 1–2. Translating Chang's proximity-based controls into a desktop IRC context would require no inventive skill; it would simply replace a “touch” gesture with a “click” or “hover” action, consistent with common GUI adaptation practices. The substance of Chang's disclosure—contextual, message-specific controls for managing conversations—applies equally across devices and input modalities.
172. Moreover, the combination of Kakuta, Matsumoto, and Chang aligns with long-standing principles of GUI evolution: reducing user effort, improving visual organization, and streamlining interaction flow. Each reference improves one dimension of the user experience—Kakuta simplifies message context, Matsumoto enhances channel awareness, and Chang refines user control. A PHOSITA would naturally integrate these improvements to achieve a more cohesive and efficient chat interface, precisely because they are incremental, modular, and independently valuable.
173. Further still, Kakuta discloses that its administration tables allow threads to be private or public, but it does not disclose the user interface mechanisms for configuring these settings. A PHOSITA would therefore be motivated to examine Chang, which provides exactly these disclosures, including

interfaces for designating messages as part of a private chat, adding users to private chats, and configuring the display of private chats.

174. Finally, neither Matsumoto nor Chang teaches away from combination. Matsumoto explicitly promotes flexible channel management and synchronized multi-window viewing. Kakuta emphasizes clarity through modular display organization. Chang, for its part, advocates user-friendly, context-driven controls to streamline interaction. None of these references discloses or implies any incompatibility with the others. To the contrary, all three reflect the same trajectory in interface design—toward unified, multi-pane chat environments with accessible, intuitive message-level controls.
175. Accordingly, a PHOSITA would have been fully motivated to combine the teachings of Kakuta, Matsumoto, and Chang to yield predictable, incremental improvements—integrating Matsumoto’s channel navigation pane, Kakuta’s threaded discussion model, and Chang’s contextual controls to enhance usability and organizational clarity. Far from being hindsight reconstruction, this combination reflects standard GUI development practices and directly addresses the same long-recognized challenges in multi-user, real-time messaging systems.

G. Invalidity Ground 5

1. Claim 30

176. I disagree with Dr. Almeroth’s assertion that Chang “does not teach that the pop-up menu is associated with any ‘message.’” Ex-2003, ¶ 217. Dr. Almeroth misreads Chang’s disclosure and overlooks how its interaction model directly establishes message-level association. Chang explicitly describes a user interface where a long-press or touch on a specific message area triggers a contextual pop-up menu offering options to display or manage related private conversations. Exh-1007, ¶¶ [0270]–[0271], Fig. 29. The pop-up appears in response to a gesture applied to that particular message region (A21, A22), providing the very type of message-specific control the claims require. A PHOSITA would recognize that this behavior constitutes an association between the message under interaction and the resulting control, since the menu appears only upon interaction with that message and nowhere else in the interface. This is the same user-action model later adopted in modern threaded chat clients—clicking or pressing a message to bring up message-specific thread options.

177. Dr. Almeroth also asserts that “nothing in Chang’s menu associates the menu options with any particular message,” and therefore that in the proposed combination, “nothing ... would be associated with any particular message.” Ex-2003, ¶218. This is incorrect. In Chang, the shaded private-chat areas A21 and A22 each correspond to a distinct conversation context initiated from a specific user interaction. When a user touches one of these areas, the system displays a context-specific menu directly over that region, not a generic menu floating elsewhere on screen. A PHOSITA would recognize that this design inherently associates the displayed controls with the message area the user acted upon—exactly the message-specific linkage that the claim language describes. The fact that multiple private-chat areas exist does not make the control “non-specific”; rather, it confirms that each menu corresponds to its respective message group or conversation area.
178. Dr. Almeroth’s argument that the combination “would merely list all available threads” misreads both Kakuta and Chang. Chang provides the ability to invoke a thread-specific control from a message itself. In Chang, when a user long-presses a specific message, the system responds contextually, presenting options to display or manage the conversation linked to that message—functionally acting as a thread summary indicator renderable proximate to the first messaging communication. Integrating this contextual, message-specific action from Chang into Kakuta’s thread system would not create a global thread list but rather allow each message to serve as an entry point to its corresponding thread.
179. In Kakuta, the thread list in Figure 12 already functions as a ***thread summary indicator***: each listed thread title (“Program,” “Meeting,” etc.) corresponds to a specific conversation that the user may open for display. Exh-1005, 9:23–31; Fig. 12. Chang teaches linking the contextual menu to the message interaction. Thus, the combination produces an interface where a user can long-press (or click) a message to open its corresponding thread window—precisely the claimed “indicator ... renderable within the messaging pane proximate the first messaging communication.” A PHOSITA would see this as a routine and predictable GUI improvement consistent with established design goals of reducing navigation steps and enhancing contextual clarity.
180. Dr. Almeroth’s view that Chang’s pop-up menu would remain independent of any particular message ignores how a PHOSITA would adapt it. In GUI design, translating Chang’s proximity-based touch interaction into Kakuta’s desktop environment would simply involve triggering the same contextual

