

Google LLC

v.

Dialect, LLC

IPR2024-00752

Patent 8,849,652 B2

Petitioner's Demonstratives

August 5, 2025

Before George R. Hoskins, Kristen L. Droesch, and Hyun J. Jung

Administrative Patent Judges

Google Exhibit 1019
IPR2024-00752

Claim 1 of the '652 Patent

1. A system for processing natural language utterances where recognized words of the natural language utterances alone are insufficient to completely determine one or more commands or requests, the system comprising:

one or more physical processors programmed with one or more computer program instructions which, when executed, cause the one or more physical processors to:

generate a first context stack associated with a first device, the first context stack comprising context information that corresponds to a plurality of prior utterances;

synchronize the first context stack with a second context stack associated with a second device such that the context information of the first context stack is updated based on related context information of the second context stack;

receive a natural language utterance associated with a command or request;

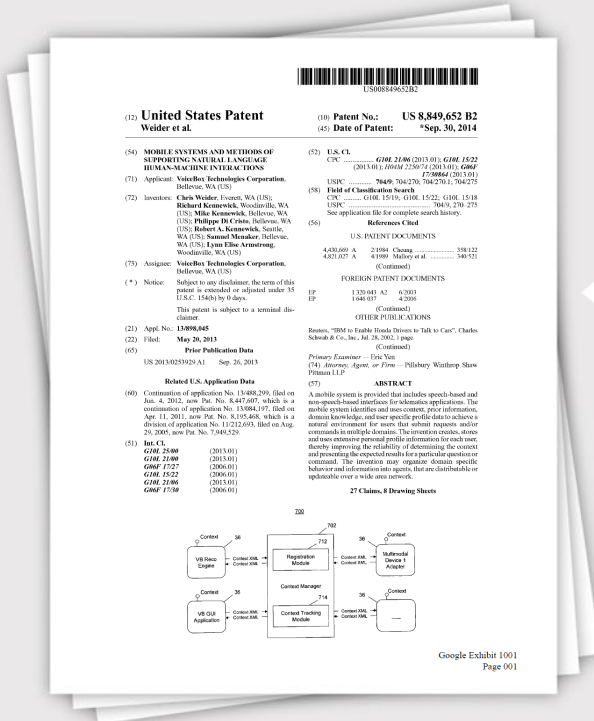
determine one or more words of the natural language utterance by performing speech recognition on the natural language utterance; and

determine the command or request based on the one or more words and the updated context information.

Independent claims 14 and 20 are substantially similar

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

2



U.S. Patent 8,849,652 B2
EX1001

Dependent Claims 3 & 8 of the '652 Patent

3. The system of claim 1, wherein the first context stack includes a plurality of context entries, and wherein the one or more physical processors are further caused to:

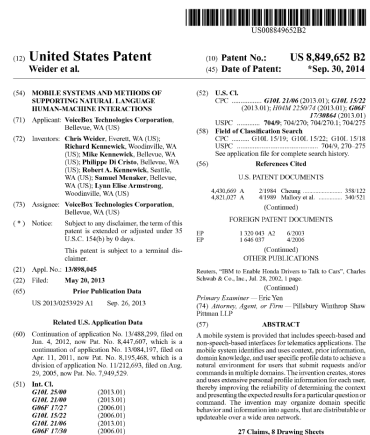
identify, from among the plurality of context entries, one or more context entries that correspond to the one or more words, wherein the updated context information includes the one or more context entries.

8. The system of claim 3, wherein the 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.

Dependent claims 16 & 18 and 22 & 24 are substantially similar



U.S. Patent 8,849,652 B2
EX1001

Instituted Grounds

Ground	Claims	Basis for Challenge under §103
1	1-9, 14-18, 20-26	Coffman (EX1003)
2	10-13, 19, 27	Coffman and Kennewick (EX1004)
3	13, 19	Coffman, Kennewick and Lee (EX1006)

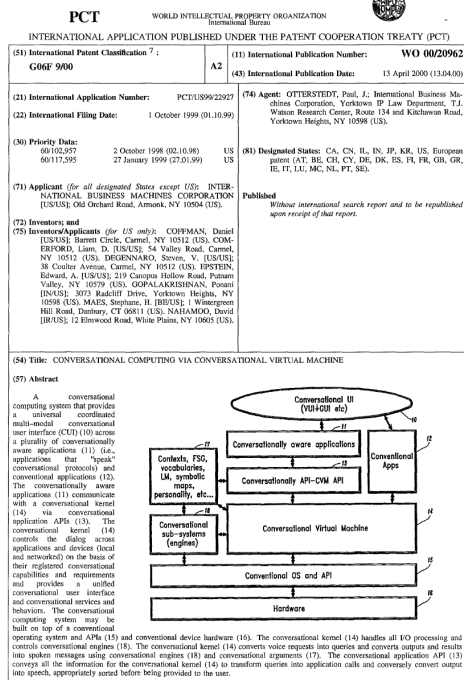
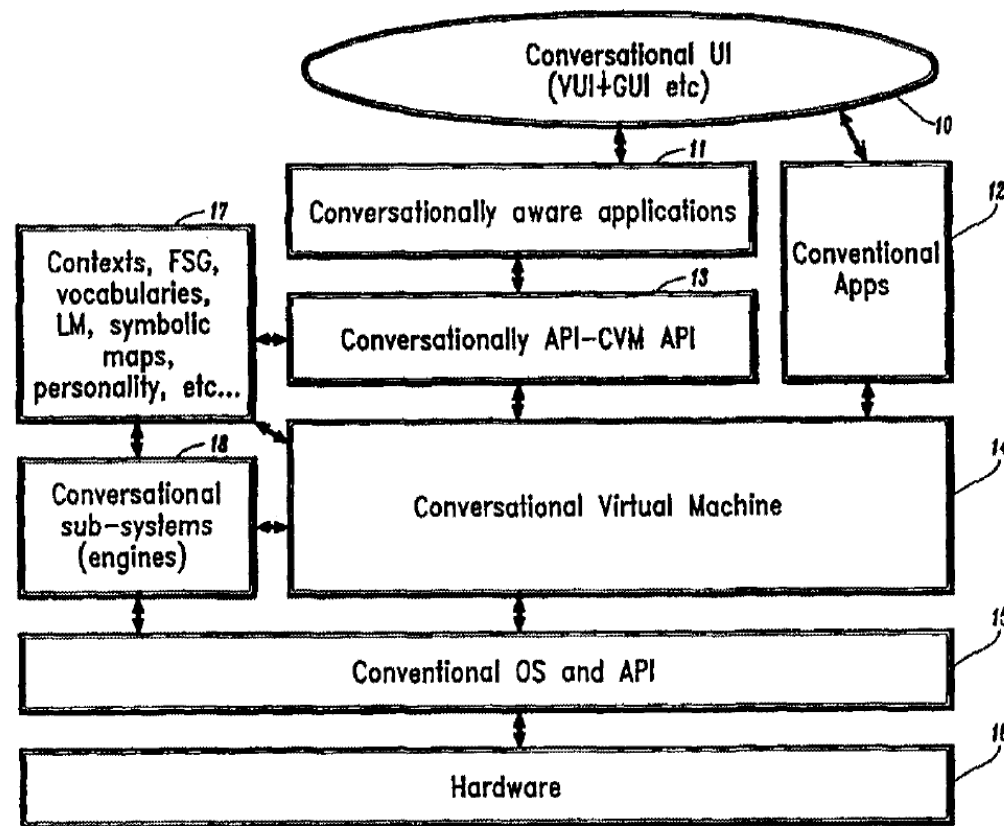
Prior Art

Coffman (EX1003)

(54) Title: CONVERSATIONAL COMPUTING VIA CONVERSATIONAL VIRTUAL MACHINE

(57) Abstract

A conversational computing system that provides a universal coordinated multi-modal conversational user interface (CUI) (10) across a plurality of conversationally aware applications (11) (i.e., applications that "speak" conversational protocols) and conventional applications (12). The conversationally aware applications (11) communicate with a conversational kernel (14) via conversational application APIs (13). The conversational kernel (14) controls the dialog across applications and devices (local and networked) on the basis of their registered conversational capabilities and requirements and provides a unified conversational user interface and conversational services and behaviors. The conversational computing system may be built on top of a conventional operating system and APIs (15) and conventional device hardware (16). The conversational kernel (14) handles all I/O processing and controls conversational engines (18). The conversational kernel (14) converts voice requests into queries and converts outputs and results into spoken messages using conversational engines (18) and conversational arguments (17). The conversational application API (13) conveys all the information for the conversational kernel (14) to transform queries into application calls and conversely convert output into speech, appropriately sorted before being provided to the user.



(Coffman (EX1003); Petition at 19)

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

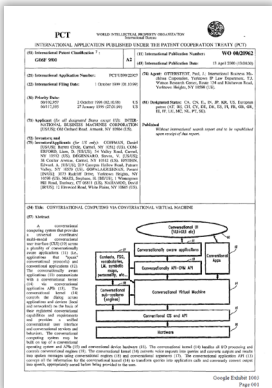
Coffman's conversational system improves on existing speech-control systems

In addition, speech interfaces for GUI based OSs have been implemented using commercially available continuous speech recognition applications for dictation and command and control. These speech applications, however, are essentially add-ons to the GUI based OSs in the sense that such applications allow for the replacement of keyboard and mouse and allows a user to change the focus, launch new tasks, and give voice commands to the task in focus.

...

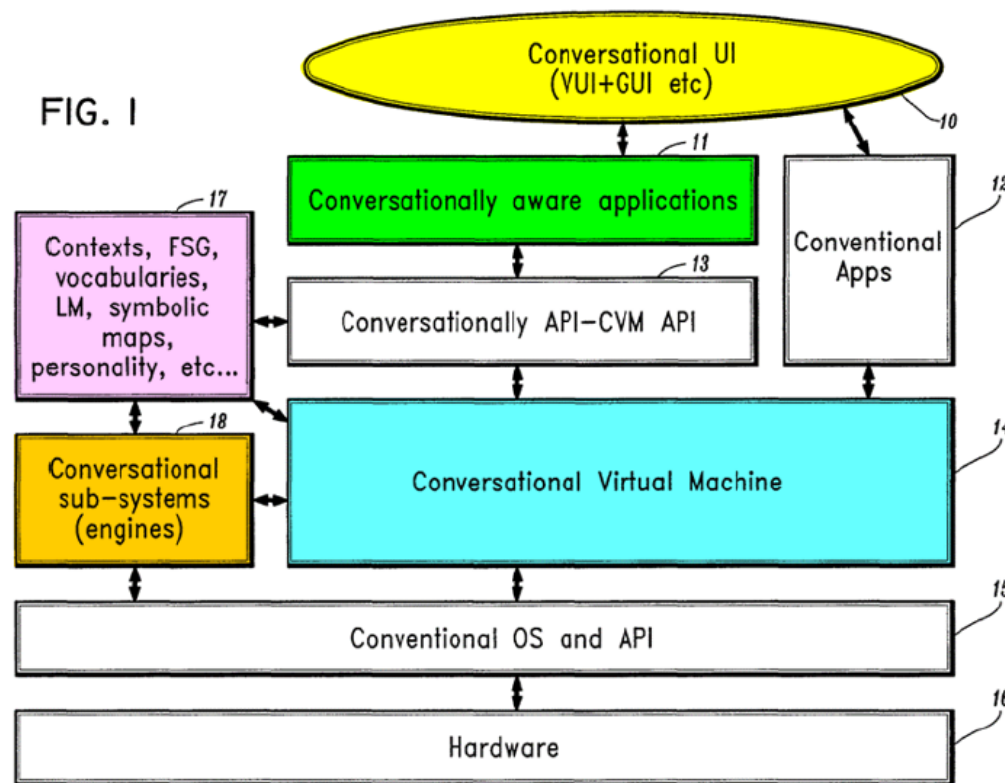
dialogs and conversations, even if such dialogs are carried over, e.g., a keyboard. It is the conversational computing paradigm that allows a user to seamlessly control multiple Windows applications, for example, running in parallel, even through a dummy terminal display such as VT 100 or a Palm Pilot screen.

