

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner,

v.

DIALECT, LLC,
Patent Owner.

IPR2024-00752
Patent 8,849,652

REPLY DECLARATION OF CHRISTOPHER SCHMANDT

TABLE OF CONTENTS

	Page
I. PERSON OF ORDINARY SKILL IN THE ART DEFINITION	3
II. RESPONSE TO MR. OCCHIOGROSSO RE “TECHNICAL INTRODUCTION TO DIALECT’S ’652 PATENT”	3
III. CLAIM CONSTRUCTION	4
IV. RESPONSE TO MR. OCCHIOGROSSO RE “PETITIONER’S MERITS CHALLENGES”	4
A. Response to Mr. Occhiogrosso re “Overview”	4
B. Response to Mr. Occhiogrosso re “Coffman”	4
C. Response to Mr. Occhiogrosso re “Petitioner’s Ground 1 Fails to Show the Synchronization of Context Stacks or Context Sets of Independent Claims 1, 14, and 20”	5
D. Response to Mr. Occhiogrosso re “Petitioner’s Ground 1 Challenge Against Dependent Claims 8, 18, and 24 Also Fails”	10
1. Response to Mr. Occhiogrosso re “Coffman’s Non- Proprietary Speech Engine and Application Program Interfaces (APIs) Receive Speech Inputs from Con conversationally-Aware Applications But Do Not Select or Invoke Conversationally-Aware Applications”	11
2. Response to Mr. Occhiogrosso re “Ground 1 Fails to Satisfy the Identification of Domain Agents Configured to Process a Command or Request of Dependent Claims 8, 18, and 24”	11
3. Response to Mr. Occhiogrosso re “The Board’s Understanding of Coffman’s Conversationally Aware Applications Was Correct”	25
V. CONCLUSION	26

I, Christopher Schmandt, declare as follows:

1. I have been asked to review and respond to certain points raised in a “Declaration of Benedict Occhiogrosso in Support of Patent Owner’s Response” dated February 21, 2025, filed in IPR2024-00752 (EX2013, “Occhiogrosso Declaration”). This Declaration will focus on portions of the Occhiogrosso Declaration that affirmatively present new facts or new rationale to which a response is warranted. To the extent other portions of the Occhiogrosso Declaration state general disagreements with conclusions that I reached (on which the Petition relied), without actually providing contrary evidence or analysis, I adhere to the analysis in my opening declaration (EX1002, “Opening Declaration”). In addition to the Occhiogrosso Declaration, I have also reviewed the Board’s Decision Granting Institution of *Inter Partes* Review (Paper 11, “DI”), my deposition transcript (EX2011), and the deposition transcript of Mr. Occhiogrosso (EX1018), and Board’s Institution Decision in IPR2024-00746 (EX2012).

I. PERSON OF ORDINARY SKILL IN THE ART DEFINITION

2. There appear to be no disagreements between the parties with respect to the qualifications of a person of ordinary skill in the art, as Mr. Occhiogrosso adopted the formulation in my Opening Declaration. (EX2013, ¶29.)

II. RESPONSE TO MR. OCCHIOGROSSO RE “TECHNICAL INTRODUCTION TO DIALECT’S ’652 PATENT”

3. At paragraphs 32-46 of the Occhiogrosso Declaration, Mr.

Occhiogrosso purports to provide a “technical introduction to Dialect’s ’652 patent.” (EX2013, ¶¶32-46.) To the extent any of Mr. Occhiogrosso’s assertions warrant a response, I address them in my analysis below.

III. CLAIM CONSTRUCTION

4. Paragraphs 47-50 of the Occhiogrosso Declaration are a section entitled “Claim Construction.” (EX2013, ¶¶47-50.) At paragraph 50 of the Occhiogrosso Declaration, Mr. Occhiogrosso notes district court constructions for terms in patents related to the ’652 patent, but he states that “[m]y opinions stated below respond to the Petition without adopting the District Court’s constructions.” (EX2013, ¶50.) Mr. Occhiogrosso does not argue that district court constructions should apply here or that the prior art fails to disclose or render obvious the challenged claims because of any of those constructions.

IV. RESPONSE TO MR. OCCHIOGROSSO RE “PETITIONER’S MERITS CHALLENGES”

A. Response to Mr. Occhiogrosso re “Overview”

5. At paragraphs 51-54 of the Occhiogrosso Declaration, Mr. Occhiogrosso provides a brief section entitled “Overview.” (EX2013, ¶¶51-54.) This section at most provides a cursory preview of Mr. Occhiogrosso’s later arguments, and as such they are addressed in my analysis below.