menu via a mouse-click or hover event over a message, rather than anywhere in the chat region. Such a translation from touch to click is trivial and would have been entirely expected. Once implemented, the control would clearly be renderable proximate to the acted-upon message—fulfilling the same structural and functional association that the claims require.

181. Furthermore, the combination of Kakuta and Chang yields predictable results under well-known GUI design principles. Kakuta already provides the functional concept of thread indicators (thread titles), while Chang provides the message-specific invocation mechanism. Integrating these two elements would give users a direct, message-linked indicator that both identifies and opens the related thread. A PHOSITA would be motivated to combine these features because doing so increases efficiency, improves visual association, and simplifies user interaction. Neither reference teaches away from such integration; to the contrary, both emphasize usability and clarity in multi-threaded communication displays.

2. Motivation to Combine

182. For the reasons discussed above and in the preceding motivation sections, a PHOSITA would have found ample motivation to combine the teachings of Kakuta and Chang. Both references address the same field and design goals—clarity, efficiency, and intuitive message management in real-time chat interfaces—and their features are complementary and easily integrated. As explained in Section II.E.3, incorporating Chang’s message-specific, proximity-based controls into Kakuta’s threaded IRC framework represents a straightforward, predictable refinement consistent with well-established GUI design principles.

III. Conclusion

183. For the foregoing reasons, I believe that Claims 1-30 are anticipated and/or rendered obvious by Kakuta, Kakuta in light of Matsumoto, Kakuta in light of Matsumoto in further view of Chang, or Kakuta in view of Chang.
184. I reserve the right to supplement my opinions in the future to respond to any arguments raised by Patent Owner or its experts and to take into account new information that becomes available to me.

185. I declare under penalty of perjury that all statements made herein are of my own knowledge and are true and correct and were made with the knowledge that willful false statements are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: November 5, 2025



Andrew Cockburn

APPENDIX A: Information Considered

<i>Petitioners' Exhibit No.</i>	<i>Description</i>
1001	U.S. Patent No. 11,973,731 to Cohen
1002	File History for Application No. 18/535,572
1003	Declaration of Andrew Cockburn
1004	Curriculum Vitae of Andrew Cockburn
1005	U.S. Patent No. 6,630,944 (“Kakuta”)
1006	U.S. Patent No. 7,426,540 (“Matsumoto”)
1007	U.S. Patent Application Publication No. 2013/0069969 (“Chang”)
1008	Return of Service in Related Action
1009	Complaint in Related Action
1010	Scheduling Order in Related Action
1011	Provisional Application No. 62/253,229
1012	John E. Grayson, Python and Tkinter Programming, (2000)
1013	Apple, Human Interface Guidelines: The Apple Desktop Interface (1987)
1014	Quentin Jones, et al. “Empirical Evidence of Information Overload Constraining Chat Channel Community Interactions.” <i>In Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work (CSCW ‘08)</i> , Association for Computing Machinery, New York, NY, USA, 323-332 (2008)
1015	Andrew Cockburn and Saul Greenberg, “Making Contact: Getting the Group Communicating with Groupware.” <i>In Proceedings of the conference on Organizational computing systems (COCS ‘93)</i> , Association for Computing Machinery, New York, NY, USA, 31–41 (1993)