(Coffman (EX1003), 3:12-16, 4:13-16)
(Reply at 12)



Coffman (EX1003)

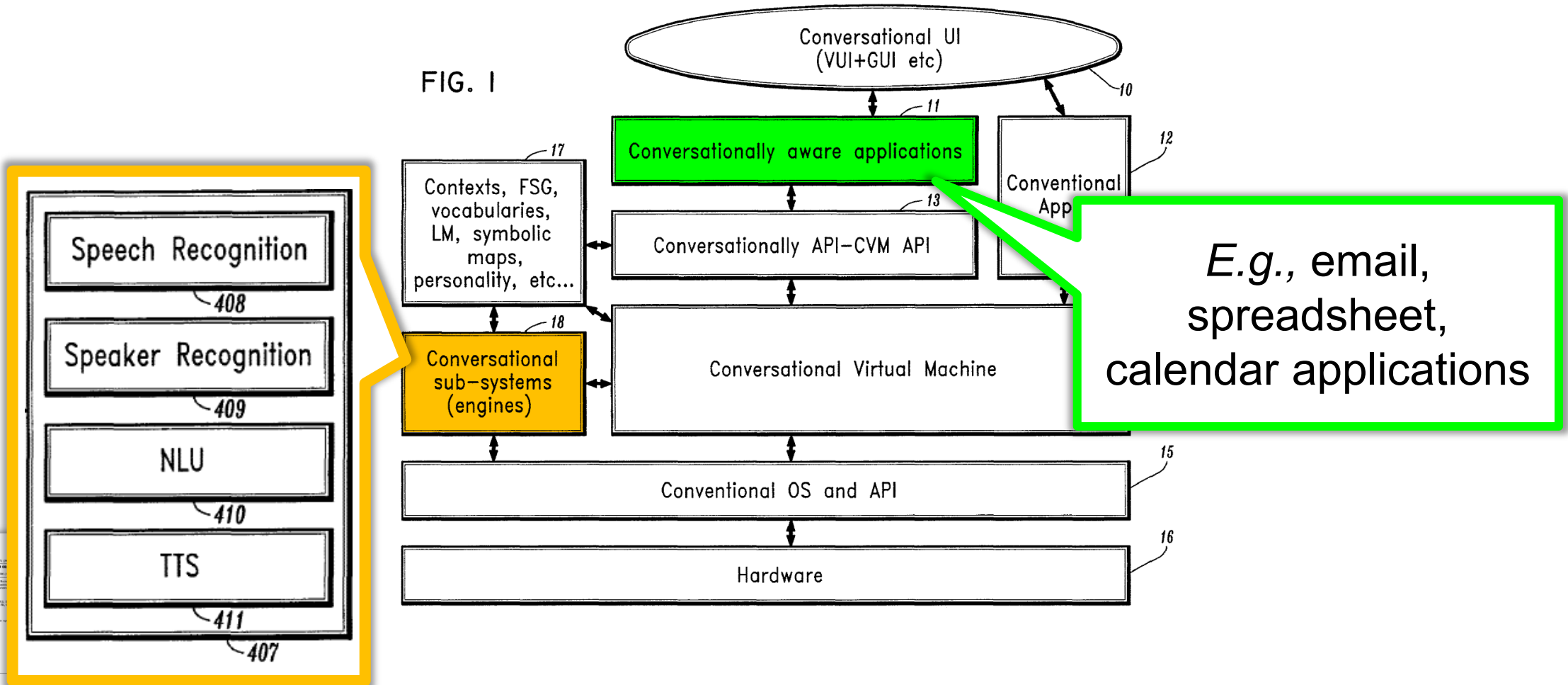
FIG. 1



(Coffman, Fig. 1 (highlighting added).) As shown above, the system includes, among other things, “a universal coordinated multi-modal conversational user interface (CUI) (10),” “a plurality of conversationally-aware applications (11),” “conversational kernel (14),” “conversational arguments (17),” and “conversational engines (18).” (Coffman, face page (Abstract).)

(Coffman (Ex1003), Fig. 1)
(EX1002, ¶42; Reply at 12)

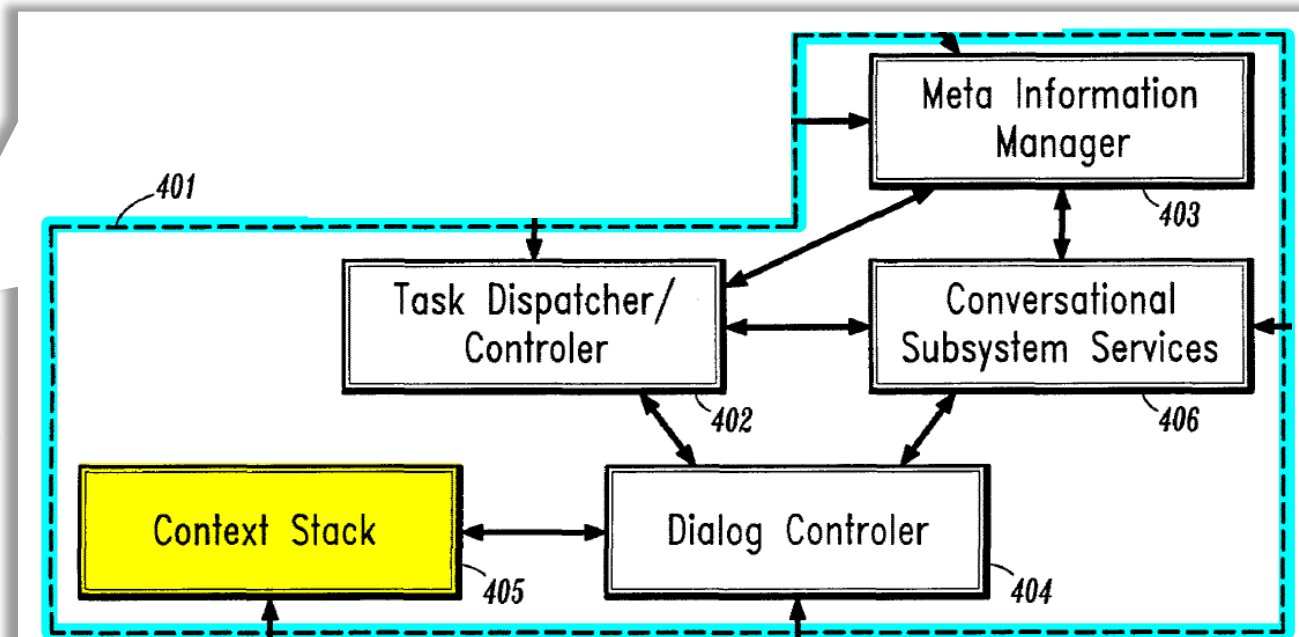
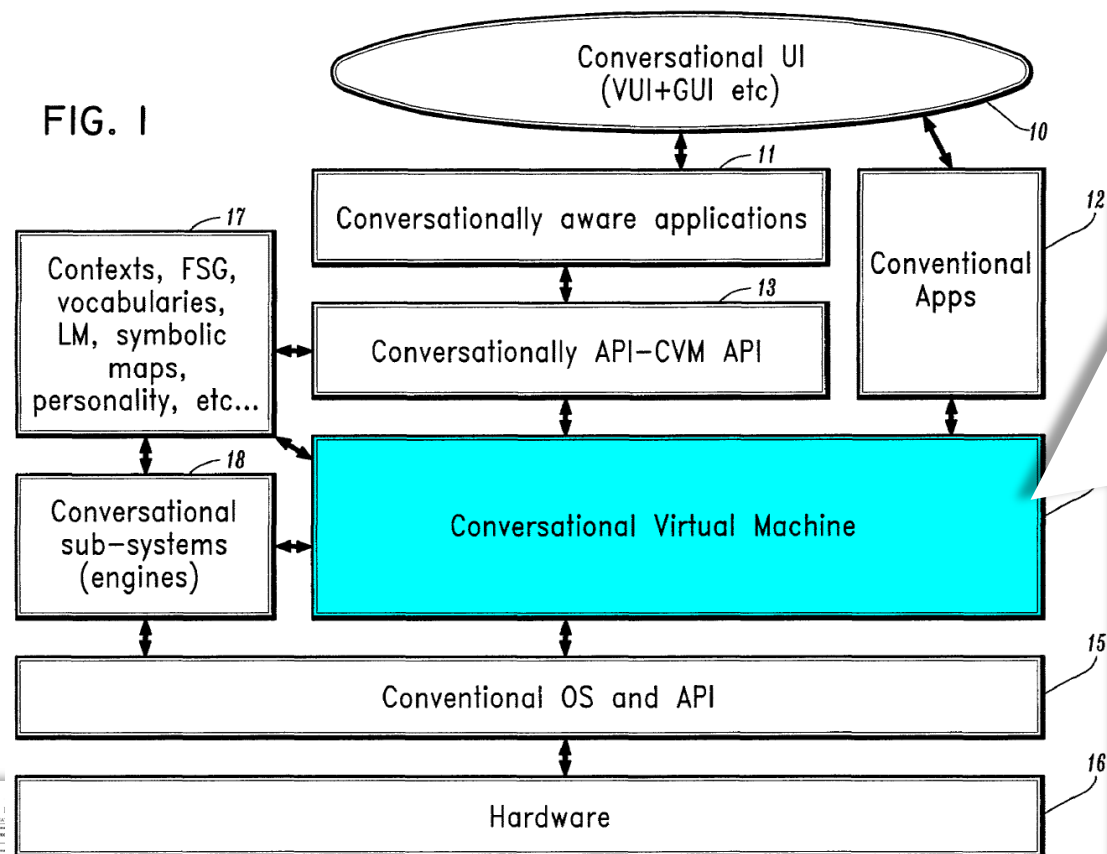
FIG. 1



(See Coffman (EX1003), Figs. 1, 4)
(Petition at 33-34, 55)

Coffman (EX1003)

FIG. 1



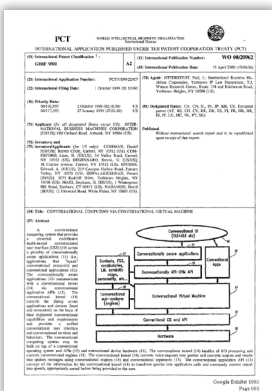
The core of the CVM 401 is the context stack 405 which operates and is managed under the control of the dialog controller 404 (it is to be understood that the context stack 405 id

(See Coffman (Ex1003), Figs. 1, 4, 29:18-20)
(Petition at 25)

Coffman context stack

The core of the CVM 401 is the context stack 405 which operates and is managed under the control of the dialog controller 404 (it is to be understood that the context stack 405 is directly related to the global history and meta information repository discussed above). In general, the context stack 405 accumulates the context (i.e., full query arguments list of attribute value n-uples, and state/mode) of each active process with an activated discourse (i.e., conversational interaction associated with a given task/process/thread) along with any data files 413 (or at least identifiers of such conversational arguments) for the different engines that may be needed for input understanding (e.g., files or arguments that the engines use for performing their respective tasks such as active FSG, topic, vocabulary, HMM (hidden markov models), voiceprints, language models or possible queries for a speech input). In other words, the term “context” refers to the state of each discourse (whether active or nonnative), which keeps track of the past history of the discourse, its current state, and the specific characteristics and full query arguments of the corresponding task (e.g, vocabulary file, language model, parsing, tags, voiceprint, TTS rules, grammar, NLU etc. of each active task/process) along with any data needed for input understanding. It is to be appreciated that, in a distributed CVM (as explained

(Coffman (EX1003), 29:18-32)
(Petition at 25-26)



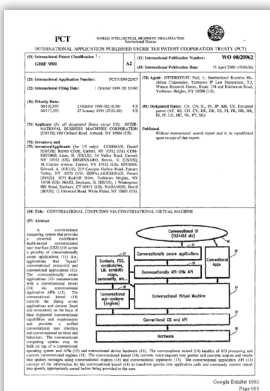
Coffman context stack

More specifically, each new task, process, or thread creates a new stack entry and is associated with a discourse. Each application may be associated with multiple discourses (e.g. the application management discourse and the application content navigation discourses). Each context associated with a given discourse comprises the latest requests made to the corresponding process/task/thread as well as the latest output. The context of a given discourse is also associated with, e.g., any active grammars, vocabularies and symbolic language which maps the actual query. Again, the latest information is stored in the history and context stacks.

...

The dialog controller 404 manages the context stack 405 by creating a new stack entry in the context stack 405 for each new task/process/thread that is spawned either local or remotely from a networked device (with task management being controlled by the task dispatcher/controller 402 as discussed below). Each active application can be associated with multiple discourses (e.g. the application management discourse and the application content navigation discourses). As explained above, each context associated with a given discourse

(Coffman (EX1003), 30:4-10, 30:14-20)
(Petition at 42)



Coffman context stack searched to identify matching process to activate

resources. Based on the modality (pointer, keyboard, file, speech), the task dispatcher 402 redirects the stream to the appropriate conversational subsystems or conventional subsystem with speech inputs being transcribed/understood. The output of these subsystems is run down the context stack 405 to extract the active query and complete it. On the other hand, outputs are

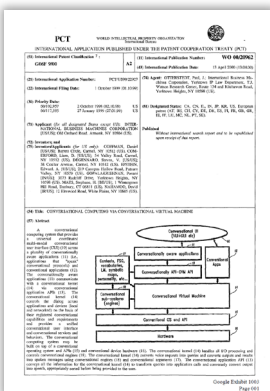
...

Completion/search for the active context is performed from context to context down the stack. That is, new queries or arguments are compared by the dialog engine by going down the stack until an acceptable match is obtained and optionally confirmation is obtained from the user. As soon a context is found that fits the utterance at the level of the NLU symbolic language, the context becomes active and the corresponding process becomes active. Until the

...

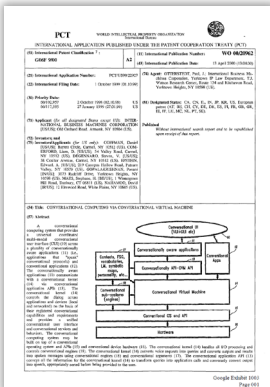
conversational objects/procedures or parallel forms). The first task to have its dialog completed by producing a non-ambiguous query is executed and the corresponding query as interpreted by the other application is stored to activate if the recognized query is rejected by the user.

