

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS, INC.,
Petitioner,

IPR2026-00208
Patent No. 9,578,092

**PETITION FOR *INTER PARTES* REVIEW
UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104**

TABLE OF CONTENTS

Table of Contents	2
Petitioner’s Exhibit List	4
I. Introduction.....	7
II. Grounds for standing	9
III. Note.....	10
IV. Summary of the ’092 patent	10
V. Effective priority date	10
VI. Prosecution history	11
VII. Level of ordinary skill in the art	12
VIII. Background.....	12
IX. Claim construction.....	17
X. Relief requested and reasons therefore.....	17
XI. Identification of how the claims are unpatentable.....	18
A. Challenged claims	18
XII. Statutory grounds.....	18
A. Statement of material facts	19
B. Ground 1	22
1. Abuan	22
2. Guzman	24
3. Reasons to combine Abuan and Guzman	26
4. Lawson	28
5. Reasons to combine Abuan, Guzman, and Lawson.....	30
6. Independent claim 1	31

7.	Independent claim 7	54
8.	Independent claim 12	64
9.	Independent claim 22	69
C.	Ground 2	72
1.	Ludwig	72
2.	Reasons to combine Abuan with Ludwig	75
3.	Dependent claims 6, 11, and 21	77
4.	Claim 30	80
XIII.	Conclusion	81
XIV.	Mandatory notices	82
A.	Real party-in-interest	82
B.	Related matters	82
C.	Lead and back-up counsel and service information	82
	Claims Appendix	84
	Certificate of Word Count	93
	Certificate of Service	94

PETITIONER’S EXHIBIT LIST

Ex.1001	U.S. 9,578,092
Ex.1002	Prosecution History of U.S. 9,578,092
Ex.1003	Declaration of Bruce McNair under 37 C.F.R. § 1.68
Ex.1004	<i>Curriculum Vitae</i> of Bruce McNair
Ex.1005	U.S. 2011/0249077 to Abuan et al.
Ex.1006	U.S. 2012/0092438 to Guzman Suarez et al.
Ex.1007	U.S. 2013/0019297 to Lawson et al.
Ex.1008	U.S. 5,689,641 to Ludwig et al.
Ex.1009	D. E. Knuth, Selected Papers on Computer Science (Cambridge Univ. Press), 1996
Ex.1010	B. W. Boehm, Software Engineering Economics (Prentice-Hall, Inc.), 1981
Ex.1011	U.S. 5,339,431
Ex.1012	U.S. 5,615,400
Ex.1013	U.S. 5,946,486
Ex.1014	U.S. 6,243,764
Ex.1015	U.S. 5,802,367
Ex.1016	U.S. 5,218,697
Ex.1017	U.S. 5,498,003
Ex.1018	U.S. 6,163,858
Ex.1019	U.S. 2012/0019610
Ex.1020	Plaintiff’s Disclosure of Asserted Claims and Infringement Contentions, <i>Damaka, Inc. v. Cisco Systems, Inc.</i> , No. 2:25-cv-00593 (E.D. Tex. Nov. 13, 2025)
Ex.1021	U.S. 2008/0178264
Ex.1022	U.S. 2005/0235044

Ex.1023	U.S. 2006/0037064
Ex.1024	U.S. 2008/0201438
Ex.1025	U.S. 2012/0170722
Ex.1026	U.S. 2005/0265322
Ex.1027	U.S. 6,314,402
Ex.1028	U.S. 2005/0060183
Ex.1029	U.S. 2009/0193394
Ex.1030	U.S. 2008/0022327
Ex.1031	U.S. 2002/0199019
Ex.1032	U.S. 6,101,607
Ex.1033	U.S. 6,311,283
Ex.1034	U.S. 6,295,048
Ex.1035	U.S. 2009/0310023
Ex.1036	U.S. 2008/0127183
Ex.1037	U.S. 7,574,202
Ex.1038	U.S. 7,461,144
Ex.1039	U.S. 2007/0185998
Ex.1040	A. V. Aho et al., Principles of Compiler Design, (Addison-Wesley), 1977
Ex.1041	IEEE 100, The Authoritative Dictionary of IEEE Standard Terms, 7 th Ed. (IEEE Press), 2000
Ex.1042	A. Silberschatz et al., Operating System Concepts, 8 th Ed. (John Wiley & Sons, Inc.), 2009
Ex.1043	U.S. 6,820,235
Ex.1044	<i>Intentionally Omitted</i>
Ex.1045	U.S. Patent No. 5,835,435
Ex.1046	U.S. Patent No. 5,742,641
Ex.1047	<i>Intentionally Omitted</i>

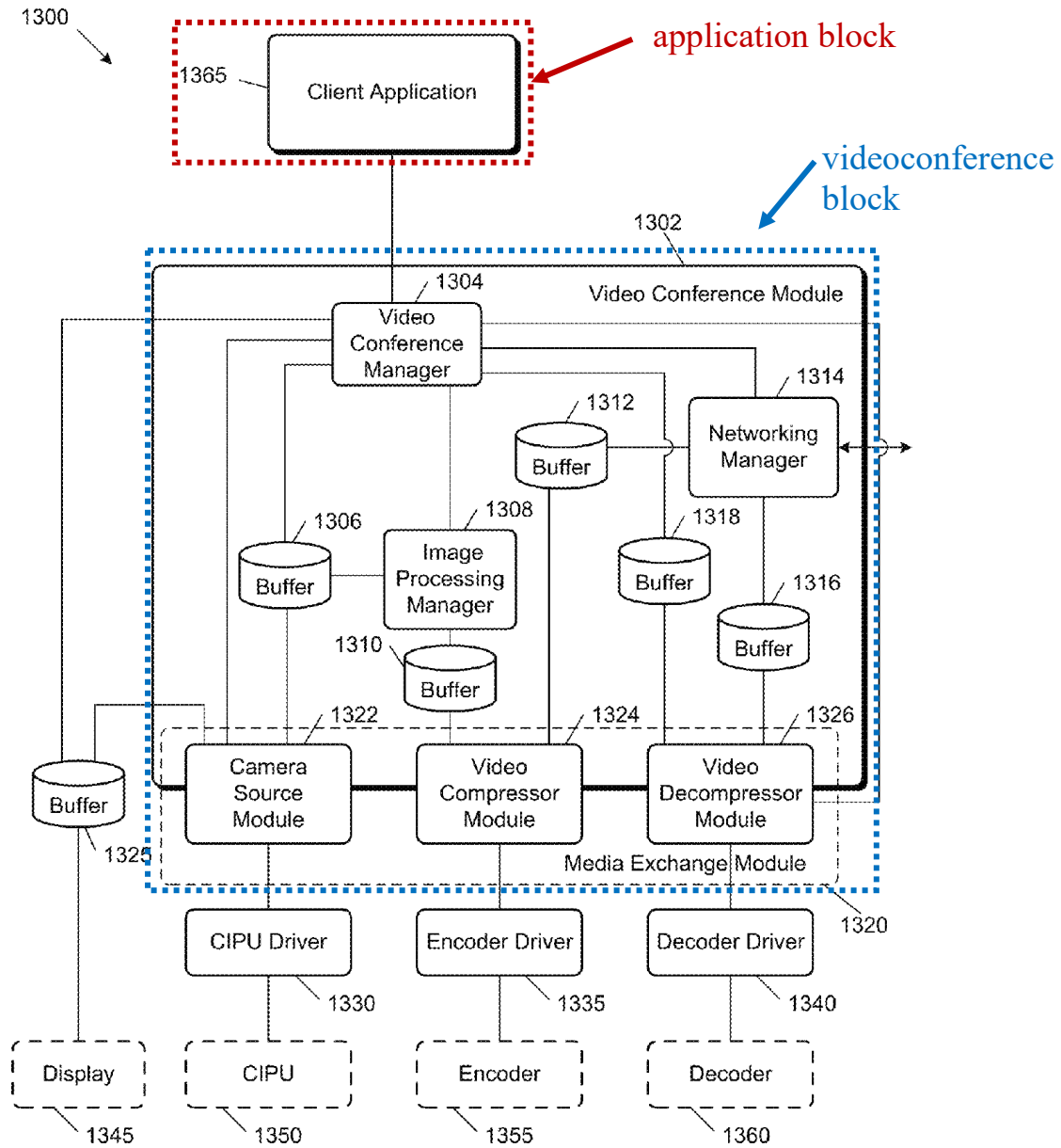
Ex.1048	<i>Intentionally Omitted</i>
Ex.1049	U.S. 2014/0184723

I. Introduction

Pursuant to 35 U.S.C. §§ 311, 314(a), and 37 C.F.R. § 42.100, Cisco Systems, Inc. (“Petitioner”) respectfully requests that the Board review and cancel as unpatentable under 35 U.S.C. §103 claims 1-30 (hereinafter, the “Challenged Claims”) of U.S. 9,578,092 (the “’092 patent,” Ex.1001).

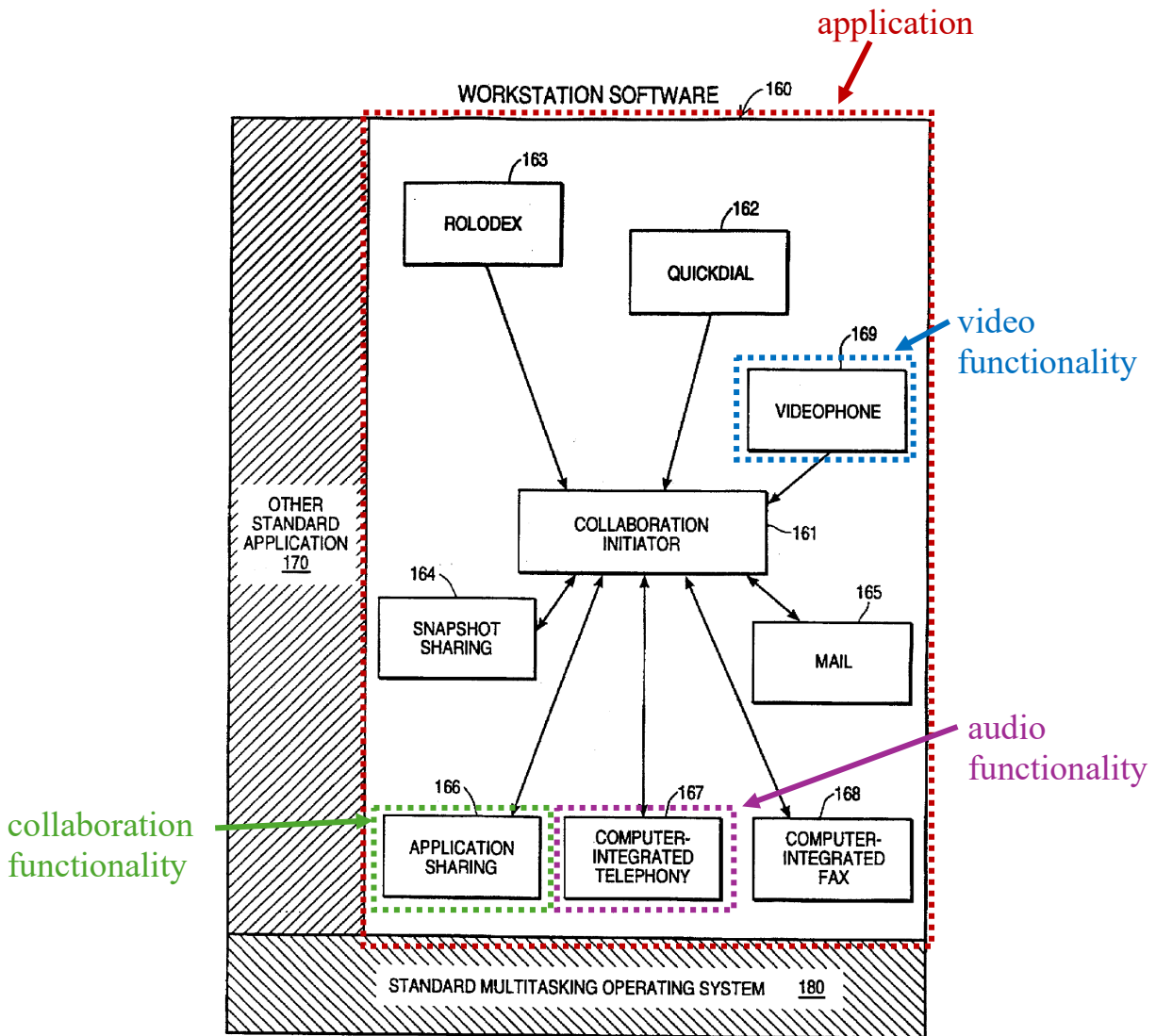
The ’092 patent is directed to “improving functionality in software applications.” Ex.1001, Abstract. The patent acknowledges that one known way to improve software was to rework an existing application to add “desired functionality” by incorporating additional executable instructions that “provide such additional functionality.” Ex.1001, 3:19-34. The claimed innovation does the same thing, except that the added instructions are characterized as being a “function block” that is added to a “superblock.” Ex.1001, 4:13-22. The superblock “may be an application,” while the function block provides capabilities for “audio, video, collaboration..., and/or other functions.” Ex.1001, 2:46-48, 4:22-27.