B. Response to Mr. Occhiogrosso re “Coffman”

6. At paragraphs 55-66 of the Occhiogrosso Declaration, Mr.

Occhiogrosso purports to provide an overview of Coffman. (EX2013, ¶¶55-66.)

Mr. Occhiogrosso makes a number of assertions that are wrong or irrelevant. For example, Mr. Occhiogrosso contends that Coffman’s context stacks are application-specific and as such a search of a context stack “does not cause Coffman’s CVM to then select a conversationally aware application to process a user’s command or request.” (EX2013, ¶¶59-60.) Mr. Occhiogrosso also asserts that Coffman does not disclose “synchronization” because Coffman allows for permissive, not mandatory, context stack updates. (EX2013, ¶66.) Mr. Occhiogrosso’s assertions are repeated in later sections of his declaration, and therefore are likewise addressed below.

C. Response to Mr. Occhiogrosso re “Petitioner’s Ground 1 Fails to Show the Synchronization of Context Stacks or Context Sets of Independent Claims 1, 14, and 20”

7. At paragraphs 67-77 of the Occhiogrosso Declaration, Mr. Occhiogrosso argues that “Petitioner’s Ground 1 fails to show the synchronization of context stacks or context sets of independent claims 1, 14, and 20.” (EX2012, ¶¶67-77.)

8. At paragraph 70, Mr. Occhiogrosso concedes that “the actions cited in Coffman indicate that the master dialog manager may instruct a slave dialog manager *to add or update* an entry to its context stack” but he argues that “Coffman does not disclose or suggest that the stacks themselves would be synchronized as required by the claims of the ’652 patent.” (EX2013, ¶70 (emphasis in original); *see*

also id., ¶¶68-69 (discussing Petition at 30-31, citing Coffman 25:10-14, 27-:7-20).)

9. The reason Mr. Occhiogrosso gives is the same one that the Board correctly rejected in the Institution Decision: according to Mr. Occhiogrosso, the claimed “**synchroniz[ing]**” requires perfect synchronization, not allowing any deviation between the respective context stacks of the synchronized devices. (EX2013, ¶¶71 (“One attribute of two synchronized context stacks is that the content of each stack is the same.”), 72 (“Another attribute of two synchronized context stacks is that the order of the context entries in the stack is the same.”).) “For two stacks to be synchronized,” according to Mr. Occhiogrosso, “it is not enough that they contain the same entries.... they would contain stack entries in an identical order as well[.]” (EX2013, ¶72 (underlining added).) But as the Board recognized, the ’652 patent itself contradicts any requirement for perfect synchronization, and it confirms that “**synchroniz[ing]**” context stacks as claimed is satisfied by adding or updating contexts stored on a context stack. (DI, pp.26-27.)

10. I disagree that the claims require perfect synchronization. Starting with the claim language, the claim itself describes what “**synchroniz[ing]**” entails. The claim states that context stacks are “**synchronized**” “**such that the context information of the first context stack is updated based on related context information of the second context stack.**” (’652, 38:50-54 (claim 1) (underlining added); *see also id.*, 40:47-51 (claim 14, same), 41:58-61 (claim 20, same with

“context set”).) The claim does not state, for example, that synchronizing causes the first context stack’s content and ordering to be identical with the second context stack. Instead, according to the claim itself, “**updating**” the first context stack’s context information “**based on related context information**” of the second context stack is enough.

11. As the Institution Decision correctly pointed out, this understanding is also consistent with the written description of the ’652 patent, which describes “[t]he synchronization process” as “attempt[ing] to determine which version of a data element or an agent are the newest or most up-to-date and propagates that element,” including replacing “a portion of the database.” (DI, p.25 (quoting ’652, 17:25–32).) Similarly, the ’652 patent describes how mobile devices “may be informed of context changes through context tracking module **714**, thereby synchronizing the context across the registered mobile devices **36**.” (’652, 19:36-44.) Neither of these passages describes or implies that “synchronizing” necessarily results in context stacks with identical content and ordering. Nor does any other passage in the ’652 patent. The Board correctly rejected Patent Owner’s attempt to read a requirement of “perfect” (or “complete”) synchronization into the claims. But even if the claims did require “perfect” synchronization, a person of ordinary skill in the art would have understood and found it obvious that Coffman’s synchronization could at least sometimes be “perfect.” For example, in an embodiment involving a master device

and just one slave device, a person of ordinary skill in the art would have understood that both the content and ordering of the master device's and slave device's respective context stacks could be identical as a result of context stack updates. It would make sense for the context stacks in this situation to be identical, since Coffman teaches that the master device is "responsible for managing the flow of I/O to the different managers to decide the active dialog and appropriately execute a query and update the context and history." (Coffman, 27:5-7.) By having identical content and ordering, the master dialog manager would be better able to carry out its responsibilities, including by helping ensure that its decision regarding the active dialog would be consistent with the slave device's decision regarding the active dialog. In fact, as Coffman notes, "[t]he role of master and slave can be dynamically switched." (Coffman, 26:1-3.)