(Coffman (EX1003), 41:2-5, 42:1-5, 37:2-4)
(Petition at 43, 55-56)



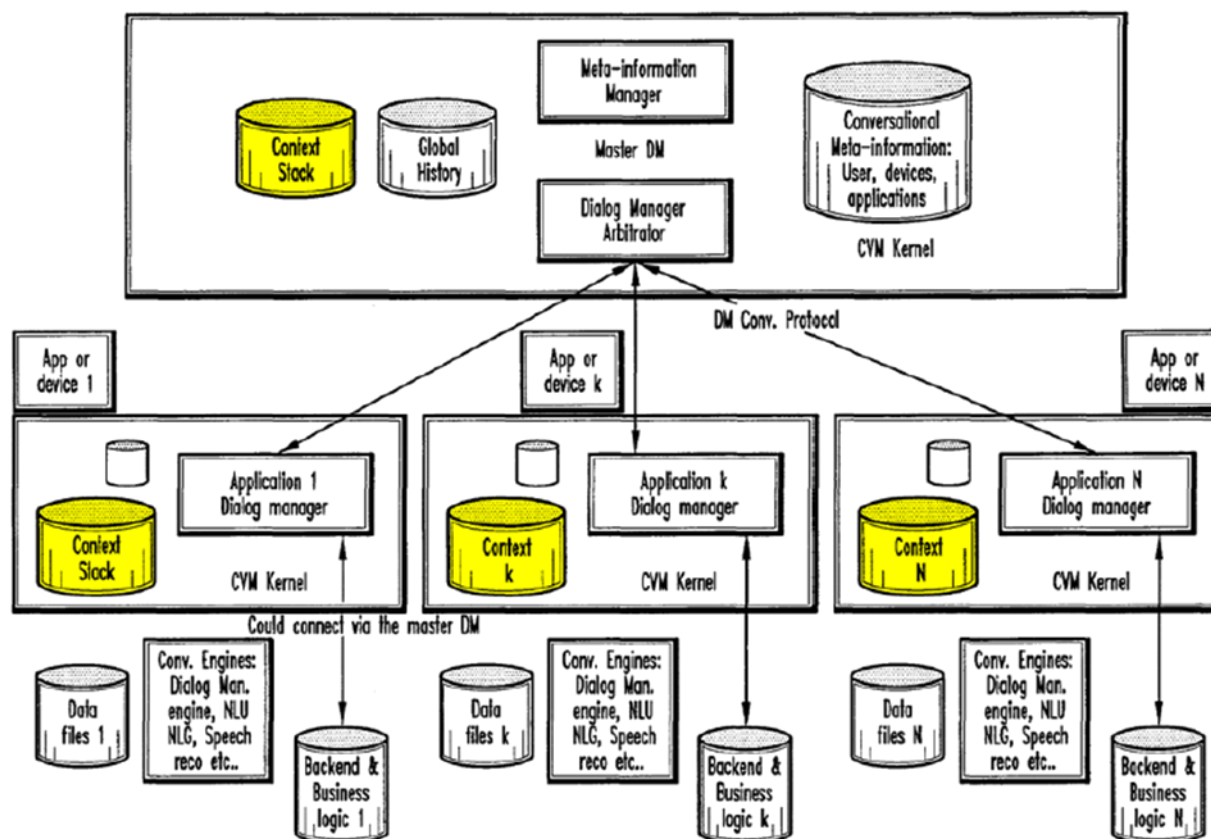
Coffman context stack can be ordered (and re-ordered)

Completion/search for the active context is performed from context to context down the stack. That is, new queries or arguments are compared by the dialog engine by going down the stack until an acceptable match is obtained and optionally confirmation is obtained from the user. As soon a context is found that fits the utterance at the level of the NLU symbolic language, the context becomes active and the corresponding process becomes active. Until the active command is completed, or until a new command is provided, the selected context is marked active, and pushed to the top of the context stack 405. When a message is returned to the user, the context is updated and then pushed to the top of the context stack 405 under the active context. The active context is updated to inform of the existence of a returned value.



Coffman context stacks can be synchronized across devices

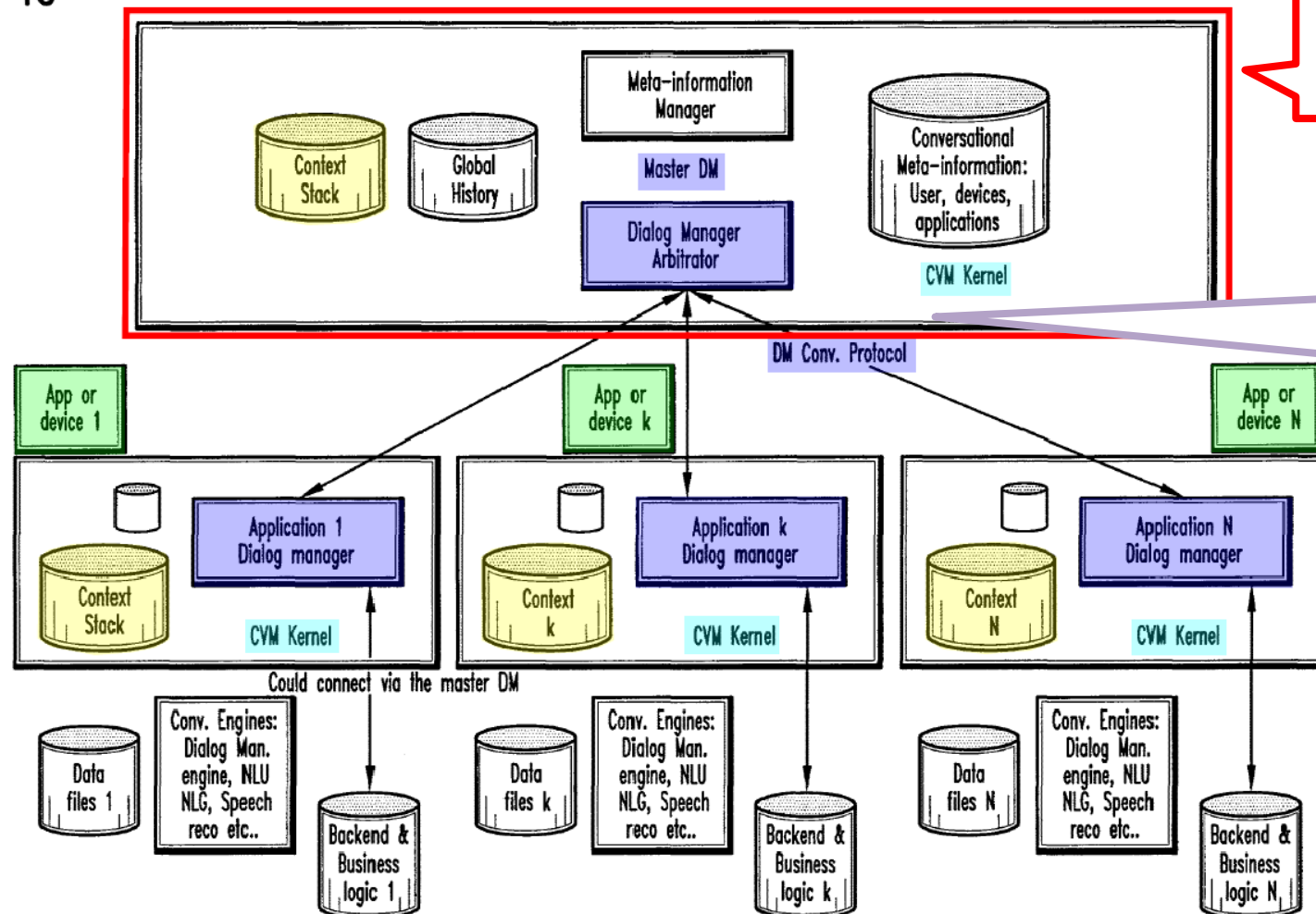
(Coffman, 27:1-7; Schmandt, ¶66.) Figure 10 of Coffman, for example, depicts a conversational system with multiple devices, each device having its own context stack (indicated in yellow highlighting):



(Petition at 28; Coffman (EX1003), Fig. 10)

Coffman context stacks can be synchronized across devices

FIG. 10



Master device

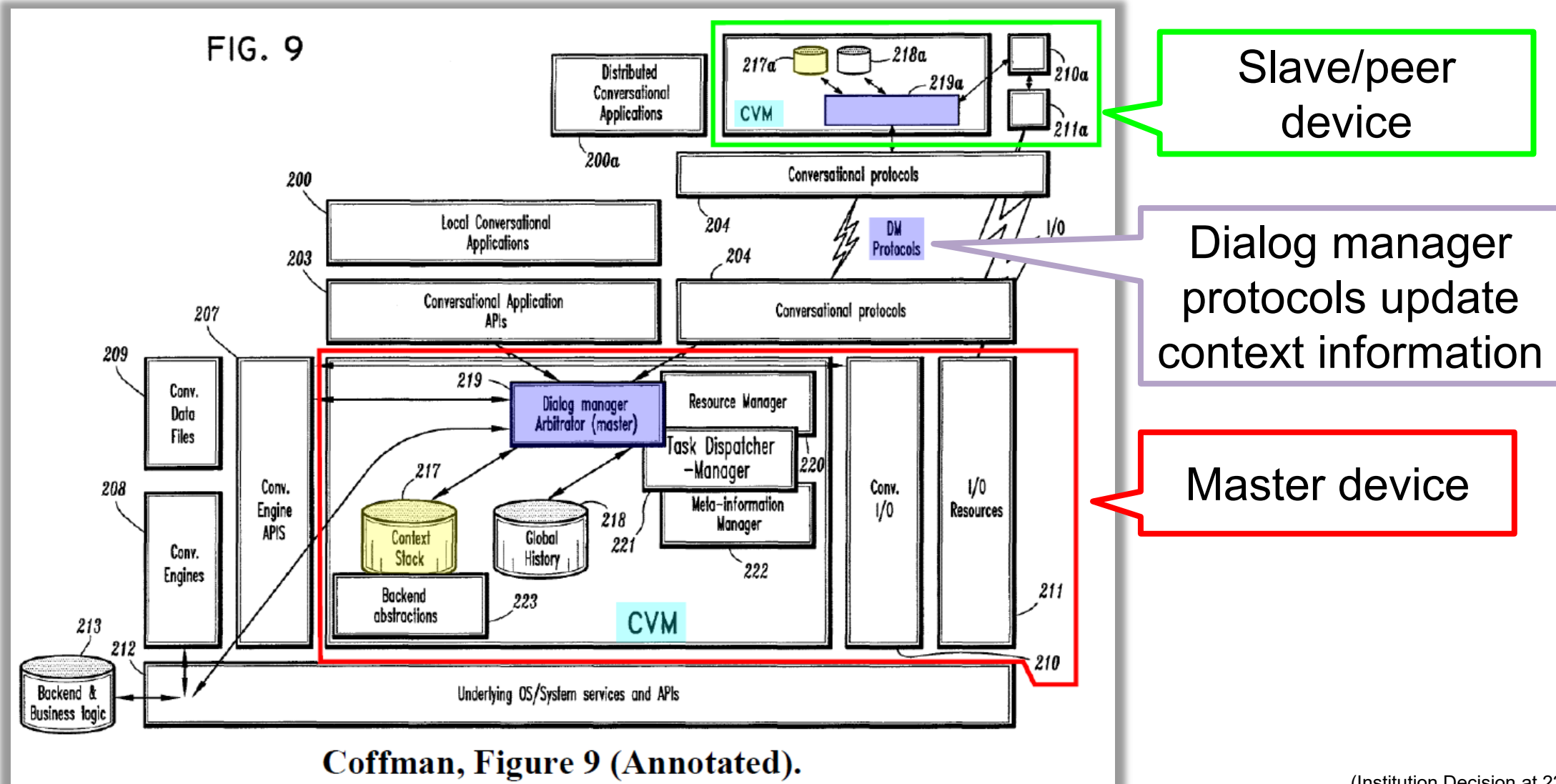
Dialog manager protocols update context information

Slave/peer devices

Coffman, Figure 10 (Annotated).