Organizing computer software into blocks of code was not new. The prior art figure below (dating from 2010) shows an application block designed to use video functions—specifically, videoconference functionality—provided by a separate block. Ex.1005, FIG. 13.



Ex.1005, Fig. 13 (annotated).

Nor was the modular design of software applications in any way “new” even in 2010. The figure below, dating from 1993, shows the modular design of an application with discrete blocks providing audio, video, and collaboration functionality. Ex.1008, FIG. 20.



Ex.1008, Fig. 20 (annotated).

Because the Challenged Claims merely recite obvious combinations of well-known concepts, Petitioner asks the Board to institute trial and find the claims unpatentable.

II. Grounds for standing

Petitioner certifies that the '092 patent is eligible for IPR and that Petitioner is not barred or estopped from requesting IPR of the Challenged Claims.

III. Note

Petitioner cites to exhibits' original page numbers. **Emphasis** in quoted material has been added. Claim terms are presented in *italics*.

IV. Summary of the '092 patent

The '092 patent purports to describe an improved “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:42-43. The improvement consists of a modular approach to software using a “superblock application” and a “function block” that provides “one or more functions (e.g., capabilities)” to the superblock. Ex.1001, 4:13-27. The function block comprises one or more blocks of executable instructions. Ex.1001, 4:60-5:2. The superblock may access the function block instructions by calling an application programming interface (API) and “without switching context to another application on the computing entity.” Ex.1001, 5:61-6:1, Abstract. For example, when the function block “provide[s] ... video capabilities for the superblock 104,” “visual functions may [] be presented without leaving the superblock application display,” i.e., with “no need to switch context.” Ex.1001, 4:28-35; Ex.1003, ¶¶33-35.

V. Effective priority date

The '092 patent is based on an application filed October 19, 2016. It claims the benefit of a provisional application filed July 16, 2013. Determining whether

the '092 patent is entitled to an earlier priority date is unnecessary in this proceeding, however, as all the cited and relied-upon prior art predate July 2013. Accordingly, Petitioner has not undertaken a priority date analysis. Petitioner does not waive any right or opportunity it may have to dispute the priority date of the '092 patent in this or another forum where the issue is relevant.

VI. Prosecution history

The '092 patent application was filed on October 19, 2016 and claims priority to a provisional application filed on July 16, 2013. Ex.1002, 132. The Examiner did not issue any office actions during prosecution. Ex.1002. The Applicant filed a terminal disclaimer after an Examiner-initiated Examiner interview. Ex.1002, 20, 123-124. The Notice of Allowance was mailed on January 3, 2017. Ex.1002, 11-19. In the Notice of Allowance, the “Reasons for Allowance” stated that the following limitations of claim 1 (and the other independent claims) were not clearly taught or fairly suggested by the prior art of record:

establish the A/V service between the superblock application and the resource via the network interface, wherein establishing is not otherwise supported by the superblock application;

provide video output for the A/V service to a video window within a display window associated with the superblock application, the video window displayable simultaneously with the display window of the superblock

application; and

provide audio for the A/V service.

Ex.1002, 18.

VII. Level of ordinary skill in the art

A Person of Ordinary Skill in The Art (“POSITA”) in July 2013 would have had a working knowledge of the software programming art that is pertinent to the ’092 Patent, including the use of software modules and application programming interfaces (APIs). A POSITA would have had a bachelor’s degree in computer science, computer engineering, or an equivalent, and three years of professional experience relating to computer programming. Lack of professional experience can be remedied by additional education, and vice versa. Ex.1003, ¶19.

VIII. Background

As of the July 2013 priority date of the ’092 patent, computer science and software programming had been developing as technical fields of study for at least forty years—and, some say, for several millennia. Ex.1009, 5, 185. Early computers were programmed using punch cards onto which numbers were encoded. The computer interpreted the numbers as instructions. Ex.1009, 227-229. Soon afterwards, the programmers began creating programs that would transform more human-friendly instructions into computer-executable punch cards—a process that came to be known as “compiling.” Ex.1009, 231 (describing the “famous IT compiler”). Ex.1003, ¶25.

Computers rapidly grew in their complexity, with new instructions and larger memories making it possible to tackle more sophisticated problems. Punch cards gave way to perforated tapes and then “teletype” terminals, which combined a keyboard for providing input to a computer with a display screen for displaying the computer’s output. Ex.1003, ¶26.

As computers grew in their sophistication, so too did programs expand in their complexity. By at least the 1980s, programmers could readily access “standard functions” provided as part of programming languages such as FORTRAN, COBOL, and Pascal. Ex.1010, 654. Even more extensive functions were available as “Program Libraries,” which would “typically [be] called as subroutines” from a program. Ex.1010, 654; Ex.1003, ¶27.

To facilitate the use of these additional “libraries,” the technology for “compiling” a program was split into a two-step process. In a first step, human-readable instructions in a single file were converted into computer-executable “object code.” Ex.1011, Abstract. In a second step, the various sources of instructions (both application source files and program libraries) were integrated into a complete stand-alone application. Ex.1011, 1:24-28; Ex.1012, 1:25-29. This process was known as “static linking.” Ex.1012, 1:29; Ex.1013, 1:52-54, 1:66-2:2; Ex.1014, 2:5-17. With static linking, the “static-link library has access to the client program code and data,” that is, “static-link libraries have access to the client

program execution context.” Ex.1015, 1:18-27, 1:39-40; Ex.1003, ¶28.

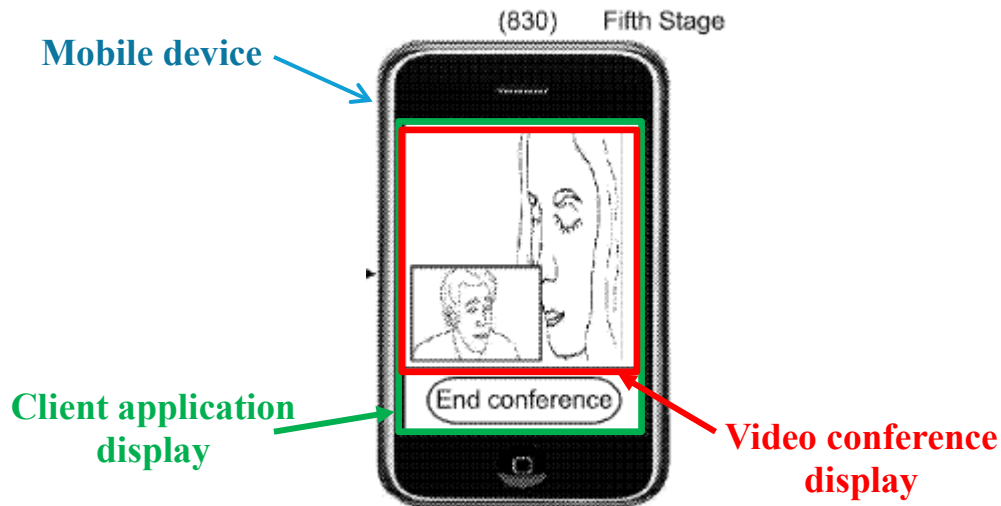
As software became even more complex, programmers recognized the value of allowing commonly used functions to be shared among multiple programs. Sharing allows multiple programs to call the same functions but avoids having each program incorporate its own redundant copy of the call functions. Technology for this came to be known as a “dynamic-link library. “In contrast to static-link libraries, several client programs can share a single copy of the code contained in a DLL, which saves memory space.” Ex.1015, 1:27-29. Just like static-link libraries, “DLLs are executed in the context of the calling code; that is, the DLL can access the code, data space, and the stack space of the client program that invoked a function contained in the DLL.” Ex.1015, 1:35-38. As a result, both static- and dynamic-linked libraries provide functions that can be executed without switching from the context of the calling program. These concepts were well-known in the computer programming art long before the ’092 patent’s earliest priority date. Ex.1016, 6:26 (“DLLs are a well-known feature...and are specific instances of APIs which are available as common resources to any application or other type of executable code”); Ex.1017, 16:33-34 (“Procedures to allow an application to incorporate new DLL's are well known in the art.”); Ex.1018, 6:65-66 (“Two forms of linking are common: static linking and dynamic linking.”); Ex.1003, ¶29.