12. Mr. Occhiogrosso also repeats the rejected argument that Coffman does not disclose the claimed synchronization because each device in Coffman's system can choose to act or not to act in response to exchanging information between devices. (EX2013, ¶¶66, 75 ("the protocols for information exchange between different dialog managers propose, rather than require, context changes"); DI, pp.26-27.) As the Institution Decision again correctly found, "claim 1 does not specify any particular conditions or points in time when the synchronization must occur." (DI, p.27 (citing '652, 38:50-54).) Mr. Occhiogrosso agreed at his deposition. (EX1018

(Occhiogrosso Dep. Tr.), 25:20-22 (claim 1 does not specify “any particular event or trigger” that causes synchronization).) And the ’652 patent specification “contemplates that [t]he synchronization may be performed automatically when any two or more of the computers are connected to these networks,’ or ‘[a]lternatively...the synchronization may be performed on demand under user control.” (DI, pp.26-27 (quoting ’652, 17:15-25).) “Both of those scenarios contemplate points in time when the devices are not synchronized — in the first scenario when the computer is not connected to the network, and in the second scenario before the user initiates the synchronization.” (DI, p.27.) Even if devices in Coffman’s system only sometimes chose to act in response to exchanging information between devices, that is enough to disclose the claimed “synchronizing.” This is because, I understand that under the patent laws, just as an accused product that sometimes, but not always, embodies a claimed method nonetheless infringes, a prior art reference that sometimes, but not always, embodies a claimed method nonetheless teaches that aspect of the invention.

13. Mr. Occhiogrosso lastly remarks that “I could not locate anything in Mr. Schmandt’s declaration or testimony, or the Petition, concerning the desirability of synchronized context stacks in Coffman.” (EX2013, ¶76.) His comment appears to be related to his unsupported requirement for perfect synchronization and is therefore irrelevant.

14. Accordingly, as the Institution Decision found, Coffman’s disclosures of adding and updating a device’s context stack with context information from another device’s context stack discloses the claimed “**synchroniz[ing]**” step. (DI, pp.20-25; *see also* EX1002, ¶¶70-71; Coffman, e.g., 25:10-14 (“[C]ontext information may be transmitted by indicating passing or pointing to the context stack/history of the device or application that the controller can access and add to its context stack.”) (underlining added) (underlining added).)

D. Response to Mr. Occhiogrosso re “Petitioner’s Ground 1 Challenge Against Dependent Claims 8, 18, and 24 Also Fails”

15. At paragraphs 78-124 of the Occhiogrosso Declaration, Mr. Occhiogrosso makes a number of arguments as to why, in his view, “Petitioner’s Ground 1 challenge against dependent claims 8, 18, and 24 also fails.” (EX2013, ¶¶78-124.) As explained below, I disagree with Mr. Occhiogrosso. Mr. Occhiogrosso’s arguments are all founded on the same misreading of Coffman, that a context stack is not used to invoke a conversationally-aware application (the claimed “**domain agent**”) to process a command/request.

16. For reference, Claim 8 depends from claim 1 via claim 3 and recites in relevant part:

[T]he one or more physical processors are further caused to:
identify one or more domain agents associated with the one or more
context entries, wherein the one or more domain agents are

configured to process the command or request; and
generate a response to the command or request using the one or more
domain agents.

(’652, 39:47-54.) Claims 18 and 24 depend from different base claims but include substantially similar features as claim 8. (’652, 41:24-31 (claim 18), 42:33-39 (claim 24).)

1. Response to Mr. Occhiogrosso re “Coffman’s Non-Proprietary Speech Engine and Application Program Interfaces (APIs) Receive Speech Inputs from Conversationally-Aware Applications But Do Not Select or Invoke Conversationally-Aware Applications”