(Institution Decision at 23)

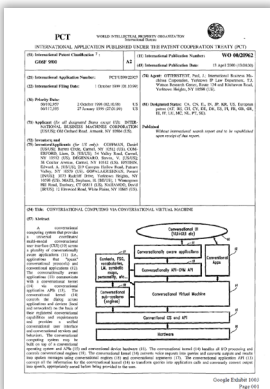
Coffman context stacks can be synchronized across devices



(Institution Decision at 22)

Coffman context stacks can be synchronized across devices

Likewise, applications can exchange engine requirement lists. With a master/slave network configuration, the master dialog manager can compile all the lists and match the functions and needs with conversational capabilities. In addition, context 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. Devices also pass information about their

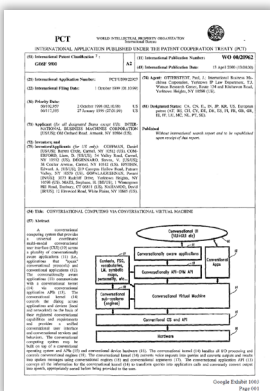


(Coffman (EX1003), 25:10-14)
(Petition at 30-31)

Coffman context stacks can be synchronized across devices

and appropriately execute a query and update the context and history. For instance, the following information can be exchanged: (1) DM architecture registration (e.g., each DM can be a collection of locals DMs); (2) pointers to associated meta-information (user, device capabilities, application needs, etc.); (3) negotiation of DM network topology (e.g., master/slave, peer-to-peer); (4) data files (conversational arguments) if applicable i.e., if engines are used that are controlled by a master DM); (5) notification of I/O events such as user input, outputs to users for transfer to engines and/or addition to contexts; (6) notification of recognition events; (7) transfer of processed input from engines to a master DM; (8) transfer of responsibility of master DM to registered DMs; (9) DM processing result events; (10) DM exceptions; (11) transfer of confidence and ambiguity results, proposed feedback and output, proposed expectation state, proposed action, proposed context changes, proposed new dialog state; (12) decision notification, context update, action update, state update, etc; (13) notification of completed, failed or interrupted action; (14) notification of context changes; and/or (15) data files, context and state updates due to action. In addition, actions, I/O events, backend accesses

(Coffman (EX1003), 27:7-20)
(Petition at 30-31)



Kennewick (EX1004)

(19) United States

(12) Patent Application Publication (10) Pub. No.: US
Kennewick et al. (43) Pub. Date:

(54) SYSTEMS AND METHODS FOR
RESPONDING TO NATURAL LANGUAGE
SPEECH UTTERANCE

Related U.S. A
(60) Provisional application
3, 2002.

(76) Inventors: Robert A. Kennewick, Seattle, WA
(US); David Locke, Redmond, WA
(US); Michael R. Kennewick JR.,
Bellevue, WA (US); Michael R.
Kennewick JR., Bellevue, WA (US);
Richard Kennewick, Woodinville, WA
(US); Tom Freeman, Mercer Island,
WA (US)

Correspondence Address:
MINTZ LEVIN COHN FERRIS GLOVSKY
AND POPEO PC
12010 SUNSET HILLS ROAD
SUITE 900
RESTON, VA 20190 (US)

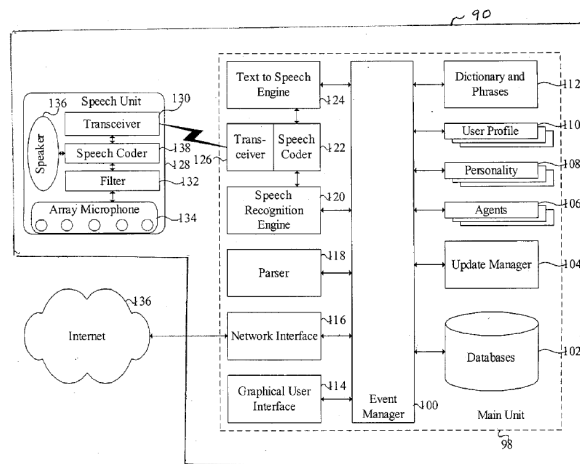
(21) Appl. No.: 10/452,147

(22) Filed: Jun. 3, 2003

(51) Int. Cl.⁷
(52) U.S. Cl.

(57) ABST

Systems and methods for rece
and/or commands and execu
mands. The systems and met
cies of prior art speech query
the application of a comple
query, retrieval, presentation
This environment makes sig
information, domain knowles
data to achieve a natural env
making queries or commands
this integrated approach, a complete speech-based natural
language query and response environment can be created.
The systems and methods creates, stores and uses extensive
personal profile information for each user, thereby improv
ing the reliability of determining the context and presenting
the expected results for a particular question or command.



Google Exhibit 1004
Page 001

(19) United States

(12) Patent Application Publication
Kennewick et al.

(10) Pub. No.: US 2004/0044516 A1
(43) Pub. Date: Mar. 4, 2004

(54) SYSTEMS AND METHODS FOR
RESPONDING TO NATURAL LANGUAGE
SPEECH UTTERANCE

Related U.S. Application Data

(60) Provisional application No. 60/384,388, filed on Jun.
3, 2002.

- Cited for features in claim 11 (and counterpart method claim 27) related to licensing, and features in claims 12-13 and 19 related to a “cognitive model”
- Not disputed by Patent Owner

(Petition at 19)

(19) United States
(12) Patent Application Publication
Lee et al.



US 2002/0087315 A1

(54) COMPUTER-IMPLEMENTED
MULTI-SCANNING LANGUAGE METHOD
AND SYSTEM

(76) Inventors: Victor Wai Leung Lee, Waterloo (CA);
Ottman A. Basir, Kitchener (CA);
Fakhreddine O. Karray, Waterloo
(CA); Jiping Sun, Waterloo (CA); Xing
Jing, Waterloo (CA)

Correspondence Address:
Jones, Day, Reavis and Pogue
North Point
901 Lakeside Avenue
Cleveland, OH 44114 (US)

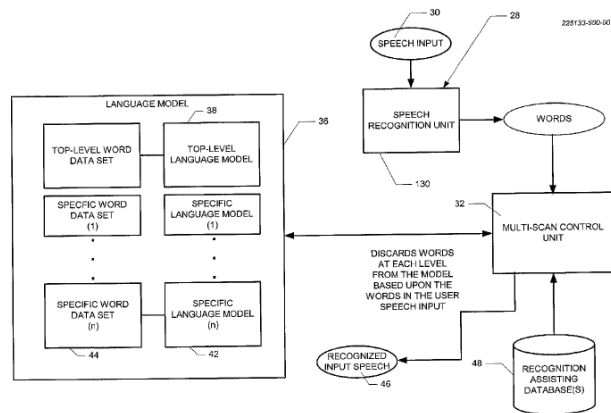
(21) Appl. No.: 09/863,576

(22) Filed: May 23, 2001

Related U.S. Application Data

(63) Non-provisional of provisional application No.
60/258,911, filed on Dec. 29, 2000.

A computer-implemented method for recognition of a user speech input which contains utterances from a language model recognizes utterances from the user speech input which has utterance terms that form a language model is selected from utterances from use of the first language model contains utterance terms from the selected first language model. Subsequent to the selected second language model from the user speech input.



(19) United States

(12) Patent Application Publication
Lee et al.

(10) Pub. No.: US 2002/0087315 A1
(43) Pub. Date: Jul. 4, 2002

(54) COMPUTER-IMPLEMENTED
MULTI-SCANNING LANGUAGE METHOD
AND SYSTEM

Publication Classification

(51) Int. Cl.⁷ G10L 15/14; G10L 15/00
(52) U.S. Cl. 704/256

Google Exhibit 1006
Page 001

- Cited for features in claims 13 and 19 related to a “cognitive model”
- Not disputed by Patent Owner

Key Disputes

Key Disputes

- Whether Coffman discloses “**synchroniz[ing]**” context stacks as claimed

1. A system for processing natural language utterances where recognized words of the natural language utterances alone are insufficient to completely determine one or more commands or requests, the system comprising:
 - [a]** one or more physical processors programmed with one or more computer program instructions which, when executed, cause the one or more physical processors to:
 - [b]** generate a first context stack associated with a first device, the first context stack comprising context information that corresponds to a plurality of prior utterances;
 - [c]** **synchronize the first context stack with a second context stack** associated with a second device such that the context information of the first context stack is updated based on related context information of the second context stack;
 - [d]** receive a natural language utterance associated with a command or request;
 - [e]** determine one or more words of the natural language utterance by performing speech recognition on the natural language utterance; and
 - [f]** determine the command or request based on the one or more words and the updated context information.

Key Disputes (cont'd)

- Whether Coffman discloses “**identify[ing]**” one or more domain agents as claimed
- Whether Coffman discloses generating a response “**using the one or more domain agents**”

3. The system of claim 1, wherein the first context stack includes a plurality of context entries, and wherein the one or more physical processors are further caused to:

identify, from among the plurality of context entries, one or more context entries that correspond to the one or more words, wherein the updated context information includes the one or more context entries.

8. The system of claim 3, wherein the 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

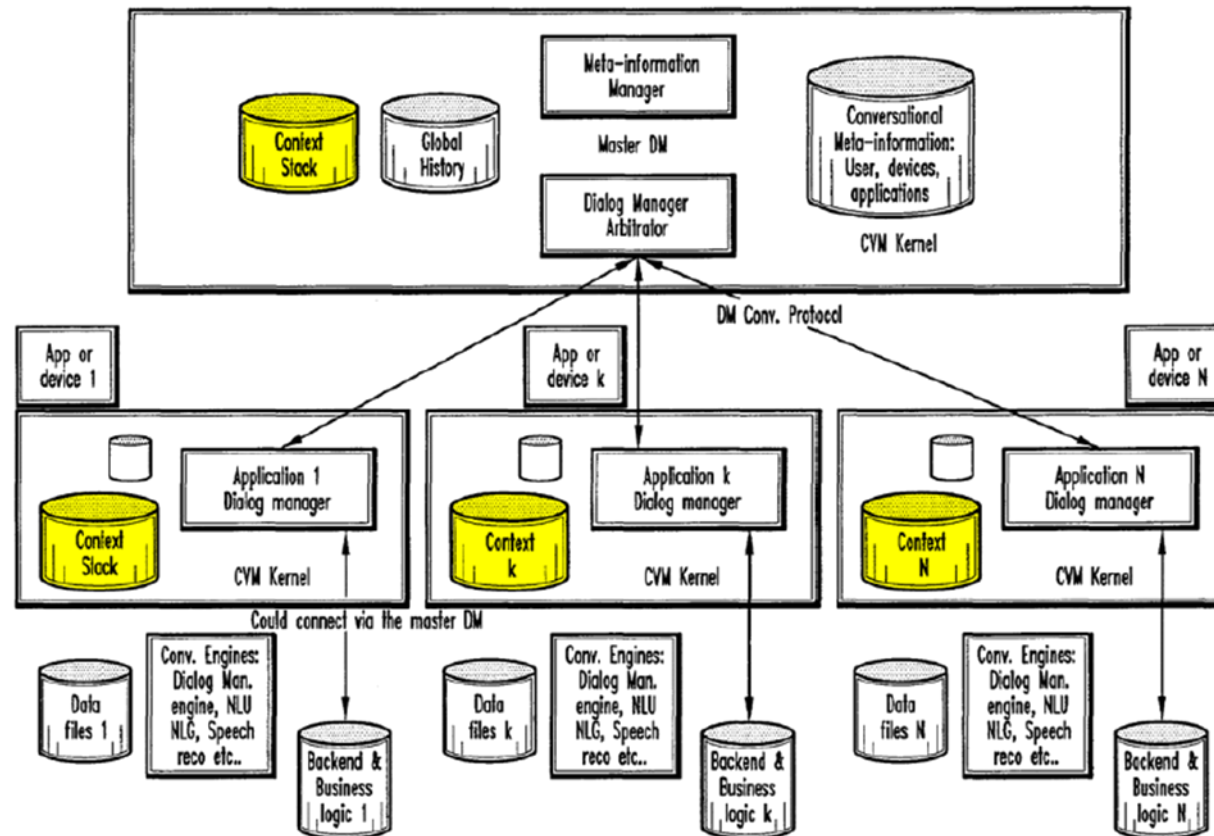
generate a response to the command or request **using the one or more domain agents**.

“synchronizing” context stacks

“synchronizing” context stacks – Petitioner

Coffman system includes multiple devices, each having a context stack

(Coffman, 27:1-7; Schmandt, ¶66.) Figure 10 of Coffman, for example, depicts a conversational system with multiple devices, each device having its own context stack (indicated in yellow highlighting):



(Petition at 28; Coffman (EX1003), Fig. 10)

“synchronizing” context stacks – Petitioner

Coffman context stacks can be synchronized across devices

Likewise, applications can exchange engine requirement lists. With a master/slave network configuration, the master dialog manager can compile all the lists and match the functions and needs with conversational capabilities. In addition, context 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. Devices also pass information about their

(Coffman (EX1003), 25:10-14)
(Petition at 30-31)

“synchronizing” context stacks – Petitioner

Coffman context stacks can be synchronized across devices

and appropriately execute a query and update the context and history. For instance, the following information can be exchanged: (1) DM architecture registration (e.g., each DM can be a collection of locals DMs); (2) pointers to associated meta-information (user, device capabilities, application needs, etc.); (3) negotiation of DM network topology (e.g., master/slave, peer-to-peer); (4) data files (conversational arguments) if applicable i.e., if engines are used that are controlled by a master DM); (5) notification of I/O events such as user input, outputs to users for transfer to engines and/or addition to contexts; (6) notification of recognition events; (7) transfer of processed input from engines to a master DM; (8) transfer of responsibility of master DM to registered DMs; (9) DM processing result events; (10) DM exceptions; (11) transfer of confidence and ambiguity results, proposed feedback and output, proposed expectation state, proposed action, proposed context changes, proposed new dialog state; (12) decision notification, context update, action update, state update, etc; (13) notification of completed, failed or interrupted action; (14) notification of context changes; and/or (15) data files, context and state updates due to action. In addition, actions, I/O events, backend accesses

(Coffman (EX1003), 27:7-20)
(Petition at 30-31)

“synchronizing” context stacks – Patent Owner

Patent Owner argues:

- Examiner concluded during prosecution that synchronizing a context entry does not show synchronization of context stacks
- Synchronization requires same entries in identical order (i.e., “perfect” synchronization)

“synchronizing” context stacks – Response to Patent Owner

Patent Owner argues: Examiner concluded during prosecution that synchronizing a context entry does not show synchronization of context stacks

Petitioner Response:

context across multiple devices” for all pending claims. There was no finding that synchronizing context could never disclose synchronizing context stacks/sets. As the Institution Decision correctly observed, unlike Coffman (which the Examiner never discussed), Mumick did not present a situation involving synchronizing context on a context stack/set—because, according to the Examiner, Mumick did not disclose synchronizing context at all. (DI, p.38 (“[T]he Examiner cited Mumick’s

“synchronizing” context stacks – Response to Patent Owner

Patent Owner argues: Examiner concluded during prosecution that synchronizing a context entry does not show synchronization of context stacks

Petitioner Response (cont'd):

As described in the Notice of Allowability mailed 11/1/10 in Application

12/617506, Mumick et al. (US 2003/0046346) teaches synchronizing plural browsers on a plurality of mobile devices including a state manager that stores navigation histories in a data store so that a user can switch from a visual browser on one device to a voice browser on a second device without losing navigation state, **but synchronization of navigation state is not the claimed synchronization of context across multiple devices** (page 2).