Also well-known before the ’092 patent’s earliest priority date was a

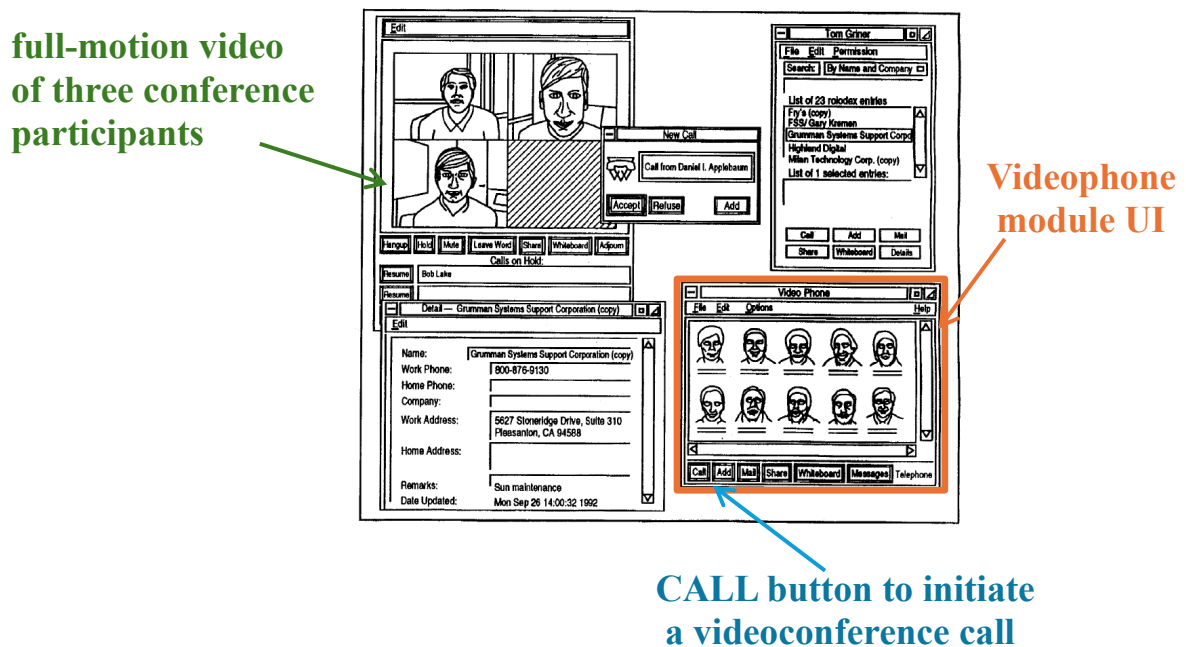
modular approach for providing the functions to the calling programs, where functionalities, such as video conferencing functionalities, were made accessible to calling programs in the form of self-contained components or modules. Ex.1008, 18:50-52 and FIG. 20 (“Software modules 161-168 communicate with operating system/GUI software 180 and other applications 170 utilizing standard function calls and interapplication protocols,” where the modules include “videophone 169,” “application sharing 166,” “computer-integrated telephony 167,” as shown in Figure 20); Ex.1049, [0054] and [0059]-[0060] (“a video call service device 602 comprises ... a network monitor module 614, a video call settings module 616, and a client management module 618,” where “the video call settings module 616 is invoked by the video call server 612” and “the client management module 618, when invoked, instructs a video call client 506 to reduce a display size used for a video call UI 104”); Ex.1019, Abstract (“a user [] select[s] a third party communication application from the user interface of the core application. The selected third party communication application is executed to enable the user to initiate a communication session with another user using the services of the selected third party communication application.”); *see also, generally* Ex.1005 and Ex.1006; Ex.1003, ¶30.

When the video conferencing functionalities were accessed from a calling application, the videos from the conference were displayed within or along with a

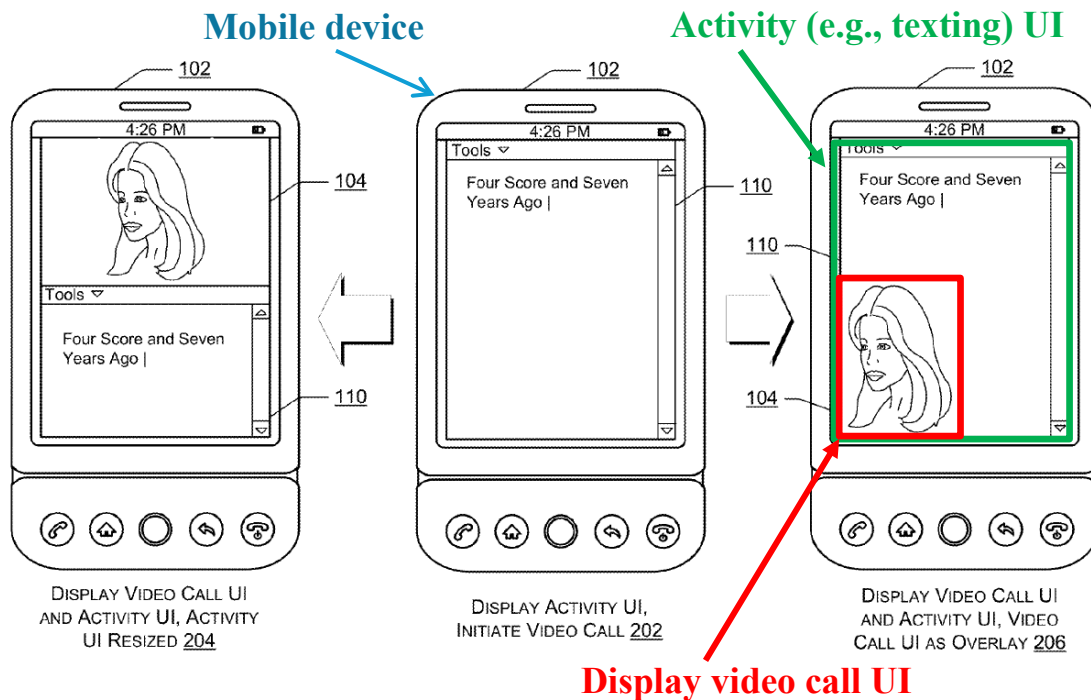
display of the calling application (or a display of another active application), as shown in the example figures below from before the earliest priority date of the '092 patent:



Ex.1005, Fig. 8 (partial and annotated).



Ex.1008, Fig. 2A (annotated).



Activity (e.g., texting) UI

Display video call UI

FIG. 2

Ex.20140184723, Fig. 2 (annotated).

Ex.1003, ¶31.

IX. Claim construction

Claim terms in IPR are construed according to their “ordinary and customary meaning” to those of skill in the art. 37 C.F.R. § 42.100(b). For the purposes of this proceeding and the grounds presented herein, Petitioner proposes that no terms require express construction. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017).

X. Relief requested and reasons therefore

Petitioner asks that the Board institute a trial for *inter partes* review and cancel the Challenged Claims in view of the analysis below.

XI. Identification of how the claims are unpatentable

A. Challenged claims

Petitioner challenges claims 1-30. Patent Owner has asserted claims 1-5, 7-8, 10, 12-19, 22-25, and 27 in the co-pending litigation. *See* Ex.1020, 2. A finding that the Challenged Claims (1-30) are unpatentable in this proceeding will resolve the parties' dispute in the co-pending litigation and obviate any need for a trial regarding the '092 patent, substantially reducing the time and expense of that litigation for all parties.

XII. Statutory grounds

Grounds	Claims	Basis
#1	1-5, 7-10, 12-20, and 22-29	35 U.S.C. § 103 over Abuan, Guzman, and Lawson
#2	6, 11, 21, and 30	35 U.S.C. § 103 over Abuan, Guzman, Lawson, and Ludwig

U.S. 2011/0249077 to Abuan et al. (Ex.1005, “**Abuan**”) published on October 13, 2011.

U.S. 2012/0092438 to Guzman Suarez et al. (Ex.1006, “**Guzman**”) published on April 19, 2012.

U.S. 2013/0019297 to Lawson et al. (Ex.1007, “**Lawson**”) published on January 17, 2013.

U.S. 5,689,641 to Ludwig et al. (Ex.1008, “**Ludwig**”) issued on November

18, 1997.

Abuan, Guzman, Lawson, and Ludwig are prior art under 35 U.S.C.

§ 102(a)(2).

Petitioner’s § 103 obviousness grounds rely on the combined teachings of the references and not on a physical incorporation of elements. *See In re Mouttet*, 686 F.3d 1322, 1332 (Fed. Cir. 2012).

Petitioner also cites below to additional prior art as evidence of the background knowledge of a POSITA and to provide contemporaneous context to support Petitioner’s assertions regarding what a POSITA would have understood from the prior art in the grounds. *See Yeda Research v. Mylan Pharm. Inc.*, 906 F.3d 1031, 1041-1042 (Fed. Cir. 2018) (affirming the use of “supporting evidence relied upon to support the challenge”); 37 C.F.R. § 42.104(b); *see also K/S HIMPP v. Hear-Wear Techs., LLC*, 751 F.3d 1362, 1365-66 (Fed. Cir. 2014); *Arendi S.A.R.L. v. Apple Inc.*, 832 F.3d 1355, 1363 (Fed. Cir. 2016).

A. Statement of material facts

1. Allowing unrestricted and unlimited use of a server-provided feature would create the undesirable risk of the server becoming overwhelmed and failing, either intentionally (in a denial-of-service attack) or unintentionally. Ex.1037, 19:5-10 (“denial of service attack (also, DoS attack) is an attack... that causes a loss of service to users, typically... by consuming the bandwidth of the victim

network or overloading the computational resources of the victim system”);

Ex.1038, 2:30-39 (“In denial of service (DOS) attacks, where... the server at the target IP address is overwhelmed by fake requests for service, and is either unable to service ‘real’ requests for service, or crashes due to overload”); Ex.1039, [0002] (“a denial of service attack, one or more attackers may send packets towards one or more servers, overwhelming the servers so that they are no longer able to serve their legitimate clients.”).

2. As of the priority date, it was common for a server to require a user login to access the server’s services. Ex.1043, Fig. 6 (showing a “typical login screen” asking for “an account name 293 and a password 295”); Ex.1021, [0024] (“A user login typically includes a user name and password.”); Ex.1022, [0028] (describing “operations typically performed” during login for “authentication and authorization”); Ex.1023, [0002] (describing the “well known” login process in which “a server computer is responsible for granting access to a remote user based on a valid combination of userID and password”).

3. A POSITA would have been familiar with voice-over-IP, which was a technique for exchanging audio communications in real time. Ex.1026, [0085] (“well-known Voice-over-IP standards that are cornerstone technology for the transmission of real-time audio”); Ex.1027, 6:39-47 (“devices which perform these functions [such as analog-to-digital (A/D) conversion, ... echo cancellation, and

other types of signal conditioning, as well as a voice activity detector (VAD) ...

function for determining the temporal boundaries of a telephone caller's speech]

are well-known in the art and are commercially available”); Ex.1028, [0026]

(“audio (such as voice-over-IP)”).

4. A POSITA would have been familiar with instant messaging, which was a technique for exchanging text-based messages in real time. Ex.1024, [0003] (“Instant messaging (“IM”) is a popular form of computer-based communication in which text-based messages are exchanged between users in real-time.”); Ex.1025, [0005] (a user “has access to a (text only) Instant Messaging client”).

5. The use of APIs to define how an application accesses functionality of a software module was well-known and commonly used. Ex.1029, [0087] (“APIs are well-known in the art”); Ex.1030, [0038] (“One of the primary purposes of an API is to provide a set of commonly-used functions”); Ex.1031, [0040] (“A well-known API”); Ex.1032, 5:32-48 (“An API is a well-known programming device.... [C]alls to API functions 60 are included in application program 56; the programmer is relieved of any task of having to incorporate or integrate the code itself of functions 60 in application program 56”).