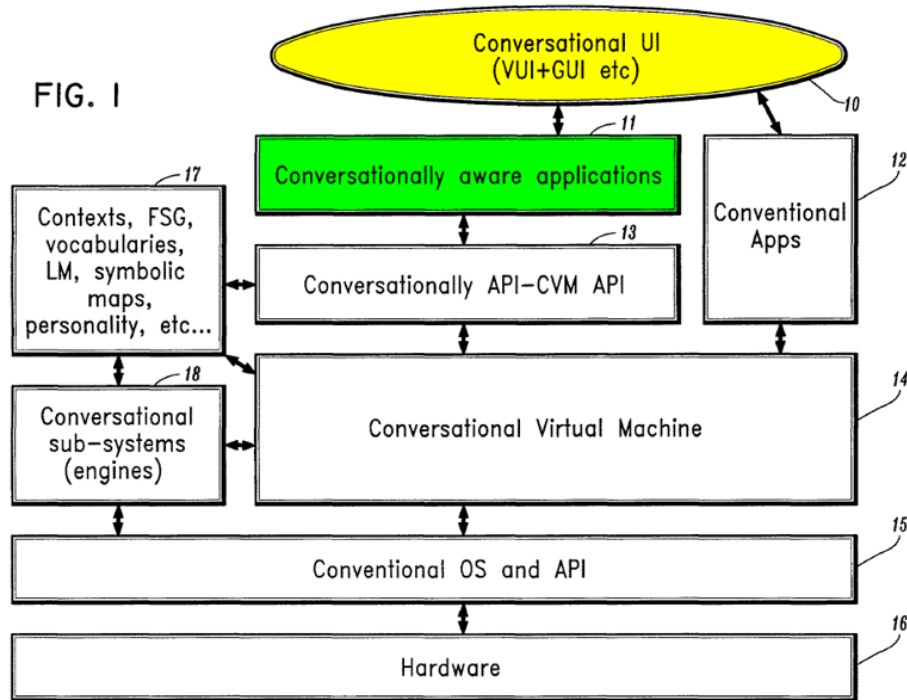
17. At paragraphs 84-98 of the Occhiogrosso Declaration, Mr. Occhiogrosso makes several arguments as to why Coffman’s conversationally-aware application does not disclose the claimed “**domain agent**” of claim 8. Because Mr. Occhiogrosso’s arguments are repeated “in more detail” in later sections of his declaration, I address them below. (EX2013, ¶¶85 (“as I explain...”), 86 (“as I discuss in more detail below...”), 92 (“As I note below...”).)

2. Response to Mr. Occhiogrosso re “Ground 1 Fails to Satisfy the Identification of Domain Agents Configured to Process a Command or Request of Dependent Claims 8, 18, and 24”

18. At paragraphs 99-119 of the Occhiogrosso Declaration, Mr. Occhiogrosso argues that “Ground 1 fails to satisfy the identification of domain agents configured to process a command or request of dependent claims 8, 18, and 24.” (EX2013, ¶¶99-119.)

19. Mr. Occhiogrosso first argues that Coffman does not disclose “**identifying**” a domain agent because, according to him, Coffman’s conversationally-aware applications merely provide a front-end user interface to the system, and in order for a user to interact with an application, the user must select it—much like how users often manually toggle among applications in a Microsoft Windows environment. (EX2013, ¶¶99-101.) I disagree. For starters, this would make no sense. If a conversationally-aware application is just a “gateway[] to use the system” as Mr. Occhiogrosso argues (EX2013, ¶99), why would there be more than one conversationally-aware application for a user to interact with? Other conversationally-aware applications would simply be redundant, and there would be no need to toggle among them, because all of the actual processing would be performed by other parts of the system. Mr. Occhiogrosso is confusing Coffman’s separate “conversational user interface”—which can serve as a gateway to use the system—with the “conversationally aware applications”:

FIG. 1



(Coffman, Fig. 1; *see also id.*, 64:3 (“a multi-modal conversational user interface (CUI) for receiving input queries”).)

20. Moreover, Coffman touts its system as an improvement over those prior art systems that were limited to “allow[ing] a user to change the focus, launch new tasks, and give commands to the task in focus.” (Coffman, 3:14-16.) Coffman’s system instead “allows a user to seamlessly control multiple Windows applications, for example, running in parallel[.]” (Coffman, 4:13-16 (underlining added).) Coffman’s “seamless control” is accomplished using a CVM’s dialog manager, to which “all the input and output events across all the modalities are caught and transferred” and “store[d]... appropriately in the context stack.” (Coffman, 37:18-20; EX1002, ¶61.) This enables the dialog manager to “manag[e] dialog across the

applications and... select[] an... application based on input queries.” (Coffman, 65:12-13 (claim 11) (underlining added).) A dialog manager “determines what information the user has, which inputs the user presents, and which application(s) should handle the user inputs.” (Coffman, 20:19-23; *see also id.*, 31:24-29 (dialog controller manages dialog “across all the conversational and conventional applications” and “manages and determines the active context and application”).) Therefore, “[f]or instance, a user may be able to access remotely information about an element of a spreadsheet and modify it if necessary, while simultaneous[ly] checking his e-mail.” (Coffman, 56:11-14 (underlining added); EX1002, ¶114.) Coffman’s dialog manager—not the user—identifies (and selects) the conversationally-aware application.