While the claim language at issue is not identical, the limitation in claim 63 is directed to a similar concept which is also more narrow in that not only is context synchronized, but context stacks/sets of distinct devices are synchronized.

(EX1012 at 00447)
(Reply at 2-3)

Notice of Allowability	Application No. 13998,045 Examiner ERIC YEN	Applicant(s) WEDER ET AL Att. UNR 2658	File Status No
-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address-- All claims being allowable, PROSECUTION ON THE MERITS IS NOW REMAINING CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL 85) or other appropriate communication will be mailed in due course. THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS. This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308. 1. ☒ This communication is responsive to Amendment filed 5/21/14. ☐ A declaration(s)/affidavit(s) under 37 CFR 1.130(b) was/were filed on ____. 2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on ____; the restriction requirement and election have been incorporated into this action. 3. ☒ The allowed claim(s) is/are 50-62 and 64-77. As a result of the allowed claim(s), you may be eligible to benefit from the Patent Prosecution Highway program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/prophets.htm, or send an inquiry to CPHFeedback@uspto.gov. 4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f). Certified copies: a) ☐ All b) ☐ Some c) ☐ None of the: 1. ☐ Certified copies of the priority documents have been received. 2. ☐ Certified copies of the priority documents have been received in Application No. ____. 3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)). * Certified copies not received: ____. Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application. THIS THREE-MONTH PERIOD IS NOT EXTENDABLE. 5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted. ☐ Including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date ____. Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(b). 6. ☐ DEPOSIT OF AND/OR INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL. Attachments: 1. ☐ Notice of References Cited (PTO-892) 2. ☒ Information Disclosure Statements (PTO/SB-08) 3. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material 4. ☐ Interview Summary (PTO-413) 5. ☐ Examiner's Amendment/Comment 6. ☒ Examiner's Statement of Reasons for Allowance 7. ☐ Other ____ Page No./Mail Date ____.			

PTOL 37 (Rev. 08-13) Notice of Allowability Part of Paper No./Mail Date 20140301

Google Exhibit 1012
Page 00446

“synchronizing” context stacks – Response to Patent Owner

Patent Owner argues: Synchronization requires same entries in identical order

Petitioner Response (cont’d):

“Perfect” synchronization not required – the claim itself describes what “**synchronizing**” entails:

synchronize the first context stack with a second context stack associated with a second device **such that the context information of the first context stack is updated based on related context information of the second context stack;**

“synchronizing” context stacks – Response to Patent Owner

Patent Owner argues: Synchronization requires same entries in identical order

Petitioner Response (cont’d):

“Perfect” synchronization not required

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); EX1017, ¶11.) 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. (EX1017, ¶11.) Nor

(Reply at 6)

“synchronizing” context stacks – Response to Patent Owner

Patent Owner argues: Synchronization requires same entries in identical order

Petitioner Response (cont’d):

“Perfect” synchronization not required

As the Institution Decision correctly pointed out, this understanding is also

Alternatively, the synchronization may be performed on demand under user control. The synchronization process attempts to determine which version of a data element or an agent are the newest or most up-to-date and propagates that element. Thus, synchronization is an incremental change processes. In some cases, a complete replacement of a database, a portion of the database or of one or more agents may be performed rather than performing a series of incremental updates.

multiple mobile devices 36. Each of the mobile devices may communicate with the context manager 702 to register via registration module 712, wherein registration may indicate events that the mobile devices 36 may subscribe to. The context manager 702 may receive input in Context XML form, for example. The other registered mobile devices 36 may be informed of context changes through context tracking module 714, thereby synchronizing the context across the registered mobile devices 36. According to one embodiment

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. (EX1017, ¶11.) Nor

(’652, 17:25-32, 19:36-44)
(Reply at 6)

“synchronizing” context stacks – Response to Patent Owner

Patent Owner argues: Synchronization requires same entries in identical order

Petitioner Response (cont’d):

“Perfect” synchronization not required

does any other passage in the ’652 patent. (EX1017, ¶11.) The Board correctly rejected Patent Owner’s attempt to read a requirement of “perfect” (or “complete”) synchronization into the claims. *See, e.g., Dayco Products, Inc. v. Total Containment, Inc.*, 258 F. 3d 1317, 1325 (Fed. Cir. 2001) (rejecting argument that claim term “received” required “complete” reception as “impermissibly read[ing] an unclaimed (and unarguably undisclosed) limitation into the claims”).

“synchronizing” context stacks – Response to Patent Owner

Patent Owner argues: Synchronization requires same entries in identical order

Petitioner Response (cont’d):



A patent’s specification “is the single best guide to the meaning of a disputed term” and “the claims themselves provide substantial guidance as to the meaning of particular claim terms.”

Phillips v. AWH Corp., 415 F. 3d 1303, 1314-1315 (Fed. Cir. 2005)

“synchronizing” context stacks – Response to Patent Owner

Patent Owner argues: Synchronization requires same entries in identical order

Petitioner Response (cont’d):

Petitioner’s expert: *Even if* “perfect” synchronization required, Coffman’s synchronization could at least sometimes be “perfect”

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

Reply Declaration of Christopher Schmandt
IPR2024-00752

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

Google Exhibit 1017 - Google v. Dialect
IPR2024-00752 - Page 001

(EX1017, ¶11)
(Reply at 7-8)

“synchronizing” context stacks – Response to Patent Owner

Patent Owner argues: Synchronization requires same entries in identical order

Petitioner Response (cont’d):

Petitioner’s expert: *Even if* “perfect” synchronization required, Coffman’s synchronization could at least sometimes be “perfect”

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.)

Reply Declaration of Christopher Schmandt
IPR2024-00752

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

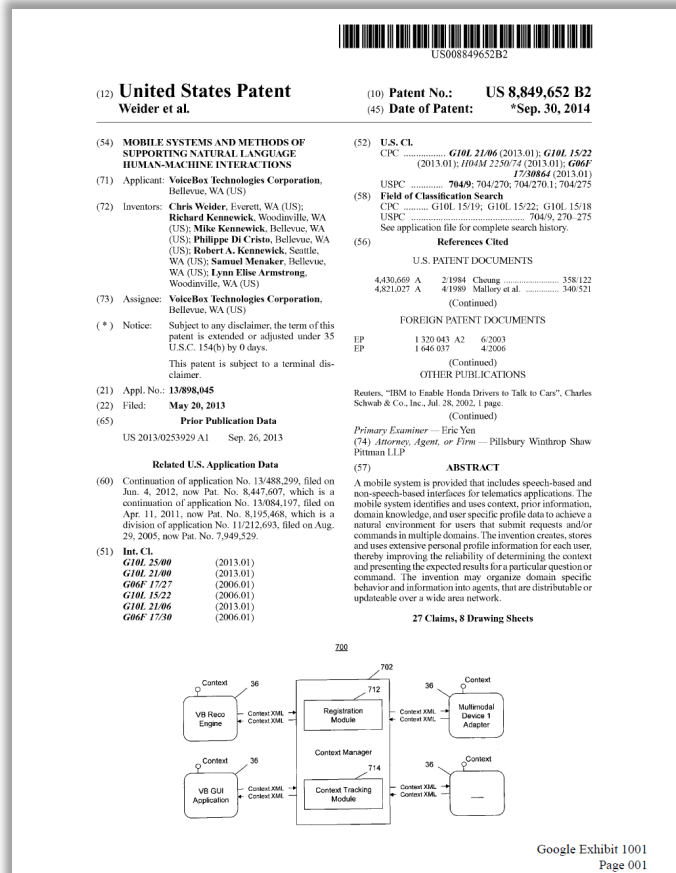
REPLY DECLARATION OF CHRISTOPHER SCHMANDT

Google Exhibit 1017 - Google v. Dialect
IPR2024-00752 - Page 001

(EX1017, ¶11)
(Reply at 7-8)

“identifying” domain agents

“identifying” domain agents



3. The system of claim 1, wherein the first context stack includes a plurality of context entries, and wherein the one or more physical processors are further caused to:

identify, from among the plurality of context entries, one or more context entries that correspond to the one or more words, wherein the updated context information includes the one or more context entries.

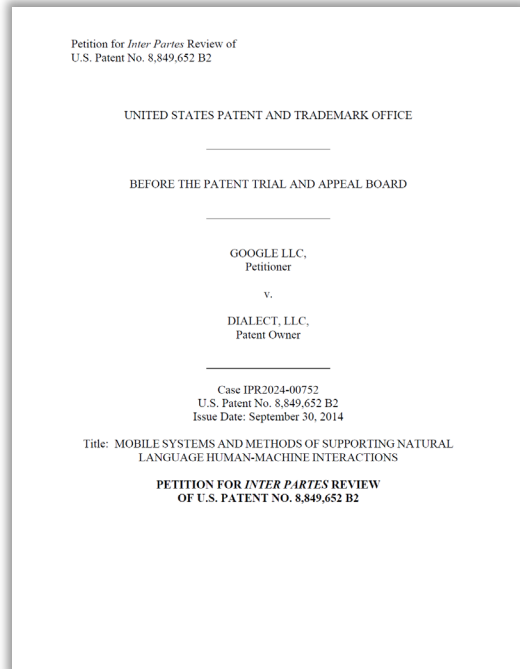
8. The system of claim 3, wherein the one or more physical processors are further caused to:

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.

“identifying” domain agents – Petitioner

Coffman conversationally-aware application = “**domain agent**”



A “**domain agent**” in Coffman corresponds to the executable software of a conversationally-aware application (and associated data, such as vocabulary, grammar, language models and NLU symbolic mapping). (Schmandt, ¶113; Coffman, e.g., 13:26-29 (“The code implementing the applications, API calls and protocol calls includes interpreted and compiled scripts and programs, with library links, conversational logic (as described below) engine call and conversational foundation classes.”).) Conversationally-aware applications “are applications that are specifically programmed for operating with a CVM core layer (or kernel) 14 via conversational application APIs 13.” (Coffman, 12:28-30.) A conversationally

(Petition at 54)

“identifying” domain agents – Petitioner

Coffman identifies conversationally-aware application to process command/query

Coffman describes completing and activating a context entry’s command/query for the associated application to process. (Coffman, 37:2-4; *see also id.*, 42:1-5, 41:2-5, 40:12-15.)

resources. Based on the modality (pointer, keyboard, file, speech), the task dispatcher 402 redirects the stream to the appropriate conversational subsystems or conventional subsystem with speech inputs being transcribed/understood. The output of these subsystems is run down the context stack 405 to extract the active query and complete it. On the other hand, outputs are

. . .

Completion/search for the active context is performed from context to context down the stack. That is, new queries or arguments are compared by the dialog engine by going down the stack until an acceptable match is obtained and optionally confirmation is obtained from the user. As soon a context is found that fits the utterance at the level of the NLU symbolic language, the context becomes active and the corresponding process becomes active. Until the

. . .

conversational objects/procedures or parallel forms). 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.