6. A POSITA would have been familiar with device drivers for output devices and would have recognized that they commonly included APIs for reporting on an output device’s capabilities. Ex.1033, 6:30-32 (using “software

(e.g., the device drivers) to determine the capabilities” of hardware resources); Ex.1034, 11:8-26 (describing device drivers determining a “display device’s maximum resolution”); Ex.1035, Abstract (“determine capabilities of graphics hardware from a device driver”), [0004] (“provid[ing] a set of application programming interfaces (APIs) to ascertain device contexts and determine capabilities of graphics hardware from a device driver”); Ex.1036, [0033] (“device capabilities ... may be exposed through one or more APIs”).

B. Ground 1

1. Abuan

Abuan generally relates to the field of communications software for mobile devices, and more particularly, to real-time video conferencing from mobile devices. Ex.1005, [0003]-[0004]. Abuan describes a modular approach to facilitating such conferencing features, with functionality organized into components such as a video processing and encoding module, media exchange module, video processing module, video conferencing and processing module, and many others. Abuan, [0043], [0099], [0056]. The modules’ functionalities are accessed via “a set of application program interfaces (APIs).” Ex.1005, [0044]; Ex.1003, ¶¶44-48.

Abuan is analogous art to the ’092 patent. First, Abuan is in the same field of endeavor as the ’092 patent, namely, the “manner in which functionality is

accessed in certain environments, such as mobile device environments.” Ex.1001, 1:42-43; *see* Ex.1005, [0001]-[0002] (describing ways that “functionality” is accessed on “portable devices, such as smartphones”). Abuan and the ’092 patent also both generally relate to implementing functionality via software. *See* Ex.1001, Title (“providing additional functionality to existing software”); Ex.1005, [0006]-[0007], [0105] (describing how instant messaging or voice-over-IP applications “use the video conferencing functions of the video conference module”). Abuan and the ’092 patent both describe at length features for facilitating videoconferencing. Ex.1001, Abstract, 4:24-27 (purpose of a function block is to provide functions including “audio, video,... conferencing, meetings, and/or other functions”); Ex.1005, [0099] (describing a “video conference module... for performing a variety of video conferencing functions”). Ex.1003, ¶49.

Second, Abuan is reasonably pertinent to a problem allegedly addressed by the ’092 patent. The ’092 patent describes purported problems associated with using multiple applications on a mobile device, potentially requiring a user to “switch[] back and forth between the video window of [a] call and [a] superblock application.” Ex.1001, 3:65-4:2. The ’092 patent’s proposed solution is to provide functionality—such as communication functions—to a superblock application. Ex.1001, 4:19-37. Abuan similarly describes how communication functions are provided to a client application, which may be a “stand-alone application” or may

itself be further “integrated into another application.” Ex.1005, [0105], [0171];

Ex.1003, ¶50.

2. Guzman

Guzman describes a video-conferencing application with a semi-transparent overlay user interface superimposed on the live video display that centralizes account setup, authentication, contact management, and call initiation. Ex.1006, Abstract, [0002]–[0006], [0033], [0037], [0049]–[0051]. Through the overlay, a user can create an account or sign in to a service that validates credentials locally or via a remote server, after which the application retrieves and displays the user’s contacts for call placement. Ex.1006, [0044]–[0046], [0060]–[0063], [0074]–[0075], [0058], [0070]–[0071]. Guzman teaches an “auto-login” option that, once selected and validated, persists the user’s session by storing prior login information so subsequent launches bypass credential prompts. Ex.1006 [0056]–[0058]. The system thus streamlines repeated access to a server-backed conferencing service by enabling persistent login behavior that avoids future authentication interruptions. Ex.1006, [0056]–[0059]; Ex.1003, ¶52.

Guzman is analogous art to the ’092 patent. First, Guzman is in the same field of endeavor as the ’092 patent, namely, the “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:42-43; *see* Ex.1006, [0001]–[0002] (describing how “mobile phones[] allow users

to engage in video conferences”). Guzman and the ’092 patent also both generally relate to implementing functionality via software. *See* Ex.1001, Title (“providing additional functionality to existing software”); Ex.1006, [0221] (describing Guzman’s “Software Architecture” for “software running on a particular machine, such as... a handheld device”). Guzman and the ’092 patent both describe at length features for facilitating videoconferencing. Ex.1001, Abstract, 4:24-27 (purpose of a function block is to provide functions including “audio, video, conferencing, meetings, and/or other functions”); Ex.1006, Title (“Overlay for a video conferencing application”). Ex.1003, ¶53.

Second, Guzman is reasonably pertinent to a problem allegedly addressed by the ’092 patent, namely, the need to provide functionality—such as communication functions—to a superblock application. Ex.1001, 4:19-37. Guzman provides a modular approach to software design for providing functionality to an application, illustrating in Fig. 22 “the software architecture of a video conferencing application 2200.” Ex.1006, [0221]. The application, which “is a stand-alone application,” includes various modules, including “user interface (UI) interaction and generation module 2205, an account setup and Verification module 2210, a contact management module 2215, an overlay generator 2220, a conference management module 2225, and a video display generator 2230.” Ex.1006, [0221]-[0222]; Ex.1003, ¶54.

3. Reasons to combine Abuan and Guzman

A POSITA would have been motivated to combine the teachings of Abuan and Guzman for multiple reasons. Ex.1003, ¶102.

Abuan generally describes technology for facilitating videoconference communications. Ex.1005, Abstract. Abuan acknowledges that a videoconference between two parties requires the call recipient to accept the call. Ex.1005, [0117] (“turn the phone call into a video conference with the permission of the other party”). Abuan illustrates in Fig. 7 that initiating a video conference also requires the participation of a third intermediary device, termed a videoconference server. It would have been obvious to a POSITA for establishing a videoconference to also require the permission of the videoconference server, since the videoconference will also consume some of the limited resources available on the videoconference server. Ex.1003, ¶103. Allowing unrestricted and unlimited use of the videoconference feature would create the undesirable risk the videoconference server becoming overwhelmed and failing, either intentionally (in a denial-of-service attack) or unintentionally. Material Fact #1; Ex.1003, ¶103.

Guzman describes a “login procedure” in which a user desiring to access a service, such as a videoconference service, provides their username and password. Ex.1006, [0056]. To balance the need for verifying users’ login credentials with users’ desire to have quick and easy access to a server, Guzman further describes

an “auto-login” option that lets the user “skip the login procedure” after they successfully complete it the first time. Ex.1006, [0056]; Ex.1003, ¶104.

A POSITA would have found it obvious to incorporate Guzman’s login techniques, including the auto-login option, into Abuan’s videoconferencing clients and server. Their combination is simply the integration of prior art elements (login procedures for accessing server-based services such as videoconferencing) according to known methods (implemented using ordinary software programming techniques within the POSITA skill set) to yield predictable results (restricting Abuan’s videoconference services to logged-in users). Ex.1003, ¶105. The combination of Abuan and Guzman is merely the application of a known technique (requiring user logins) to a known system (Abuan’s videoconferencing architecture) ready for improvement (since Abuan does not discuss requiring login to yield predictable results (restricting Abuan’s videoconference services to logged-in users). Ex.1003, ¶105. In addition, the combination is also merely the use of known technique (requiring user logins) to improve similar devices (Abuan’s videoconference clients and server) in the same way (limiting access to logged-in users). Ex.1003, ¶105. Finally, Guzman teaches that having “an auto login option” in a “video conferencing application,” which corresponds to Abuan’s videoconferencing system, “allows the user to skip the login procedure (e.g., the entering of a user name and password) the next time the user opens the video

conferencing application.” Ex.1006, [0056]. A POSITA would have had a reasonable expectation of success because Guzman teaches an auto login option being used to allow a user to log into a video conferencing application. Ex.1006, [0056]-[0058]; Ex.1003, ¶106.

4. Lawson

Lawson describes a cloud-based communication platform that mediates incoming and outgoing voice, video, and messaging connections for client applications. Ex.1007, [0027]-[0031], [0042]-[0047], [0055]-[0057]. Lawson describes a layered security model in which a policy engine authenticates client communications using authorization tokens and enforces permissions before permitting connection establishment. Ex.1007, [0028], [0033]-[0035], [0038]-[0039], [0048]; Ex.1003, ¶¶56-57.

Lawson is analogous art to the ’092 patent. First, Lawson is in the same field of endeavor as the ’092 patent, namely, the “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:42-43; Ex.1007, [0046]-[0048] (describing a “functions to control the general functionality of the mobile device 14, including but not limited to its communication capabilities”). Lawson and the ’092 patent also both generally relate to implementing functionality via software. *See* Ex.1001, Title (“providing additional functionality to existing software”); Ex.1007, [0060] (“enable software

applications... on a mobile device to interface with a telephony or communication platform”). Lawson and the ’092 patent both describe features for facilitating audio and video communications. Ex.1001, Abstract, 3:19-31; Ex.1007, [0034], [0048] (“communication session that supports communications selected from audio, video, chat, and/or other suitable communication mediums”), [0057]; Ex.1003, ¶58.

Second, Lawson is reasonably pertinent to a problem allegedly addressed by the ’092 patent, namely, the need to provide functionality—such as communication functions—to a superblock application. Ex.1001, 4:19-37. Lawson provides a modular approach to software design for providing functionality to an application, illustrating in Fig. 17 that a mobile device “include[s] a series of modules relating to its communication function.” Ex.1007, [0046]. These modules include an “audio engine 42” used “to control a majority of voice communication functions, including interfacing with any device hardware 44 such as speakers and microphones.” Ex.1007, [0046]. The audio engine will also “initiate and control communication sessions including voice communication sessions using Internet protocol.” Ex.1007, [0046]. The mobile device further includes a “cloud interface module” that is “configured within a browser function of the mobile device.” Ex.1007, [0047]. The cloud interface module is used by “untrusted client applications to prove that they are authorized to perform particular actions.”

Ex.1007, [0048]; Ex.1003, ¶59.

5. Reasons to combine Abuan, Guzman, and Lawson

A POSITA would have been motivated to combine the teachings of Lawson with Abuan and Guzman for multiple reasons. Ex.1003, ¶86.

Abuan and Guzman generally describe client applications for facilitating videoconference communications. Ex.1005, Abstract; Ex.1006, Abstract. Lawson describes techniques “for communicating with a client application.” Ex.1007, Abstract. Lawson describes the benefits of employing “authentication and/or authorization security measures that function to secure the communication channels,” such as thwarting attacks from “malicious parties.” Ex.1007, [0028]. Lawson’s techniques are imminently “adapted to function for any suitable type of communication,” including “video calls.” Ex.1007, [0028]; Ex.1003, ¶87.

A POSITA would have found it obvious to incorporate Lawson’s authentication and authorization security measures in the videoconferencing system of the Abuan-Guzman combination. Their combination with Lawson is simply the integration of prior art elements (e.g., username-and-password authentication, authorization checks, and videoconferencing) according to known methods (implemented using ordinary software programming techniques within the POSITA skill set) to yield predictable results (restricting access to Abuan-Guzman’s videoconferencing system to authorized users). Ex.1003, ¶88. Further,

the combination is merely the use of known techniques (e.g., logging into a server) to improve similar devices (Abuan-Guzman's videoconference clients and server) in the same way ("prevent[ing] malicious parties from posing" as someone else). Ex.1007, [0028]; Ex.1003, ¶88.