21. Mr. Occhiogrosso builds on his first mistaken argument by next asserting that Coffman’s context stack search does not disclose “**identifying**” a conversationally-aware application but instead identifies a process performed at the CVM. (EX2013, ¶¶104 (“Coffman associates a ‘process’ with tasks and threads performed *at the CVM.*”) (emphasis in original), 102-105.) But none of the passages of Coffman that Mr. Occhiogrosso cites as purported “confirmation” do anything other than describe context stack structure and that there are associated tasks, threads and processes. For example, Mr. Occhiogrosso cites Coffman’s statement that “[t]his stack structure allows to maintain non-ambiguous conversational interactions

with multiple tasks, threads or processes.” (EX2013, ¶104 (quoting Coffman, 42:16-17).) Nothing about this statement even suggests, let alone expressly discloses, that the “tasks, threads or processes” are performed at the CVM.

22. On the contrary, Mr. Occhiogrosso’s citation to Coffman’s statement that task dispatcher/controller 402 “dispatches and coordinates different tasks and processes that are spawned (by the user and machine)” (EX2013, ¶105 (quoting Coffman, 30:26-28) (underlining added)) actually confirms that the tasks, threads and processes are associated with applications launched by the user—such as conversationally-aware applications. This is consistent with Coffman’s system being able to “allow[] a user to seamlessly control multiple Windows applications, for example, running in parallel[.]” (Coffman, 4:13-16 (underlining added).) It is also consistent with Coffman’s statements equating “applications” with “tasks, processes and discourses.” (*Compare* Coffman, 6:6-8 (“natural dialog...across multiple tasks, processes and discourses”) *with* Coffman, 36:29-31 (“natural dialog...across multiple applications”).) It is also consistent with Coffman’s statement that “[t]he context stack is updated...on the application output,” which indicates that an “application”—and not just “tasks,” “threads” and “processes”—is associated with context stack entries, and involved in processing associated with context stack entries. (Coffman, 42:27-29 (underlining added).)

23. But even if the tasks, processes and threads Coffman describes could

be understood as encompassing processes performed at the CVM, it is enough that the tasks, processes and threads would at least encompass conversationally aware applications, such that a conversationally aware application would at least sometimes be identified to process a query. This is because, as I understand, under the patent laws, combinations of prior art that just sometimes, and not necessarily always, meet the claim elements are sufficient to show obviousness.

24. As further alleged “confirm[ation] that Coffman processes user request and commands at the CVM,” and not with a conversationally-aware application, Mr. Occhiogrosso points to statements in Coffman that “all conversational subsystem calls are hidden to the application” and that the CVM drives execution of the conversational subsystem and returns “the results.” (EX2013, ¶¶106 (quoting Coffman, 29:7-11), 107.) But these conversational subsystems relate to services such as speech recognition and text-to-speech synthesis. (Coffman, 29:1-6.) The “results,” therefore, are simply basic system input and output processing—for example, returning text recognized from speech input, or converting a text-based output provided by an application into speech audio. (Coffman, 41:2-5 (“speech inputs being transcribed/understood”); EX1002, ¶78.) In fact, rather than being the results of performing a command/request, these “results”—such as recognized text from speech input—are used to search the context stack and determine the command/request that should be performed in the first place: “The output of these

subsystems is run down the context stack 405 to extract the active query and complete it.” (Coffman, 41:4-5 (underlining added); EX1002, ¶78.) Then, when the application returns output as a result of performing the command/request, the context stack is updated. (Coffman, 42:27-29 (“The context stack is updated pending on voice, keyboard, mouse or any other input or command and on the application output.”) (underlining added).)

25. Mr. Occhiogrosso also points to Coffman’s statements that “[t]ypically, different devices, CVMs or different applications will have their own dialog manager, context stack 217, 217a and global history 218, 218a” and that a “context stack 217 comprises all the information associated with an application.” (EX2013, ¶¶108 (citing Coffman, 27:1-2, 21:23-24), 109.) According to Mr. Occhiogrosso, these statements mean that Coffman’s context stacks must each be application-specific. (EX2013, ¶108 (“a context stack for one application”).) As explained below, Mr. Occhiogrosso is again mistaken. But even if he were correct, Coffman explains that user input may not necessarily match any entry in a context stack, and if not, other context stacks are searched to find a match:

When a user interacts with the collection of devices, the interaction may always be via a central unit such as a PVA (personal vehicle assistant) 710 in a car, or a speech browser 713. The task dispatcher and context stack accumulates the contexts associated to all the devices and will parse and dispatch commands to each device accordingly. If the user