(Coffman (EX1003), 41:2-5, 42:1-5, 37:2-4)
(Petition at 55-56)

“identifying” domain agents – Patent Owner

Patent Owner argues:

- Coffman’s conversationally-aware applications merely provide a front-end UI to the system—for a user to interact with an application, *the user must select it*
- Coffman’s context stack search identifies a process performed at the CVM, not a conversationally-aware application
- Coffman’s context stacks are application-specific

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s conversationally-aware applications merely provide a front-end UI to the system—for a user to interact with an application, the user must select it

Petitioner Response:

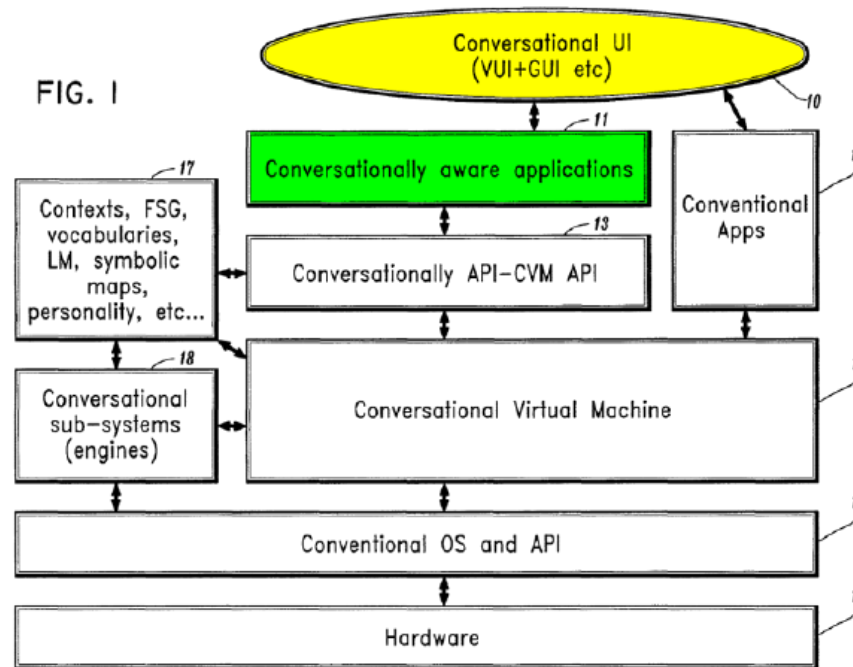
environment. (POR, pp.37-38.) Not so. For starters, this would make no sense. If a conversationally-aware application is just a “gateway[] to use the system” as Patent Owner argues (POR, p.38), why would there be more than one conversationally-aware application for a user to interact with? Other conversationally-aware 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. (EX1017, ¶19.) Patent Owner is confusing Coffman’s separate

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s conversationally-aware applications merely provide a front-end UI to the system—for a user to interact with an application, the user must select it

Petitioner Response (cont’d):

of the system. (EX1017, ¶19.) Patent Owner is confusing Coffman’s separate “conversational user interface”—which can serve as a gateway to use the system—with the “conversationally aware applications”:



(Coffman (EX1003), Fig. 1, 64:3 (“a multi-modal conversational user interface (CUI) for receiving input queries”))
(Reply at 12)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s conversationally-aware applications merely provide a front-end UI to the system—for a user to interact with an application, the user must select it

Petitioner Response (cont’d):

Coffman touts system’s seamless control of multiple Windows applications running in parallel, accomplished using CVM’s dialog manager to identify applications

dialogs and conversations, even if such dialogs are carried over, e.g., a keyboard. It is the conversational computing paradigm that allows a user to seamlessly control multiple Windows applications, for example, running in parallel, even through a dummy terminal display such as VT 100 or a Palm Pilot screen.

. . .

accordance with one embodiment of the present invention. As illustrated, a conversational input interface according to an embodiment of the present invention can process multi-modal input, that is, files/streams/resources, speech via a phone 600, keyboard 601, pointing devices 602, handwriting devices 603, including natural interfaces. This means that all the input and output events across all the modalities are caught and transferred to the dialog manager (that also stores it appropriately in the context stack). Spoken input from a speech client (e.g., telephone 600) is

(Coffman (EX1003), 4:13-16, 37:18-20)
(Reply at 12)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s conversationally-aware applications merely provide a front-end UI to the system—for a user to interact with an application, the user must select it

Petitioner Response (cont’d):

Coffman touts system’s seamless control of multiple Windows applications running in parallel, accomplished using CVM’s dialog manager to identify applications

11. The system of claim 1, wherein the conversational kernel comprises:

a dialog manager for managing dialog across the applications and for selecting an active dialog, context and application based on input queries;

. . .

mouse, video input etc) across all registered applications. In particular, the conversational dialog manager 219 determines what information the user has, which inputs the user presents, and which application(s) should handle the user inputs.

. . .

A dialog controller 404 manages the dialog (conversational = speech and multi- modal: GUI, keyboard, pointer, mouse, video input, etc.) across all the conversational and conventional applications (registered with the task dispatcher/controller 402). As explained above, applications exchange (via API call or negotiation protocols) information about their state, how they interpret a latest input, and the confidence level for such interpretation. The dialog controller 404 manages and determines the active context and application. It also manages the

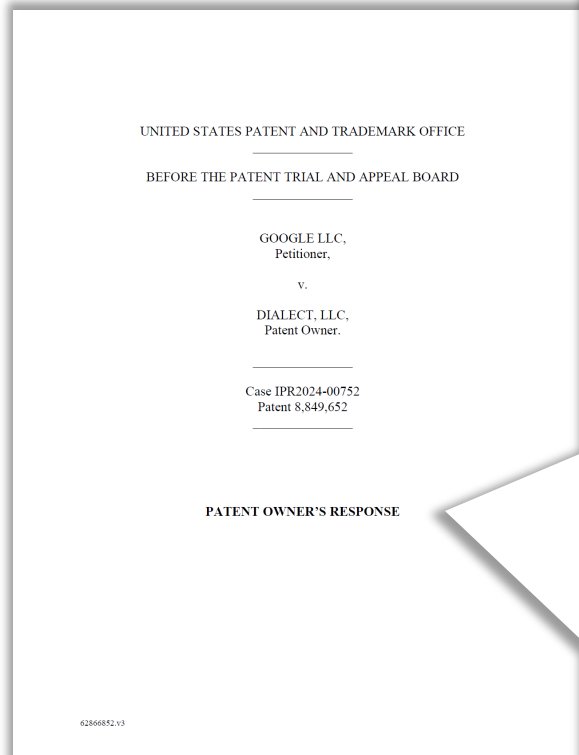
(Coffman (EX1003), 65:12-13, 20:19-23, 31:24-29)
(Reply at 12-13)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stack search identifies a process performed at the CVM, not a conversationally-aware application

Petitioner Response:

None of PO’s cited Coffman passages do anything other than describe context stack structure and that there are associated tasks, threads and processes:



selecting a conversationally-aware application. Occhiogrosso, ¶¶95, 104. Indeed, Coffman associates a “process” with tasks and threads performed *at the CVM*. See Coffman, 42:16-17 (“This stack structure allows to maintain non-ambiguous conversational interactions with multiple tasks, threads or processes.”); 21:1-3 (“The task dispatcher/manager 221 dispatches and coordinates different tasks and processes that are spawned (by the user and machine) on local and networked conventional and conversational resources”); 30:1-2 (“context stack (as well as other CVM components)”); 30:4-5 (“[E]ach new task, process, or thread creates a new stack entry and is associated with a discourse.”); FIG. 2 (illustrating CVM components including task dispatcher/manager 221). This disclosure confirms that

(PO Resp. at 40)
(Reply at 13-14)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stack search identifies a process performed at the CVM, not a conversationally-aware application

Petitioner Response (cont’d):

PO’s cited Coffman passages are consistent with tasks/threads/processes corresponding to conversationally-aware applications

On the contrary, Patent Owner’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” (POR, p.41 (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. (EX1017, ¶22.) This is consistent with Coffman’s system being able

(Coffman (EX1003), 30:26-28)
(Reply at 14)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stack search identifies a process performed at the CVM, not a conversationally-aware application

Petitioner Response (cont’d):

PO’s cited Coffman passages are consistent with tasks/threads/processes corresponding to conversationally-aware applications

applications. (EX1017, ¶22.) 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); EX1017, ¶22.) 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”) (cited in Pet. at 22) *with* Coffman, 36:29-31 (“natural dialog...across multiple applications”) (cited in Pet. at 55); EX1017, ¶22.) It is also consistent with Coffman’s statement that “[t]he context stack is

(Coffman (EX1003), 30:26-28)
(Reply at 14)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stack search identifies a process performed at the CVM, not a conversationally-aware application

Petitioner Response (cont’d):

PO’s cited Coffman passages are consistent with tasks/threads/processes corresponding to conversationally-aware applications

In another aspect, the CVM provides the capability to have natural dialog with NLU, NLG, contexts and mixed-initiatives sorted across multiple tasks, processes and discourses (with multiple domains). A conversational input interface is provided whereby a set of multi-mode running in parallel[.]” (Coffman, 4:13-16 (underlining added); EX1017, ¶22.) It is

The interaction between the task dispatcher/controller 402, dialog controller 404 and context stack 405 of the CVM 401 in processing input and output data streams will now be explained in greater detail. It is to be appreciated that the present invention provides NLU interfaces with contexts and mixed-initiatives sorted across multiple tasks (with multiple domains). More specifically, the present invention provides the capability to have a natural dialog with NLU, NLG and mixed initiative across multiple applications, with multiple domains. In this regard, each application will provide the CVM 401 its own parsing and

(Coffman (EX1003), 6:6-8, 36:29-31)
(Reply at 14)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stack search identifies a process performed at the CVM, not a conversationally-aware application

Petitioner Response (cont’d):

PO’s cited Coffman passages are consistent with tasks/threads/processes corresponding to conversationally-aware applications

¶22.) 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); EX1017, ¶22.)

as appended to the active utterance (speech utterance or any other mode of input). The context stack is updated pending on voice, keyboard, mouse or any other input or command and on the application output.

(Coffman (EX1003), 42:27-29)
(Reply at 14)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stack search identifies a process performed at the CVM, not a conversationally-aware application

Petitioner Response (cont’d):

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. *See Unwired Planet, LLC v. Google*



“[C]ombinations of prior art that sometimes meet the claim elements are sufficient to show obviousness.”

Unwired Planet, LLC v. Google Inc., 841 F.3d 995, 1002 (Fed. Cir. 2016)

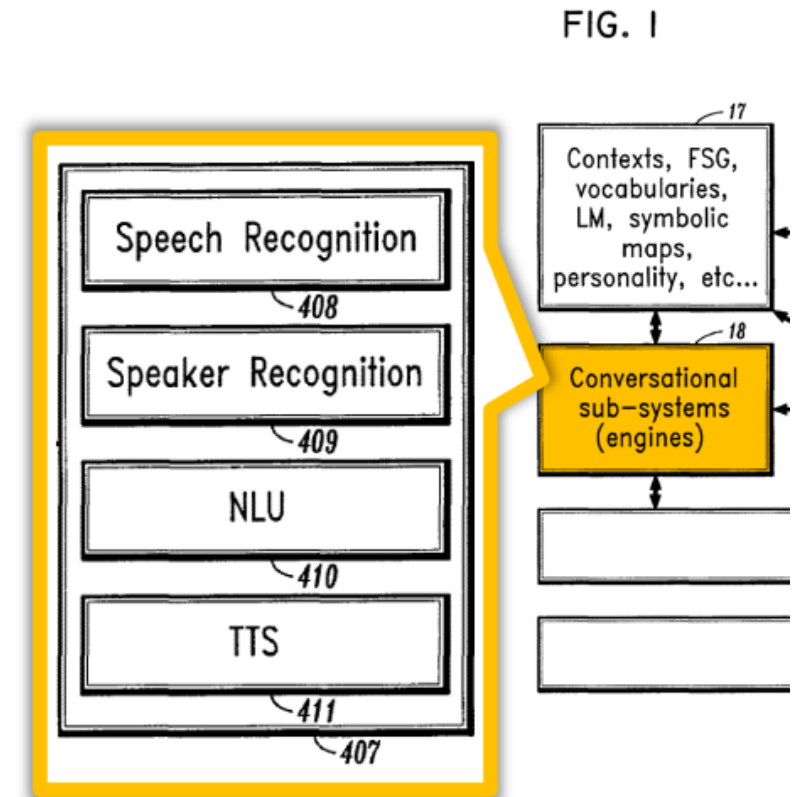
(Reply at 15)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stack search identifies a process performed at the CVM, not a conversationally-aware application

Petitioner Response (cont’d):

As further alleged “confirm[ation] that Coffman processes user request and commands at the CVM,” and not with a conversationally-aware application, Patent Owner 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.” (POR, p.41 (quoting Coffman, 29:7-11).) 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



(Reply at 15)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stack search identifies a process performed at the CVM, not a conversationally-aware application

Petitioner Response (cont’d):

Petition, p.36; EX1017, ¶24.) 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); Petition, p.36; EX1017, ¶24.) 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); EX1017, ¶24.)