A POSITA would also have been motivated to combine Abuan with Lawson because Lawson suggests "includ[ing] one or more security ... management tools" for a communication session that supports communications selected from audio, video ... and/or other suitable communication mediums," which corresponds to Abuan's "real-time communication session that involves the transmission of the captured video images[, i.e.] a video conference." Ex.1007, [0048]; Ex.1005, [0034]. Lawson provides the implementation details of the security management tools, such as including an authorization token in communication requests "to authorize actions" such as "mak[ing] and/or receiv[ing] calls." Ex.1007, [0048]. A POSITA would have had a reasonable expectation of success because Lawson teaches that the "authorization security measures [] function to secure the communication channels." Ex.1007, [0028]; Ex.1003, ¶89.

Additional reasons to combine are provided in the detailed analysis below.

6. Independent claim 1

[1.0] A computer program product in a non-transitory computer readable medium for use in a data processing system:

Abuan describes a "video conferencing and processing module 1300 of a

dual camera mobile device” that is “implemented as software [*“a computer program product”*] processes that are specified as a set of instructions recorded on *a computer readable storage medium.*” Ex.1005, [0170], [0183], [0233]. Abuan also teaches that “these instructions [of the software processes] are executed by one or more processing unit(s) [*“for use in a data processing system”*]” of the mobile device. Ex.1005, [0233], [0249] (“a mobile computing device can include one or more processing units”); Ex.1003, ¶¶60-63.

[1.1] [a data processing system] that comprises a processor coupled to a network interface,

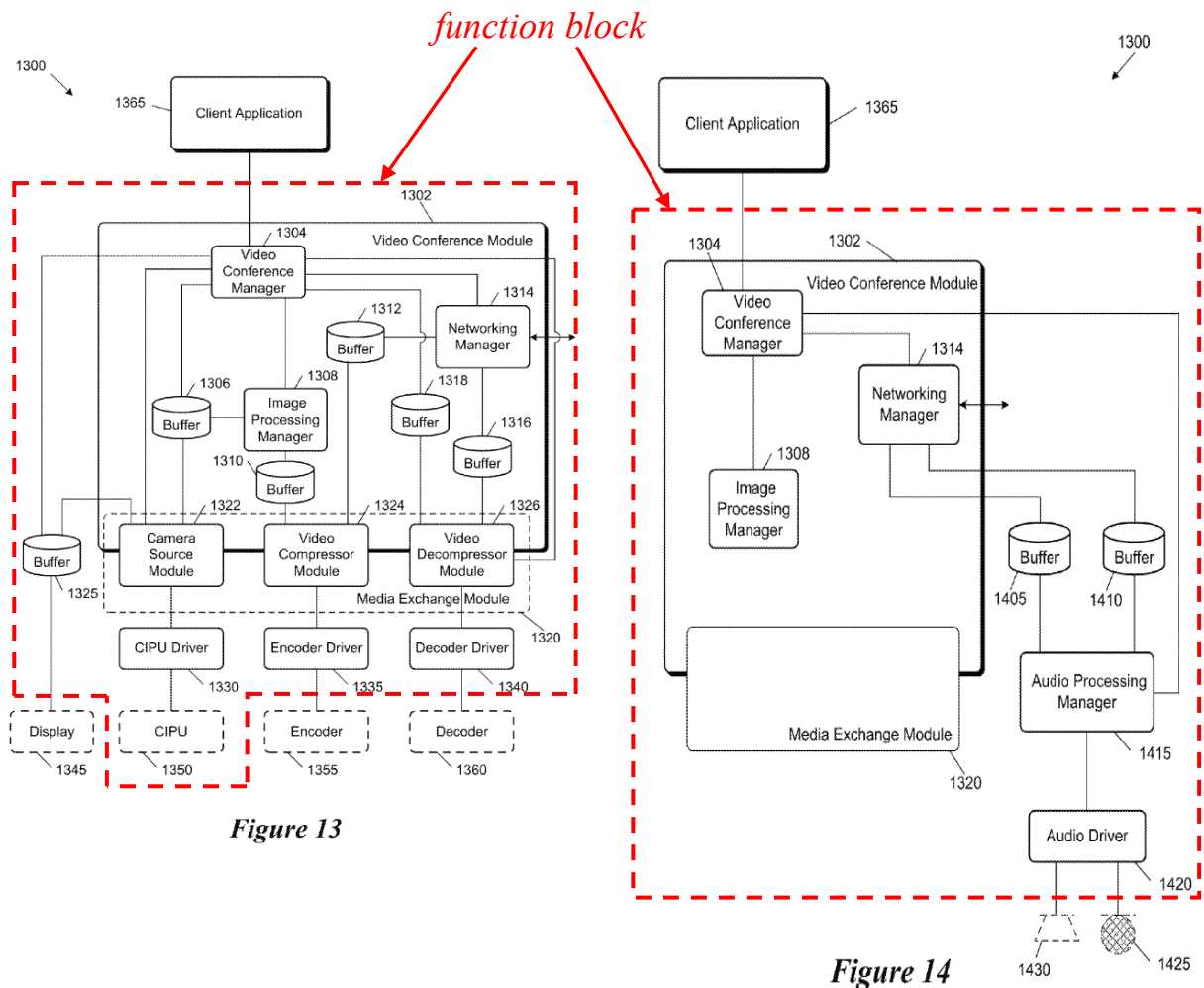
Abuan discloses that its mobile device “has *a network interface.*” Ex.1005, [0003]. A POSITA would have found it obvious that the network interface was implemented as the “wireless communication subsystem(s) 1925.” Ex.1005, [250]. Ex.1003, ¶66. Abuan describes the one or more processing unit(s) discussed at [1.0] as “one or more processors, cores of processors, or other processing units.” Ex.1005, [0233]. Abuan also discloses that the processing units are coupled to peripherals including a “wireless communication subsystem(s) 1925.” Ex.1005, [0249]-[0250], Fig. 19; Ex.1003, ¶¶64-69.

[1.2] the computer program product holding computer program instructions that comprise a function block,

As discussed at [1.0], Abuan discloses the video conferencing and processing module 1300, implemented as a software “that [is] specified as a set of

instructions,” teaching “the computer program product holding computer program instructions.” Ex.1005, [0233]; Ex.1003, ¶70.

“function block”: The various modules and components (i.e., the video conferencing and processing module 1300 excluding the client application 1365 and hardware devices such as display 1345) of Abuan’s mobile device correspond to the claimed “function block.” See, e.g., Ex.1005, [0056], [0099], [0170], [0183]. The function block is annotated in Abuan’s Figures 13 and 14 below. Ex.1003, ¶71.



Ex.1005, Figs. 13 and 14 (annotated); Ex.1003, ¶71.

[1.3] the function block having audio/video (A/V) call functionality, the function block operative when executed by the processor to:

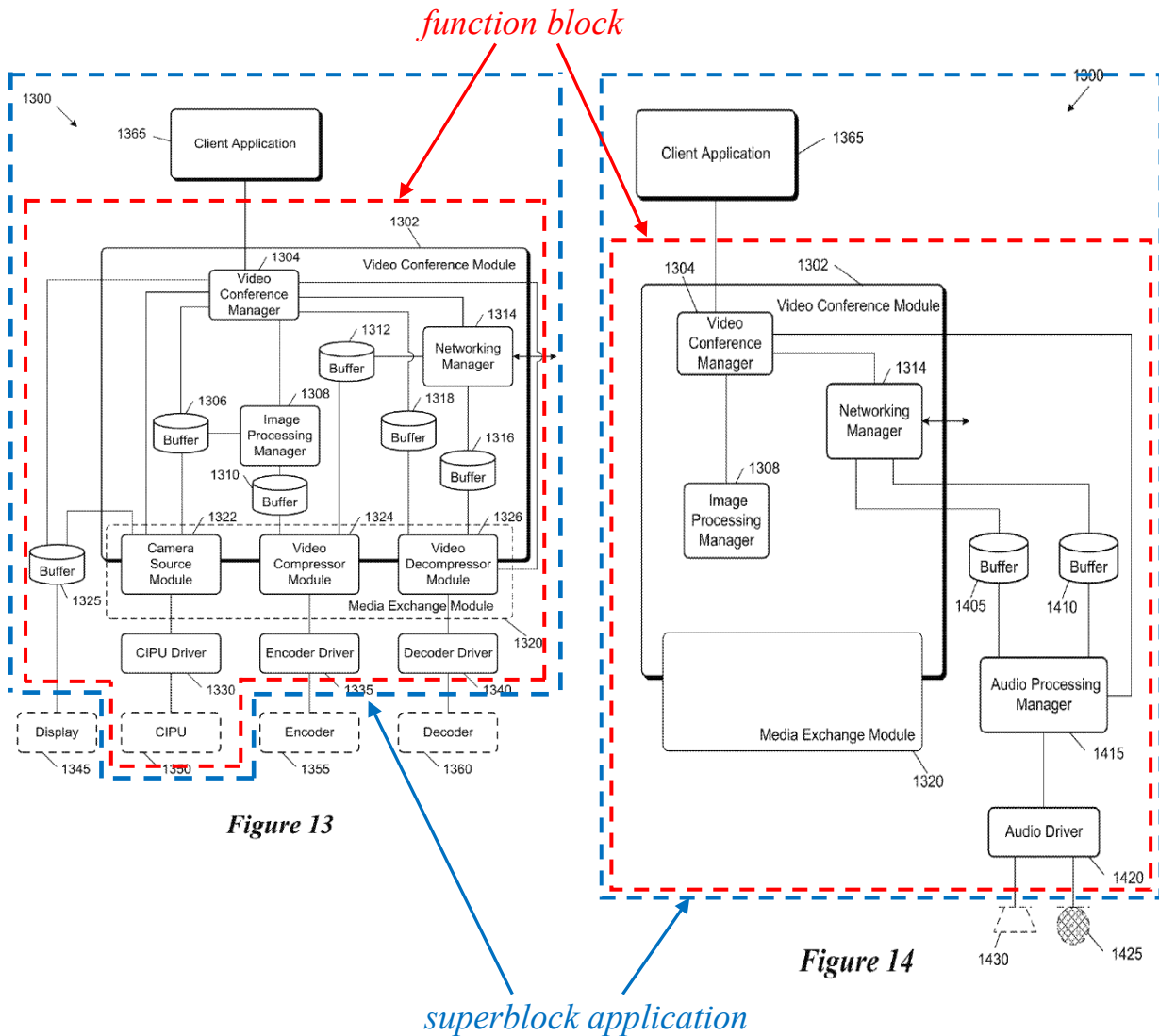
Abuan describes the video conference module 1302 (part of the “*function block*”) as providing “video conferencing functions” to the client application 1365, such as starting a video conference (“*audio/video (A/V) call functionality*”) when instructed by the client application. Ex.1005, [0171], [0186]-[0187], FIG. 15. A POSITA would have found it obvious that the video conference includes video and audio. Ex.1005, [0106], [0138], [0149], [0153] (“mobile devices in a video conference communicate data (e.g., audio and video images)”, [0185] (“multi-participant audio conference”); Ex.1003, ¶¶72-74.