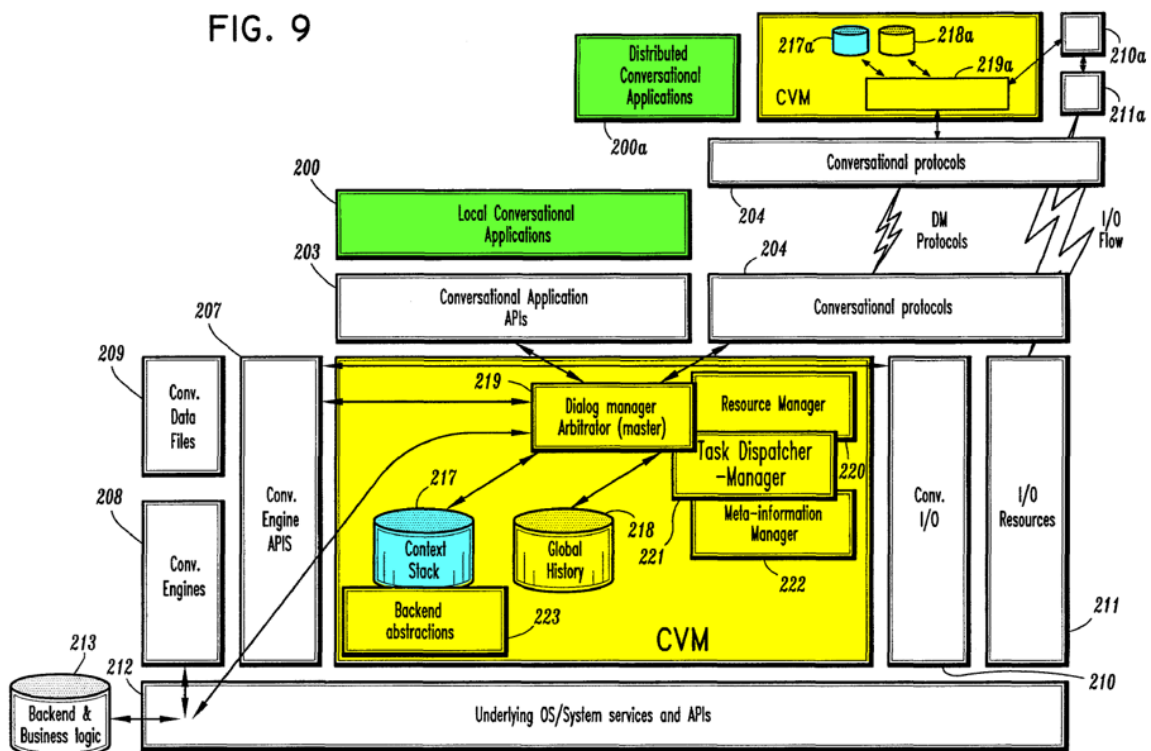
interacts with the entire collection of devices, then a device is always active (the last activated context). This devices check [sic] if a new command fits its context stack. If not, it passes to a neighboring device that becomes active. The process is iterated until a match is found, and possibly confirmed by the user, or the request bounces back to the first device. In that case, an error or confusion message is returned to the user.

(Coffman, 55:27-56:3 (underlining added); EX1002, ¶91.) Therefore, even if a context stack is limited to a single application—and even if it could be said that the user selected the application—because Coffman’s system may not necessarily find a matching context entry in that context stack, when a matching entry is actually found in a context stack, Coffman’s system has “**identified**” the associated application for processing the command/request. (*See also* Coffman, 37:2-4 (“The first task to have its dialog completed by producing a non-ambiguous query is executed and the corresponding query as interpreted by the other application is stored to activate if the recognized query is rejected by the user.”) (underlining added); EX1002, ¶98.)

26. Moreover, the passages from Coffman that Mr. Occhiogrosso cites simply do not require that every context stack is limited to a single application. Instead, Coffman states that “[t]ypically, different devices, CVMs or different applications will have their own...context stack 217, 217a.” (Coffman, 27:1-2.)

Coffman does not say “different devices, CVMs and different applications....”

Coffman’s statement is entirely consistent with two devices that each have their own context stack that includes context entries for multiple applications. This understanding is also consistent with the figure associated with Coffman’s statement, which depicts just one **context stack** associated with each device and the **CVM** and “**conversational applications**” on that device:



(Coffman, Fig. 9 (highlighting added).)