(Reply at 15-16)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stacks are application-specific

Petitioner Response:

function. Typically, different devices, CVMs or different applications will have their own dialog manager, context stack 217, 217a and global history 218, 218a. Through the dialog

Patent Owner 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.” (POR, p.42 (citing Coffman, 27:1-2, 21:23-24).) According to Patent Owner, these statements mean that Coffman’s context stacks must each be application-specific. (POR, p.42 (“a context stack for one application”).) Patent Owner is again mistaken—the

(Reply at 16)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stacks are application-specific

Petitioner Response (cont’d):

Coffman does not require that every context stack is limited to a single application

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. (EX1017, ¶26.) This understanding is also consistent with the figure

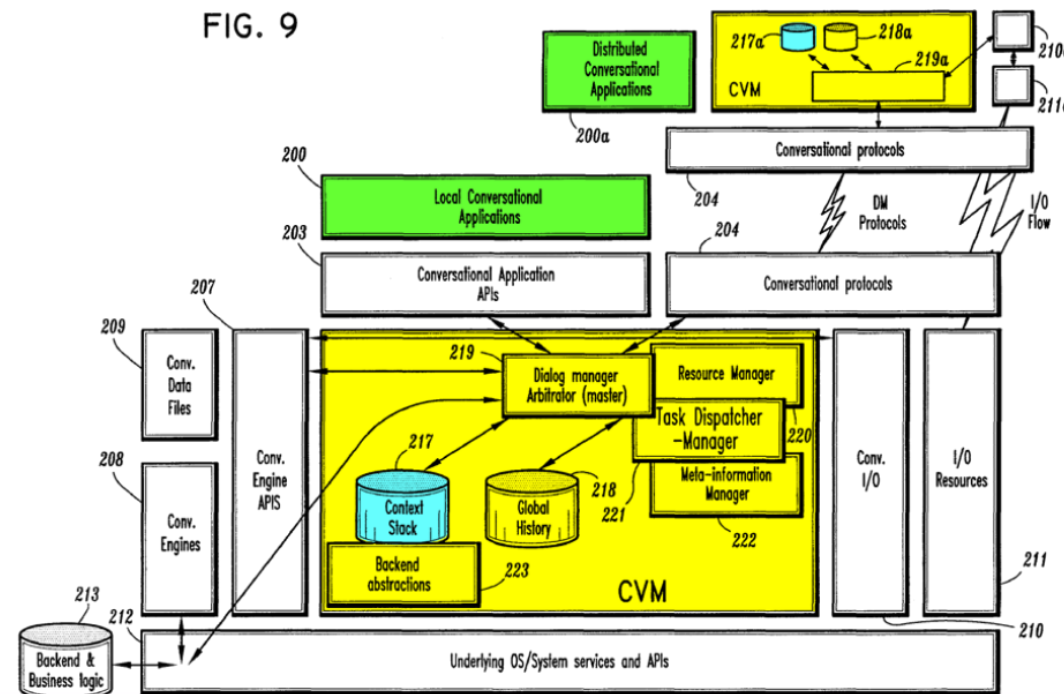
(Reply at 16-17)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stacks are application-specific

Petitioner Response (cont’d):

Coffman does not require that every context stack is limited to a single application applications. (EX1017, ¶26.) 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:



(Reply at 16-17)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stacks are application-specific

Petitioner Response (cont’d):

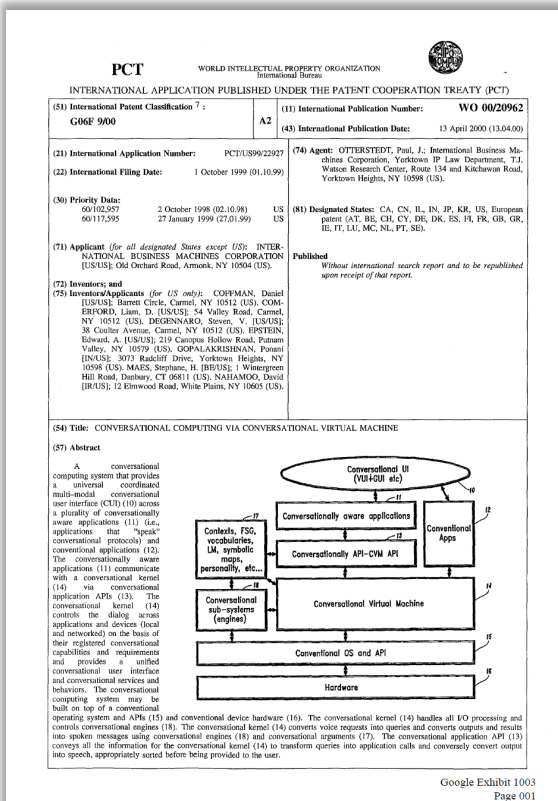
PO’s cited Coffman statement referring to *alternative* “stand-alone” type of conversationally-aware application

The distributed system further comprises **a conversational browser 713 which is a**

compelling speech enabled applications that can operate with CVM. A conversational browser can run on top of a CVM and interprets CML to build a conversational dialog while presenting a CML page. As shown in Fig. 7, and as explained in detail in the above incorporated IBM Docket No. YO998-392P patent application, legacy applications 700 can be accessed via a conversational transcoder proxy to transcode conventional formats like HTML or DB2 into XML. The conversational browser interprets CML (conversational mark-up language), which is a speech markup language based on XML specifications. It can be viewed as one of the most compelling applications to run on top of CVM. **The conversational browser can be stand-alone**

applications carrying its own CVM. CML allows new experienced application developers to

(Coffman (EX1003), 59:12-13)
(Reply at 17)



“identifying” domain agents – Response to Patent Owner

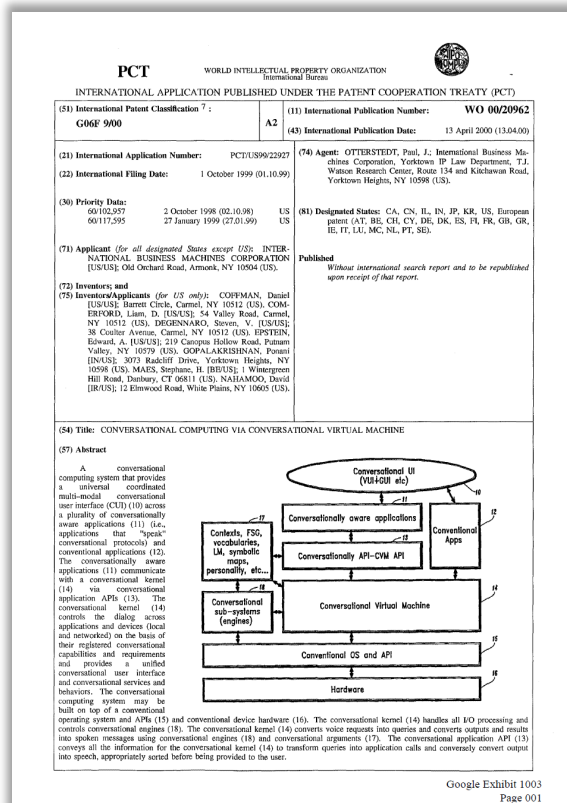
Patent Owner argues: Coffman’s context stacks are application-specific

Petitioner Response (cont’d):

PO’s cited Coffman statement is a restatement 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. Conventional calls to the underlying operating system could be captured and implemented with CVM, which allows portability. In this instance, CVM is configured as a stand-alone platform for existing platforms.

(Coffman (EX1003), 28:10-18)
(Reply at 18-19)



“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stacks are application-specific

Petitioner Response (cont’d):

PO’s cited Coffman statement is a restatement 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. Conventional calls to the underlying operating system could be captured and implemented with CVM, which allows portability. In this instance, CVM is configured as a stand-alone platform for existing platforms.

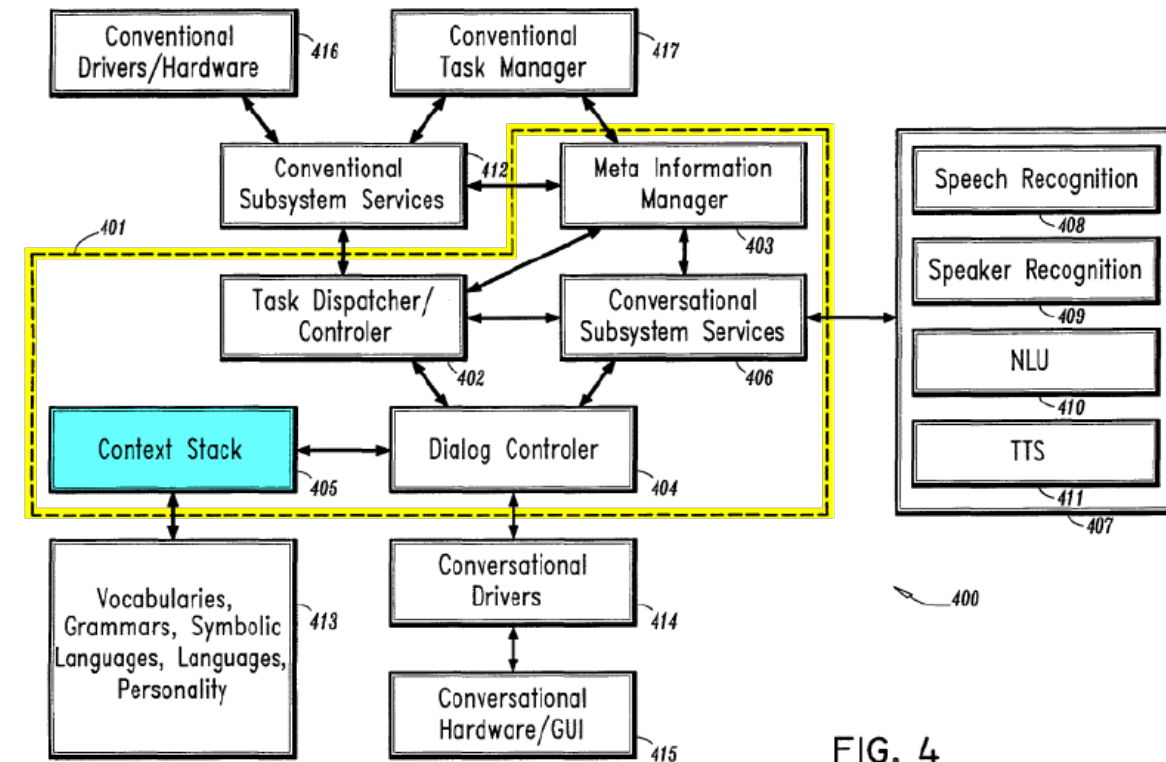


FIG. 4

(Coffman (EX1003), 28:10-18, Fig. 4)
(Reply at 18-19)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stacks are application-specific

Petitioner Response (cont’d):

(highlighting added); EX1017, ¶27.) 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. (EX1017, ¶27.) 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); Petition, p.44; EX1017, ¶27; *see also id.*, Fig. 4 (showing single context stack 405).) Patent Owner’s unduly narrow interpretation

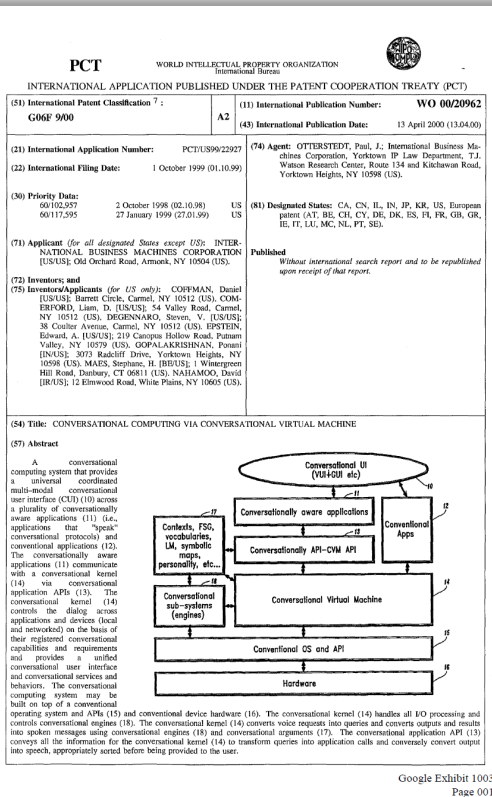
(Reply at 19)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stacks are application-specific

Petitioner Response (cont’d):

Even if PO were correct that a context stack in Coffman is limited to a single application, Coffman explains that user input may not necessarily match any entry in a context stack, and if not, other context stacks are searched:



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 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 (EX1003), 55:27-56:3)
(Reply at 19-20)

“identifying” domain agents – Response to Patent Owner

Patent Owner argues: Coffman’s context stacks are application-specific

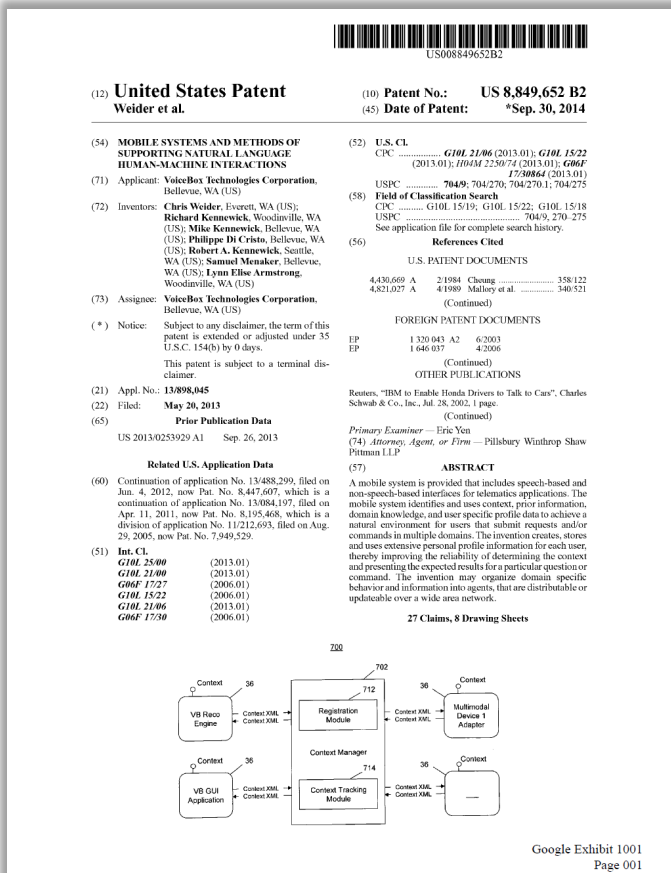
Petitioner Response (cont’d):

(Coffman, 55:27-56:3 (underlining added); Petition, p.44.) 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. (EX1017, ¶25; *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); Petition, p.47.)