As discussed at [1.0], the video conferencing and processing module, which includes “*the function block*,” is implemented as a set of instructions that, when executed by processors, cause the processors to perform the actions indicated in the instructions, teaching “*the function block operative when executed by the processor.*” Ex.1003, ¶¶72-75.

[1.4] receive from a superblock application a request for an audio/video (A/V) service;

“*superblock application*”: As discussed at [1.2], Abuan describes a video conferencing and processing module 1300 (including the client application 1365 and the various modules), corresponding to the claimed “*a superblock application.*” The *superblock application* (and the “*function block*”) are annotated

in Abuan's Figures 13 and 14 below. Ex.1003, ¶76.



Ex.1005, Figs. 13 and 14 (annotated); Ex.1003, ¶76.

As discussed at [1.3], Abuan describes the video conference module 1302 receiving instructions from (“*receive from ... a request*”) the client application (part of “*the superblock application*”) to start a video conference, i.e., an audio/video conference call (“*for an audio/video (A/V) service*”). Ex.1005, [0172], [0186]-[0187]; Ex.1003, ¶77.

[1.5] determine that the superbloc application is authorized to access the A/V service functionality;

Abuan illustrates in Figure 7 a “video conference request messaging sequence 700” that initiates a videoconference. When a video conference client (e.g., client application 1365) sends a request to start an audio/video conference call, as discussed at [1.4], a “Video Conference Request” message is sent to a video conference server 715. Ex.1005, [0108]-[0109]; Fig. 7. Thus, Abuan’s Fig. 7 process assumes that a user is already logged into, and authorized to use, the video conference server. Ex.1003, ¶80.

Like Abuan, Lawson describes a cloud-based communication system to support communication sessions between an initiating client device and a destination. Ex.1007, Abstract. The communication session can include “voice/audio, video, screen sharing,” or other suitable media. Ex.1007, [0057]; Ex.1003, ¶81.

Lawson describes using “security... management tools,” such as “an authorization token, which can be defined as an electronic listing of possible privileges for its bearer.” Ex.1007, [0048]. For example, authorization tokens are used to control “a user’s abilities to make and/or receive calls” or to request “communications selected from audio, video, chat, and/or other suitable communication mediums.” Ex.1007, [0048]; Ex.1003, ¶¶82-83.

Lawson explains that an authorization token is “verified” by checking its

value at a “communication platform,” such as “servers and/or services configured for trafficking and connection communications between [] users.” Ex.1007, [0051]; Ex.1003, ¶84.

A POSITA would have found it obvious to incorporate Lawson’s authorization-checking concepts into Abuan’s system for multiple reasons, including to prevent unauthorized users from wasting resources and bandwidth of the videoconference server. Ex.1003, ¶¶86-89.

In the combination, Lawson’s authorization token would be used to authorize Abuan’s client application (and the user of that application) to “make and/or receive [videoconference] calls.” Ex.1007, [0048]. Abuan’s client application would include the authorization token with its instructions to the videoconference module to start a videoconference. The videoconference module would then verify the received authorization token by communicating with the videoconference server (which would incorporate the verification techniques Lawson describes for a communication platform such as a server). This server-supported verification process would be used to determine whether the client application is authorized to “use[] the video conferencing functions of the video conference module 1302.” Ex.1005, [0171]. In this way, Abuan’s videoconference module would determine whether the client application (“*determine that the superblock application*”) may use the videoconference module’s services to start

an audio/video conference call (“*is authorized to access the A/V service functionality*”). Ex.1003, ¶¶78-89.

[1.6] determine that the AN service requires access to a resource that is external to the data processing system;¹

Abuan illustrates in Figure 7 a “video conference request messaging sequence 700” that initiates a videoconference. When a video conference client (e.g., client application 1365) sends a request to start a videoconference, a “Video Conference Request” message is sent to a video conference server 715. Ex.1005, [0108]-[0109]; Fig. 7. As discussed at [1.4], the client application 1365 “sends instructions to the video conference module 1302 such as instructions to start a conference and end a conference.” Ex.1005, [0172]. It would have been obvious for the video conference server 715 to be a separate device from the “client device 705” running the video conference client application, as shown in Figure 7. Ex.1005, Fig. 7; Ex.1003, ¶92. Thus, video conference server 715 is “*a resource that is external to*” the mobile device (which “include[s the] one or more processing units”) (“*the data processing system*”). Ex.1005, [0249]. Ex.1003, ¶¶92-93.

¹ Petitioner notes that there is no earlier introduction of a “AN service” in claim 1. For the purposes of this petition, Petitioner treats the term “the AN service” as referring to “the A/V service” recited in [1.4].

As noted at [1.5], Abuan's Fig. 7 process assumes that a user is already logged into, and authorized to utilize a video conference module that communicates with, the video conference server. It would have been obvious to a POSITA, however, for the video conference server to require users to login to use the server's videoconferencing capabilities, because requiring a user to login was nearly universally employed by servers at the time. Material Fact #2; Ex.1003, ¶94. For example, the first step of Lawson's communication process is "receiving an executed login of a user." Ex.1007, [0061]. Consistent with that understanding and with Abuan's general architecture, Guzman describes a videoconferencing application for providing videoconference communications between two or more clients. Ex.1006, Abstract, [0034]; Ex.1003, ¶94.

Guzman's video conferencing application 2200 (analogous to Abuan's video conferencing and processing module 1300, which includes *the function block*, as shown at [1.4]) includes "an account setup and verification module 2210" that facilitates a user logging into the videoconferencing server. Ex.1006, [0222], [0230]. Guzman's login process allows the user to "enable auto-login," and thus "skip the login procedure (e.g., the entering of a user name and password) the next time the user opens the video conferencing application." Ex.1006, [0056]. Guzman also explains that the user's login credentials (username and password) are verified either "through information stored on the device with the video conferencing

application, or by accessing a remote server that stores login information for numerous users at different locations.” Ex.1006, [0057]. Like Abuan’s videoconference server, Guzman’s remote server is an “*external resource*.” Ex.1003, ¶¶95-97, 101.

It would have been obvious to a POSITA that using a client application to start a videoconference would include the client application sending instructions to a video conference module 1302 and the video conference module 1302 (part of “*the function block*”) determining whether the user’s credentials may be verified locally or if, instead, the user needs to log into the remote server (whether “*the AN service requires access to a resource that is external to the data processing system*”). Ex.1003, ¶97. Notably, the claim language does not specify how to determine whether “*access to a resource*” is required; thus, Guzman’s disclosure that user credentials are verified either locally or via contacting a remote server renders the claim limitation obvious because Guzman’s video conferencing application would obviously *determine[]* whether to verify the credentials locally or remotely. For example, where the user had previously logged in and selected the “auto-login” option, accessing the videoconferencing server to log in would not be necessary; otherwise, accessing the videoconferencing server to log in would be necessary. Ex.1003, ¶98.

As another example, it would have been obvious for the video conferencing

application to confirm that the user has provided username and password values through the user interface. *See* Ex.1006, [0074]. Where the user failed to provide a password value, for example, it would be obvious for the video conferencing application to determine that contacting the remote server was unnecessary to confirm that the provided password (i.e., no password at all) was invalid. Ex.1003, ¶99.

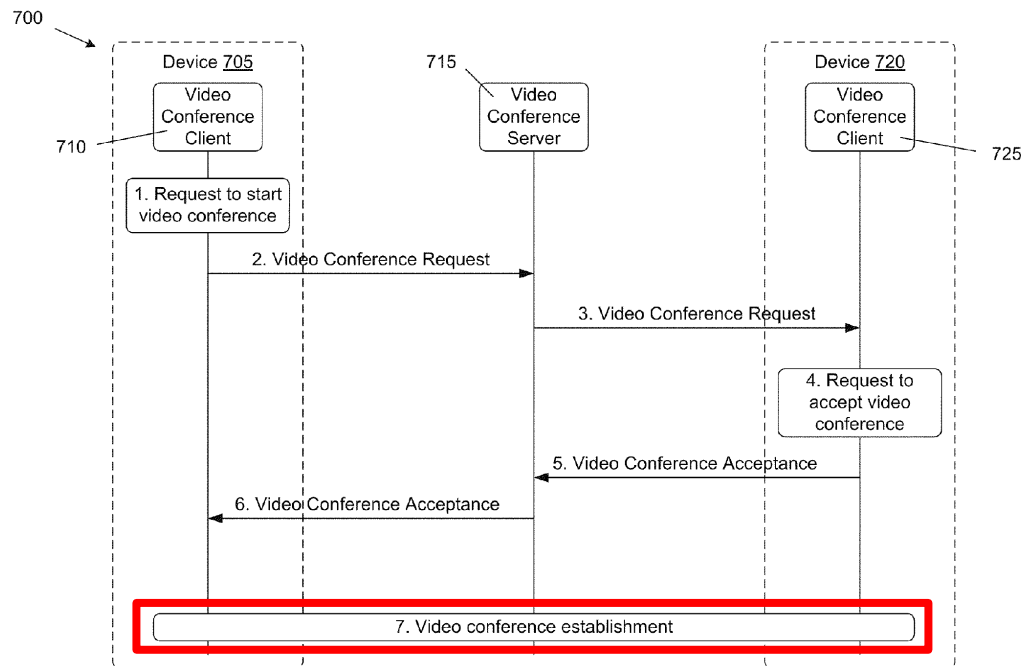
It would have been obvious for the video conference server 715, which “routes messages among video conference clients,” to be a centralized server that also “stores login information” associated with the video conferencing application. Ex.1005, [0109]; Ex.1006, [0057]. For example, Guzman contemplates “a centralized server ... associated with the video conferencing application” and in the context of Abuan’s videoconference system shown in Fig. 7, the videoconference server is the only server. Ex.1006, [0089]. Guzman’s remote server storing login information also provides other services, such as maintaining a user’s address book with the user’s contacts. Ex.1006, [0069] (“The address book may be stored locally or on a remote server (e.g., the same server... used to verify login information).”). Further, Guzman expressly contemplates that the functionality of various modules—such as the “account setup and verification module”—may be integrated into other modules or components. Ex.1006, [0235]; Ex.1003, ¶100.

As such, it would have been obvious to integrate Guzman's remote-server login functionality into Abuan's videoconference server for several reasons, including because, in the context of Abuan's videoconference system shown in Figure 7, the videoconference server is the only server. Ex.1003, ¶¶100, 102-106. Thus, under the combined teachings of Abuan and Guzman, when a user's client application sent instructions to a video conference module 1302 to start a video conference and the video conference module 1302 determined that it was necessary to verify the user's credentials with a remote server, the video conference module 1302 (part of "*the function block*") would send the credentials to videoconference server 715 for verification. Ex.1003, ¶¶90-106.

[1.7.1] establish the A/V service between the superbloc application and the resource via the network interface,

Abuan describes initiating videoconferencing services ("*establish the A/V service*"), for example, by exchanging call setup messages with a videoconference server ("*the resource*"). Ex.1005, [0108]-[0109], Fig. 7. For example, a "network layer 640... **establishes a network connection** (not shown) between the dual camera mobile device and a remote device of a video conference." Ex.1005, [0104]. Abuan illustrates in Fig. 7 the messages exchanged to initiate a videoconference from a videoconference client 710 to videoconference client 725 via videoconference server 715. These videoconference clients are the same as client application 1365 discussed at [1.4]. *See* Ex.1005, [0108], [0171]. The

process culminates in “Video conference establishment” across all three systems:



Ex.1005, Fig. 7 (annotated).