<i>Petitioners' Exhibit No.</i>	<i>Description</i>
1016	Thomas F. Stafford and Dennis Gonier, <i>What Americans Like About Being Online</i> , 47 Communications of the ACM, 107, 107-12 (2004)
1017	Petitioners' Reasons for Parallel Petitions and Petition Ranking
1018	'731 Patent Claim Elements
1019	Declaration of Christopher Kao In Support of Petitioner's Unopposed Motion for Admission Pro Hac Vice of Christopher Kao
1020	Declaration of Brock S. Weber In Support of Petitioner's Unopposed Motion for Admission Pro Hac Vice of Brock S. Weber
1021	Reply Declaration of Andrew Cockburn For <i>Inter Partes</i> Review of U.S. Patent No. 11,973,731
1022	Transcript of Deposition of Kevin C. Almeroth, October 28, 2025
1023	<i>mIRC User Guide</i> (Sept. 9, 2016), available at https://web.archive.org/web/20160909085546/https://www.mirc.com/help/mirc.pdf#expand .
1024	Antony Fountain, Jeremy Huxtable, Paula Ferguson & Dan Heller, <i>Motif Programming Manual, Open Source Edition</i> , Vol. 6A (Dec. 2001), available at https://www.ist.co.uk/motif/download/6A/6A_book.pdf
1025	John E. Grayson, <i>Python and Tkinter Programming</i> 16, 154–55, 365–66 (2000).
1026	Ben Shneiderman, <i>The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations</i> (Dep't of Computer Sci., Human-Computer Interaction Lab & Inst. for Sys. Rsch., Univ. of Md., College Park, 1996), available at https://doi.ieeecomputersociety.org/10.1109/VL.1996.545307 .

<i>Petitioners' Exhibit No.</i>	<i>Description</i>
1027	Jakob Nielsen, <i>Usability Engineering</i> , ch. 5 (Morgan Kaufmann Publishers Inc. 1994), available at https://dl.acm.org/doi/10.5555/2821575 .
1028	<i>New Ways to Interact with Your Tablet PC</i> (Aug. 25, 2004), available at https://web.archive.org/web/20051221130655/http://www.microsoft.com/windowsxp/tabletpc/evaluation/overviews/pcinteract.msp .

<i>Wrinkl's Exhibit No.</i>	<i>Description</i>
2001	TechCrunch Article -- "Slack launches threaded messaging to take conversations off to the side" available at https://techcrunch.com/2017/01/18/slack-launches-threadedmessaging-to-take-conversations-off-to-the-side/
2002	Hubert Florin, <i>Threads in Slack A: Long Design Journey (Part 1 of 2)</i> , Slack Design, available at https://slack.design/articles/threads-in-slack-a-long-designjourney-part-1-of-2/
2003	Expert Declaration of Kevin C. Almeroth, Ph.D.
2004	CV of Kevin C. Almeroth, Ph.D.
2005	Wrinkl Wins MediaPost's 2018 Appy Award, May 16, 2018, available at https://www.globenewswire.com/newsrelease/2018/5/16/1507676/0/en/Wrinkl-Wins-MediaPost-s-2018-Appy-Award.html
2006	EXHIBIT A, Initial Claim Chart Relating the Accused Product to the Asserted Claims of U.S. Patent No. 11,973,731
2007	Response to Non-Final Office Action dated March 9, 2020, filed June 8, 2020 in US Patent Application No. 16/399,349

<i>Wrinkl's Exhibit No.</i>	<i>Description</i>
2008	XChat: Channel List (nexlab.azzurra.org), available at https://commons.wikimedia.org
2009	Davey Alba, Slack's New Threaded Messages Tame Your Meandering Chats, WIRED, January 18, 2017
2010	U.S. Patent No. 11,539,649

<i>Papers</i>	<i>Description</i>
16	Patent Owner Response