(Reply at 20)

generating a response
using domain agent(s)

“generating a response”



3. The system of claim 1, wherein the first context stack includes a plurality of context entries, and wherein the one or more physical processors are further caused to:

identify, from among the plurality of context entries, one or more context entries that correspond to the one or more words, wherein the updated context information includes the one or more context entries.

8. The system of claim 3, wherein the 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.

“generating a response” – Petitioner

Coffman teaches conversationally-aware application returns output that is processed (e.g., text-to-speech) to provide to the user:

drivers) of the conventional OS 15. In general, conversational sub-systems 18 are responsible for converting voice requests into queries and converting **outputs and results into spoken messages** using the appropriate data files 17 (e.g., contexts, finite state grammars, vocabularies, language models, symbolic query maps etc.) The conversational application API 13 conveys all the information for the CVM 14 to transform queries into **application calls and conversely converts output into speech, appropriately sorted before being provided to the user.**

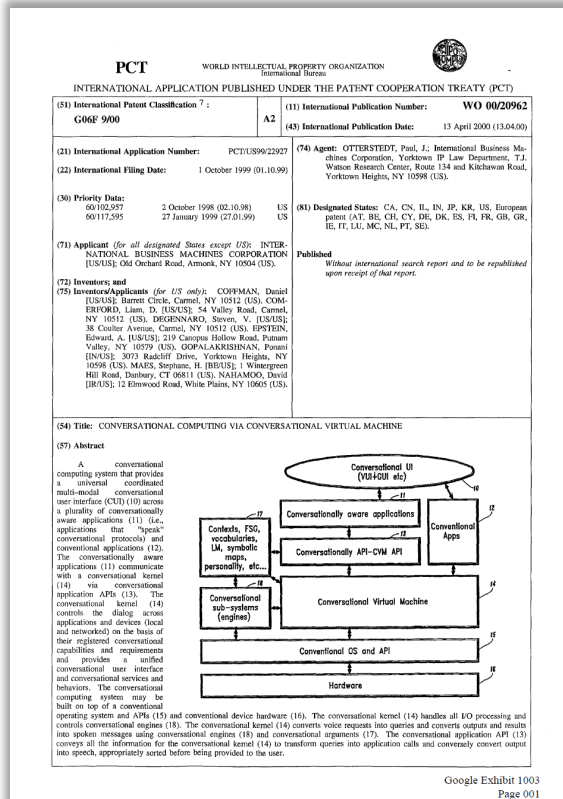
...

coordination among the devices. Again, the CVM automatically handles all the input/output issues, including the conversational and conventional subsystems. Again, the 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.**

...

With regard to **conversational output dispatches and interface, the CVM 401 produces output to files/streams/resources, display (single or multi-windows, GUI, color, images, movies), audio. Individually, each output stream behaves conventionally.** However, according

(Coffman (EX1003), 13:8-13, 58:8-11, 39:3-5)
(Petition at 56-57)



“generating a response” – Petitioner

Coffman teaches conversationally-aware application returns output that is processed (e.g., text-to-speech) to provide to the user:

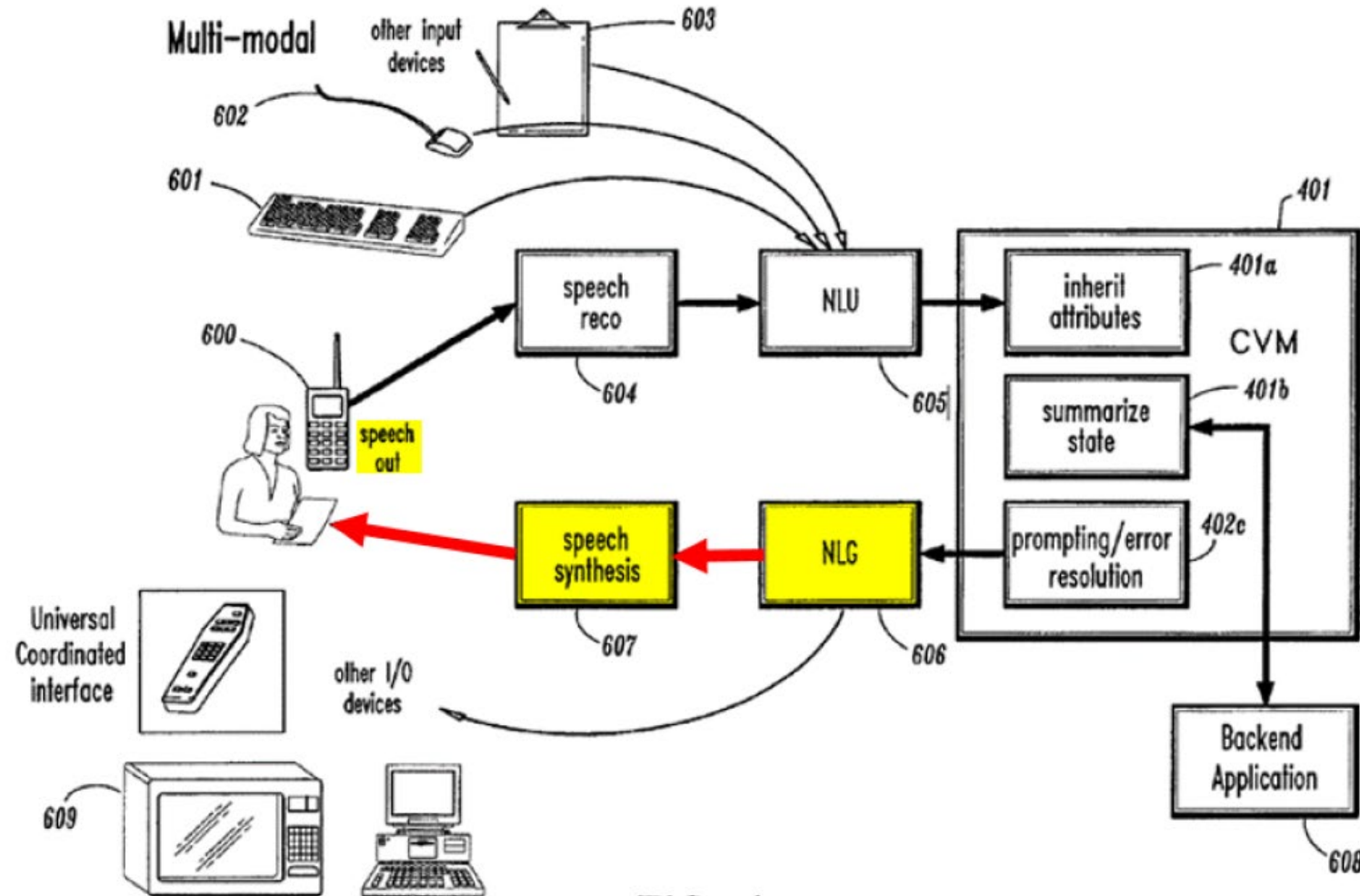
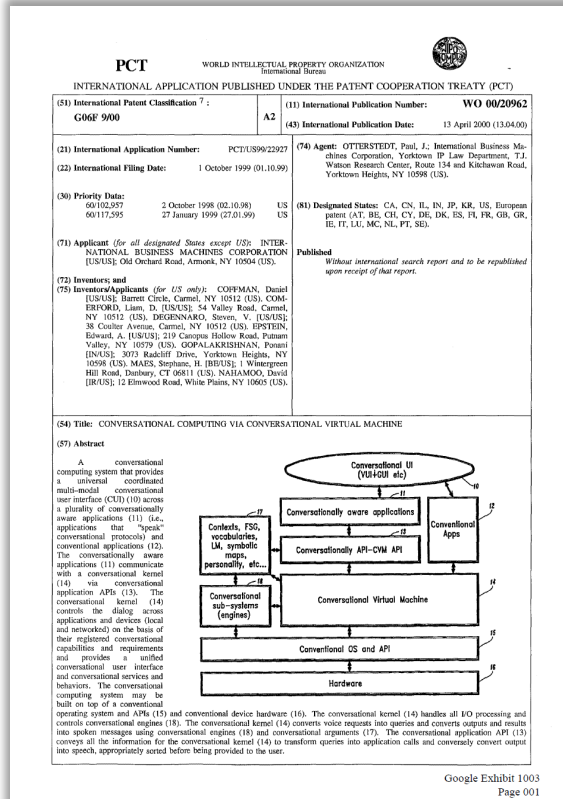


FIG. 6

(Coffman (EX1003), Fig. 6)
(Petition at 56-57)

“generating a response” – Patent Owner

Patent Owner argues:

- Coffman does not disclose selection of an application based on a determined meaning of a user's utterance
- Coffman's disclosures at 13:8-13 and 58:9-11 describe actions at the CVM 14, not a conversationally-aware application

“generating a response” – Response to Patent Owner

Patent Owner argues: Coffman does not disclose selection of an application based on a determined meaning of a user’s utterance

Petitioner Response:

Repeat of earlier PO argument – Coffman does disclose selecting application based on determined meaning

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 “manage[s] 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* Petition, p.55.)

(Reply at 21)

“generating a response” – Response to Patent Owner

Patent Owner argues: Coffman does not disclose selection of an application based on a determined meaning of a user’s utterance

Petitioner Response (cont’d):

Petitioner’s expert:

(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,

Reply Declaration of Christopher Schmandt
IPR2024-00752

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

Google Exhibit 1017 - Google v. Dialect
IPR2024-00752 - Page 001

(EX1017, ¶28)
(Reply at 21)

“generating a response” – Response to Patent Owner

Patent Owner argues: Coffman does not disclose selection of an application based on a determined meaning of a user’s utterance

Petitioner Response (cont’d):

Patent Owner’s expert:

Q So just putting aside the verbal response part, though, what part of Coffman's system would actually look up the value of a particular cell in the spreadsheet file, would it be the conversationally aware spreadsheet application or something else?

MR. HELGE: Object to form.

THE WITNESS: I think once the speech recognition engine that the CVM accesses determines that it is that particular spreadsheet and that particular cell, then the 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.

(EX1018, 36:13-24)
(Reply at 21-22)

1 UNITED STATES PATENT AND TRADEMARK OFFICE
2 BEFORE THE PATENT TRIAL AND APPEAL BOARD
3
4 GOOGLE LLC,)
5 Petitioner,) Case IPR2024-00752
6 VS.) U.S. Patent
7 DIALECT LLC,) No. 8,849,652 B2
8 Patent Owner.)
9) Case IPR2024-00753
10) U.S. Patent
11) No. 7,640,160 B2
12
13 REMOTE DEPOSITION OF
14 BENEDICT OCCHIOGROSSO
15 WEDNESDAY, APRIL 9, 2025
16
17
18
19
20
21
22
23 STENOGRAPHIC REPORTER: CHRISTA YAN, CSR NO. 14316, RPR
24
25

“generating a response” – Response to Patent Owner

Patent Owner argues: Coffman’s disclosures at 13:8-13 and 58:9-11 describe actions at the CVM 14, not a conversationally-aware application

Petitioner Response:

Coffman 13:8-13 describes how CVM subsystems convert output/results from an application into speech:

drivers) of the conventional OS 15. In general, conversational sub-systems 18 are responsible for converting voice requests into queries and converting outputs and results into spoken messages using the appropriate data files 17 (e.g., contexts, finite state grammars, vocabularies, language models, symbolic query maps etc.) The conversational application API 13 conveys all the information for the CVM 14 to transform queries into application calls and conversely converts output into speech, appropriately sorted before being provided to the user.

Coffman 58:9-11 likewise describes how the CVM converts user input into “application calls” and converts the output from the application into speech

coordination among the devices. Again, the CVM automatically handles all the input/output issues, including the conversational and conventional subsystems. Again, the 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.

(Reply at 23)

“generating a response” – Response to Patent Owner

Patent Owner argues: Coffman’s disclosures at 13:8-13 and 58:9-11 describe actions at the CVM 14, not a conversationally-aware application

Petitioner Response (cont’d):

EX1017, ¶29.) These passages from Coffman are consistent with the admissions of Patent Owner’s own expert, discussed above, that a conversationally aware application would be invoked to respond to a user command/request, based on determining the meaning of the user’s utterance. (EX1018, 36:13-24; *see generally id.*, 35:19-36:24.)

Thank you