Ex.1003, ¶¶107-108.

As discussed at [1.4], Abuan’s client application uses services provided by components of *the function block*, such as video conference module 1302 and its submodules, such as video conference manager 1304. Ex.1005, [0186]-[0195]. For example, Abuan explains that the client application “uses the video conferencing functions of the video conference module 1302.” Ex.1005, [0171]. Within video conference module 1302, a networking manager 1314 “establishes the connections between the dual camera mobile device and the other device of the video conference at the start of the video conference.” Ex.1005, [0179]. It would have been obvious for the networking manager 1314 of the video conference module

1302 to establish such network connections using a network interface like network interface 650. Ex.1003, ¶112; *see* Ex.1005, [0174] & [0102] (“The video conference module 1302 is the same as the video conference module 625,” which “includes ... a network layer 640”), [0104] (“the network layer 640 performs ... all of the networking functionalities for video conferencing”). In the connection establishment process of Fig. 7, the videoconference client device 705 “communicates with the video conference server 715 over a network (e.g., a cellular network, a local area network, a wireless network, a network of networks, the Internet etc.) **through a network interface** such as the network interface 650.” Ex.1005, [0109]; *see also id.* [0053], [0101], [0106]. It would have been obvious to a POSITA from Abuan’s description that networking manager 1314 and network layer 640 (part of *the function block*) are responsible for handling the network communications aspect of “*establish[ing] the A/V service*,” such as sending and receiving request and acceptance messages via the network interface. Ex.1003, ¶¶109-112.

Thus, Abuan describes how the various modules comprising *the function block* create videoconferencing connections (“*establish the A/V service*”) from the client application (*superblock application*) to the videoconference server (“*the resource*”) using a *network interface*. Ex.1003, ¶¶107-112.

[1.7.2] wherein establishing is not otherwise supported by the superblock application;

As discussed at [1.4], the *superblock application* includes the *function block* and the client application 1365.

Abuan explains that “client application 1365 [part of “*the superblock application*”]... uses the video conferencing functions of the video conference module 1302 [part of “*the function block*”],” such as the functionality for *establishing the A/V service*. Ex.1005, [0171], *supra*, [1.7.1]. It would have been obvious to a POSITA for client application 1365 to lack corresponding functionality within itself for several reasons. Ex.1003, ¶¶113-123. First, that understanding would be consistent with the well-known principle of modular application design where “subprograms [are] compiled separately. A set of relocatable object modules can be linked together and loaded for execution by a linking loader.” Ex.1040, 518-519; Ex.1003, ¶116.

Further, Abuan identifies an example of client application 1365 as “a voice-over-IP (VoIP) application (e.g., Skype), or an instant messaging application.” Ex.1005, [0171]. A POSITA would have been familiar with voice-over-IP, which was a technique for exchanging audio communications in real time. Material Fact #3; Ex.1003, ¶¶114-115. A POSITA would also have been familiar with instant messaging, which was a technique for exchanging text-based messages in real time. Material Fact #4; Ex.1003, ¶¶114-115. Thus, a POSITA would not expect a voice-over-IP or instant messaging application to provide functionality for

videoconferencing, which involves video communication techniques absent from audio- and text-based communication applications. In another example, Abuan explains that client application 1365 is a phone application or a contacts application, which a POSITA would have understood not to support videoconferencing functionality. Ex.1005, [0117]-[0119], [0141]-[0148]; Ex.1003, ¶¶117-120.

Second, Abuan describes how an application accesses the functionality of the other software modules via defined application programming interfaces (APIs). Ex.1005, [0235]-[0237]. This structure puts functionality in the API-implementing component (e.g., *the function block*) and not in the API-calling component (e.g., *the superblock application*):

An API allows a developer of an API-calling component (which may be a third party developer) to **leverage specified features provided by an API-implementing component**.

Ex.1005, [0239]; Ex.1003, ¶121.

It will be appreciated that the **API-implementing component 1810 may include additional functions, methods, classes, data structures, and/or other features that are not specified through the API 1820 and are not available to the API-calling component 1830.**

Ex.1005, [0247]; Ex.1003, ¶121.

The use of APIs to define standard, programmatic ways for an application to access functionality of a software module was well-known and commonly used. Material Fact #5; Ex.1003, ¶122. Thus, a POSITA would have found it obvious for Abuan’s client application to access functionality of various software modules (including videoconferencing module 1302) that was *not supported* in the client application itself. Ex.1003, ¶122.

Third, a POSITA would have found it obvious for client application 1365 not to support the functionality provided by the video conference module 1302—or the other “*function block*” modules identified in [1.4]—because that would render the modules superfluous and unnecessary. Ex.1003, ¶123; *see* Ex.1032, 5:44-68. If the client application 1365 already included full support for videoconferencing, then a POSITA would have found it nonsensical for client application 1365 to use other software components, such as video conference module 1302, to perform tasks that the client application 1365 could already do for itself. Ex.1003, ¶123. Maintaining multiple redundant codebases for performing the same functionality would have been seen as wasteful and potentially problematic. Ex.1003, ¶123. For these reasons, a POSITA would have found it obvious for client application 1365 not to provide functionality for videoconferencing (except through its use of the videoconference module and other *function block* modules). Ex.1003, ¶123.

Petitioner’s analysis is consistent with the ’092 patent specification, which

describes function block 200 (part of superblock 104) as providing functions or capabilities to the part of the superblock 104 that lacks such functions or capabilities, i.e., to the part of the superblock 104 that excludes the function block 200. Ex.1001, 4:19-27, 5:3-4; Ex.1003, ¶124.

[1.8.1] provide video output for the A/V service to a video window within a display window associated with the superblock application

As discussed at [1.4], Abuan's video conference manager 1304 is part of "*the function block*." Abuan illustrates in Fig. 15 a videoconferencing process performed by videoconference manager 1304. Ex.1005, [0186]. The process includes "generat[ing] (at 1520) composite images for displaying on the dual camera mobile device (i.e., local display). These composite images may include ... various combinations of [video] images from the cameras of the local dual camera mobile device and [video] images from cameras of the other device participating in the video conference." Ex.1005, [0190], [0034] ("video images [in] a video conference"). The composite image may be a "PIP display," meaning a picture-in-picture display. Ex.1005, [0036]; Ex.1006, [0132]; Ex.1003, ¶126.

Abuan illustrates in Fig. 8 a user interface provided to a user via the client application running on a mobile device. During the videoconference, both parties' videos are displayed on the mobile device using the "picture-in-picture" concept:

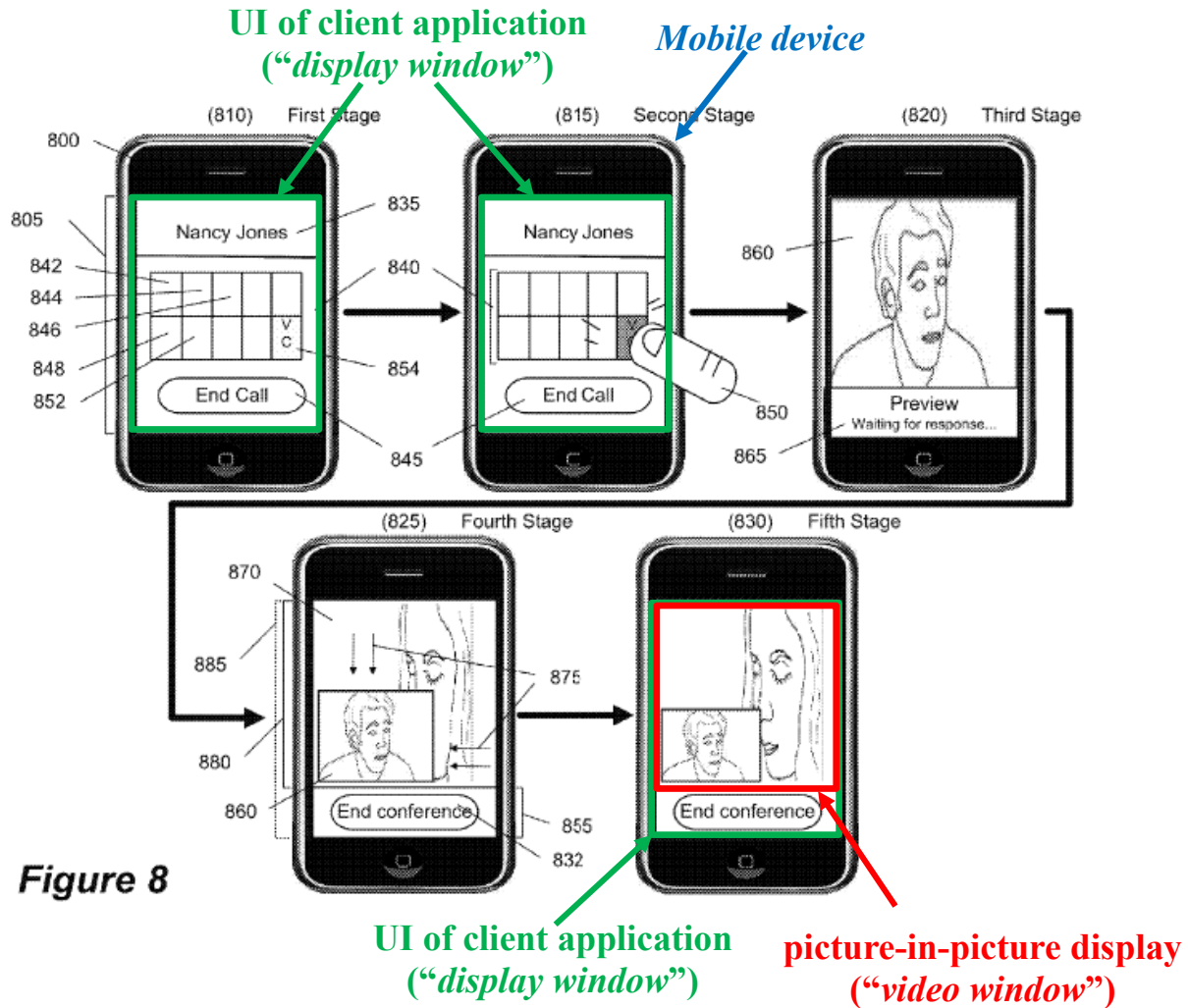


Figure 8

ABUAN, Fig. 8

Ex.1003, ¶127.

The picture-in-picture display of the videos from the mobile devices in the videoconference is a “video window.” The entire display area or UI of the client application is a “display window.” The user interface shown in Fig. 8 corresponds to the client application, such as client application 1365 (“superblock application”). Ex.1005, [0108], [0110], [0112], [0117]-[0129], [0171]-[0172];

Ex.1003, ¶¶128-130.

In summary, Abuan describes how video conference manager 1304 generates video output of a picture-in-picture display of videos from mobile devices in a videoconference for display within a window of the client application running on one of the mobile devices, which renders this limitation obvious. Ex.1003, ¶¶125-130.