27. What Coffman’s statement is actually referring to is an optional, alternative “stand-alone” type of conversationally-aware application, such as a “conversational browser,” that “can be [a] stand-alone application[] carrying its own CVM.” (Coffman, 59:12-13.) When Mr. Occhiogrosso was asked about his

understanding of Coffman's "stand-alone" applications at his deposition, he stated that he did not recall them. (EX1018, 32:11-23.) Respectfully, his understanding of Coffman is incomplete. Coffman's statement that "different devices, CVMs or different applications will have their own...context stack 217, 217a" is a restatement of Coffman's description of alternative ways to implement its CVM system:

Referring now to Fig. 4, a diagram illustrates a detailed architecture of a conversational system and the core functional modules of the conversational kernel of the CVM system according to one embodiment of the present invention. It is to be understood that the system of Fig. 4 and the accompanying description are for purposes of illustration to provide implementation examples and that one of ordinary skill in the art can envision other components or system architectures for implementing a CVM according to the spirit of the present invention. Furthermore, it is to be appreciated that each of these elements can be introduced in stand-alone mode within an application or as platform under an existing operating system, or a true CVM with a core kernel built around these different new elements.

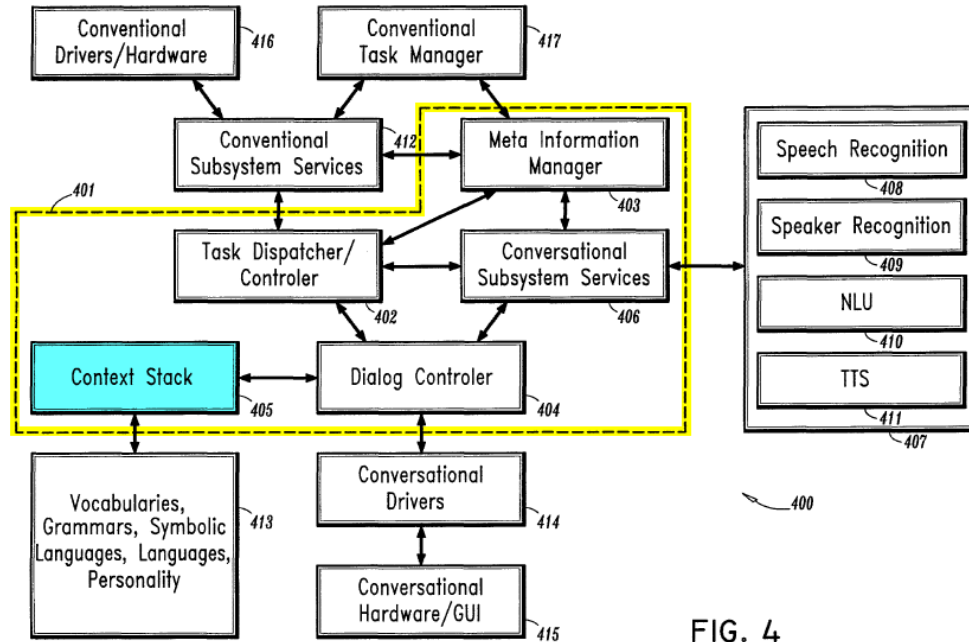


FIG. 4

(Coffman, 28:10-18, Fig. 4 (with elements **context stack 405** and **CVM 401**) (highlighting added).) Coffman does not require that every application have its own, dedicated context stack. In fact, with a more complete understanding of Coffman, its statement that “different devices, CVMs or different applications will have their own...context stack 217, 217a” suggests that—in a device-based CVM implementation—multiple applications actually do share a single context stack. This is consistent with Coffman’s other statements that “dialog controller 404 will control the dialog of **all** applications and manage the context through the context stack 405” and describing how “[a] context stack accumulates the contexts associated to all the devices and will parse and dispatch commands to each device accordingly.” (Coffman, 32:27-28, 55:27-30 (underlining added to both); EX1002, ¶91; *see also id.*, Fig. 4 (showing single context stack 405).) Mr. Occhiogrosso’s

unduly narrow interpretation of Coffman requiring per-application context stacks cannot be reconciled with Coffman's other disclosures.

28. Mr. Occhiogrosso next asserts that Coffman also fails to disclose **“generate a response to the command or request using the one or more domain agents,”** as claimed, by largely repeating the arguments already addressed above. (EX2013, ¶¶110-111.) Mr. Occhiogrosso states, for example, that “[n]owhere in the discussion of Coffman do I find a selection of an application, such as a calendar application, based on a determined meaning of a user’s utterance.” (EX2013, ¶111.) But as discussed above, Coffman’s system “allows a user to seamlessly control multiple Windows applications, for example, running in parallel[.]” (Coffman, 4:13-16 (underlining added).) A user is able to control multiple Windows applications because Coffman’s dialog manager “manag[es] dialog across the applications and... select[s] an... application based on input queries.” (Coffman, 65:12-13 (claim 11) (underlining added).) Therefore, “[f]or instance, a user may be able to access remotely information about an element of a spreadsheet and modify it if necessary, while simultaneous[ly] checking his e-mail.” (Coffman, 56:11-14; *see also* EX1002, ¶114.) Coffman therefore teaches that if a user asks for the value of a particular spreadsheet cell, the CVM identifies the conversationally aware spreadsheet application, and the application would lookup and provide that information. In fact, Mr. Occhiogrosso conceded at his deposition that the spreadsheet application itself

would access that information and be used in generating a response. (EX1018, 36:13-24 (“[T]he spreadsheet application would access the spreadsheet and the cell and again, using the API and the CVM and its resources, would create the verbal response.”); *see generally id.*, 35:19-36:24.) Coffman nowhere discloses such application-specific functionality is moved to or duplicated by the CVM or its subsystems. For example, if the spreadsheet application were only a “conventional” application, there should be no dispute that the application itself would perform these sorts of operations. Coffman never suggests that, in order to make the spreadsheet application “conversationally aware,” the application’s core functionality would be moved to the CVM or conversational subsystems.¹ In fact, Coffman explains that “[t]he context stack is updated...on the application output.” (Coffman, 42:27-29 (underlining added).)

29. Mr. Occhiogrosso states that he does not believe that cites to Coffman’s disclosures at 13:8-13 and 58:9-11 “can support Petitioner’s conclusions because

¹ Coffman in fact describes “conversational foundation classes,” which are components that an application developer can build or link into their application to help make the application “conversationally aware”—*not* to move (or duplicate) actual application-specific functionality to the CVM. (Coffman, 13:29-32, 14:8-12.)

Coffman’s disclosures at 13:8-13 and 58:9-11 describe actions at the CVM 14, not a conversationally-aware application.” (EX2013, ¶112.) Mr. Occhiogrosso is again mistaken. Coffman at 13:8-13 describes how subsystems convert output/results from an application into speech, for example, as explained above. (Coffman, 13:8-13 (“[C]onversational sub-systems 18 are responsible for converting voice requests into queries and converting outputs and results into spoken 10 messages using the appropriate data files 17 (e.g., contexts, finite state grammars, vocabularies, language models, symbolic query maps etc.)”); *see also id.*, 42:27-29 (“The context stack is updated pending on voice, keyboard, mouse or any other input or command and on the application output.”).) And Coffman at 58:9-11 likewise describes how the CVM converts user input into “application calls”—i.e., an application is called (invoked) to process a command/request based on the user input—and converts the output from the application into speech. (Coffman, 58:9-11 (“[T]he API conveys all the information for the CVM to transform queries into application calls and conversely converts output into speech, appropriately sorted before being provided to the user”).) As I discussed above, Mr. Occhiogrosso conceded at his deposition that a conversationally-aware application is invoked to respond to a user command/request. (EX1018, 36:13-24; *see generally id.*, 35:19-36:24.)

30. Mr. Occhiogrosso’s remaining arguments in this section are repackaged

versions of those already addressed above. (EX2013, ¶¶113-119.)

3. Response to Mr. Occhiogrosso re “The Board’s Understanding of Coffman’s Conversationally Aware Applications Was Correct”

31. At paragraphs 120-124 of the Occhiogrosso Declaration, Mr. Occhiogrosso offers his opinion that “the Board’s understanding of Coffman’s conversationally aware applications was correct” in another IPR (IPR2024-00746) that involved a different patent.

32. To start, I note that the language in the challenged claim in IPR2024-00746 is different than the ’652 patent claims. As Mr. Occhiogrosso notes, the challenged claim in IPR2024-00746 recited “selecting at least one domain agent based on the determined meaning, the selected domain agent being an autonomous executable that receives, processes, and responds to requests associated with the determined context.” (EX2013, ¶121.) In contrast, here, dependent claims 8, 18, and 24 recite “identify[ing] one or more domain agents associated with the one or more context entries, wherein the one or more domain agents are configured to process the command or request.”

33. Moreover, to the extent the Board’s findings in another IPR are relevant or informative, the Board’s institution decision in that other IPR was made on an incomplete, preliminary record. Here, with a more complete record, and as shown above and in my Opening Declaration, Coffman’s teachings regarding a CVM

system with conversationally-aware applications discloses and renders obvious the limitations of dependent claims 8, 18, and 24.

V. CONCLUSION

34. In signing this Declaration, I recognize that the Declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in this proceeding. If required, I will appear for cross-examination at the appropriate time. I reserve the right to offer opinions relevant to the invalidity of the challenged claims at issue and/or offer testimony in support of this Declaration.

35. I hereby declare that all statements made herein of my own knowledge are true and that all statements are made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001.

Dated: May 15, 2025

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'Chris Schmandt', written over a horizontal line.

Christopher Schmandt
Oak Bluffs, MA