[1.8.2] the video window displayable simultaneously with the display window of the superblock application; and

See [1.8.1]. Abuan shows in Fig. 8 that the user's local video ("*video window*") is shown simultaneously with other parts of the client application user interface, including the remote user's video ("*simultaneously with the display window of the superblock application*"). Ex.1003, ¶¶131-132.

[1.9] provide audio for the A/V service.

As discussed at [1.4], Abuan's various modules shown in Figures 13 and 14 correspond to the claimed "*function block*." Abuan discloses that the modules include an "audio processing manager 1415 [that] processes audio data captured by the dual camera mobile device" and that "retrieves audio data from the buffer 1410 and decodes the audio data, which is then output through the audio driver 1420 to the speaker 1430." Ex.1005, [0184]. The "speaker 1430" would obviously be a hardware component of the mobile device. For example, Abuan illustrates in Fig. 19 an example dual camera mobile computing device architecture 1900" having "audio subsystems 1930 [that] is coupled to a speaker." Ex.1005, [0249], [0251].

The decoded and outputted audio is audio “received from the other device,” that is, an audio portion of the videoconference (“*audio for the A/V service*”). Ex.1005, [0189]. Ex.1003, ¶134.

In summary, Abuan discloses that audio processing manager (part of “*the function block*”) that decodes and outputs audio to a speaker on the mobile device, which renders this limitation obvious. Ex.1003, ¶¶133-135.

[2.0] The computer program product of claim 1 wherein the function block is further operative when executed by the processor to

See [1.0] and [1.3]. Ex.1003, ¶136.

[2.1] operate as a bridge between the superblock application and the external resource.

As discussed at [1.7.1], Abuan describes the client application sending instructions to the video conference module to start an audio/video conference call, and the networking manager of the video conference module performing the communications between the client applications on the caller and callee devices (e.g., video conference clients 710 on device 705 and 725 on device 720 in Figure 7 of Abuan discussed at [1.7.1]) and the video conference server to establish the audio/video conference call between the devices. Ex.1005, [0171]-[0172]. The networking manager is also responsible for facilitating communications between the devices during the audio/video conference call, routing and/or exchanging images and audio data between the devices (via the video conference server, as

discussed at [1.7.1]). Ex.1005, [0180], [0185]. Thus, the networking manager (part of “*the function block*”) of the video conference module *operate[s] as a bridge between* the client application (part of “*the superblock application*”) on each device and the video conference server (“*the external resource*”). Ex.1003, ¶¶137-140.

[3.0] The computer program product of claim 1 wherein

See [1.0]. Ex.1003, ¶141.

[3.1] the request is an application programming interface (API) call from the superblock application to the function block.

As discussed at [1.4], Abuan’s instruction to start or end a conference are each an example of a “*request for an audio/video (A/V) service.*” As discussed at [1.7.2], Abuan contemplates using an application programming interface (API) between a calling component (e.g., Abuan’s client application, which is a part of a *superblock application*) and an implementing component (e.g., Abuan’s videoconferencing module, which is part of a *function block*). Ex.1003, ¶¶142-143.

Abuan further explains that an API call is used “to initiate performance of an operation.” Ex.1005, [0243]. Starting or ending a videoconference is an operation, so it would have been obvious to a POSITA for Abuan’s client application to initiate such operations by calling one or more functions or methods defined in the API of the videoconferencing module. Ex.1003, ¶¶143-144.

[4.0] The computer program product of claim 1 wherein

See [1.0]. Ex.1003, ¶145.

[4.1] the superblock is configured to initialize the function block when the superblock application is loaded and executed.

Abuan describes a process 1500 that begins with “receiving (at 1505) instructions to start a video conference.” Ex.1005, [0187]. For example, the video conference may be started by the user interacting with the user interface of the client application (“*superblock application*”) shown in Figures 8-10. *See, e.g.*, Ex.1005, [0110]. The user’s interaction, in turn, causes the client application to start initializing the modules required for a videoconference. Ex.1005, [0187] (“instructions to start a video conference... are received... from a user through a user interface... and forwarded to the video conference manager 1304 by the client application 1365”). Ex.1003, ¶¶146-149.

Abuan describes initializing the videoconference manager 1304 and other modules in response to the request to start a videoconference:

Next, the process 1500 **initializes (at 1510) a first module** that interacts with the video conference manager 1304. The modules of some embodiments that interact with the video conference manager 1304 include the CIPU 1350, the image processing manager 1308, the audio processing manager 1415, and the networking manager 1314.

Ex.1005, [0188]; *see also id.* [0189]-[0190] (describing how initialization continues until “all of the required modules have been initialized”). Ex.1003, ¶149.

Since the module initialization occurs in response to user's interaction with the client application user interface, it would have been obvious to a POSITA for initialization to occur when the client application "*is loaded and executed.*" For example, Abuan describes how an application is "read into memory for processing by a processor." Ex.1005, [0234]; *see also id.* [0233] ("applications are implemented as software processes that are specified as a set of instructions... executed by one or more processing unit(s)"). Ex.1003, ¶¶146, 150.

[5.0] The computer program product of claim 1 wherein the function block is further operative when executed by the processor to

See [1.0] and [1.3]. Ex.1003, ¶151.

[5.1] determine that the superblock application is authorized to access functionality provided by the function block for the A/V service.

As discussed at [1.5], the combination of Abuan and Lawson renders it obvious that the video conference module 1302 (part of the "*function block*") *determine[s]* that the client application 1365 (part of the "*the superblock application*") is authorized to use the video conferencing functions of the video conference module 1302 such as making and/or receiving audio/video conference calls. Ex.1005, [0171]; Ex.1003, ¶152.

7. Independent claim 7

[7.0] A computer program product in a non-transitory computer readable medium for use in a data processing system

See [1.0]. Ex.1003, ¶153.

[7.1] [a data processing system] that comprises a processor coupled to a network interface,

See [1.1]. Ex.1003, ¶154.

[7.2] the computer program product holding computer program instructions that comprise a function block,

See [1.2]. Ex.1003, ¶155.

[7.3] the function block having audio/video (A/V) call functionality, the function block operative when executed by the processor to:

See [1.3]. Ex.1003, ¶156.

[7.4] receive from a superblock application a request for a communication session,

See [1.4]. Ex.1003, ¶157.

[7.5] wherein the function block is configured to provide functionality for the communication session to the superblock application;

As discussed at [1.4] and [1.7.1], Abuan describes the client application sending instructions to the video conference module to start an audio/video conference call. In response, the various component modules of the video conference module together (“*the function block*”) provide audio, video, and/or conferencing functionalities to the client application. For example, the networking manager handles the network communications aspect of establishing the audio/video conference call. Ex.1005, [0208], [0211]. The media exchange module provides access to “various video processing and encoding functionalities” while the audio processing manager 1415 processes audio data. Ex.1005, [0237], [0184].

The video conference manager 1304 controls the operation of the audio/video conference call, including initializing required modules and ceasing the operation. Ex.1005, [0175]. The image processing manager processes and encodes images of the audio/video conference call. Ex.1005, [0178]. Ex.1003, ¶¶158-160.

[7.6] notify the superblock application that the communication session is being requested;

Abuan discloses that when the video conference module receives instructions from the client application on a mobile device to start an audio/video conference call, as discussed at [1.4], a video conference tool on the client application is activated to establish the audio/video conference call, and a hold stage or wait state indicator is displayed on the user interface of the client application 1365 (part of “*the superblock application*”) to indicate the activation and that the mobile device is waiting for a response from the other mobile device (to accept or reject the audio/video conference call). Ex.1005, [0108]-[0115]; Ex.1003, ¶161.

For example, Figure 8 of Abuan shown below shows the user interfaces of the client application displayed on a device 705 during the establishment of the audio/video conference call between device 705 and another device 720. Ex.1005, [0110]. The user interfaces belong to the client application on device 705. Ex.1003, ¶162. The user interfaces display a “smaller version of the video captured by the device,” “a still image that is stored on the device,” “a full screen display of a

video being captured by the device,” “a ‘Preview’ notation at the bottom of this video,” etc., to indicate that the audio/video conference call is in the process of being established. Ex.1003, ¶¶162-163.

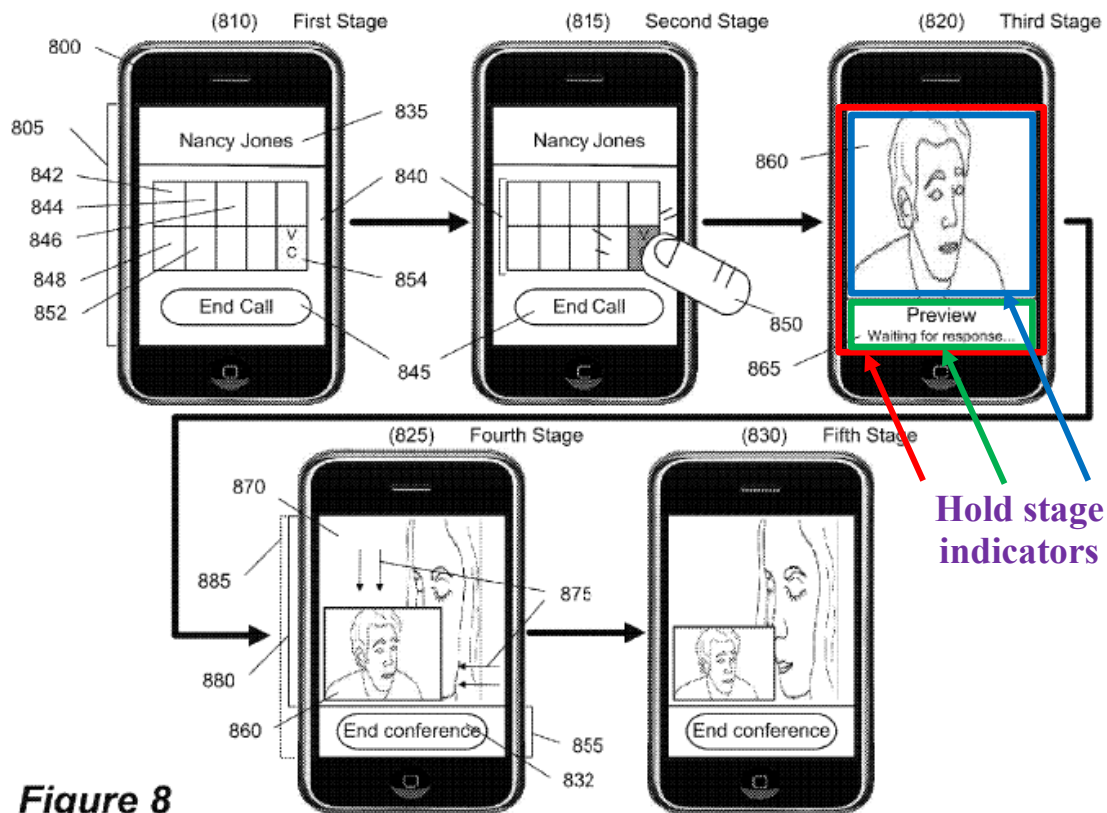


Figure 8

Ex.1005, Fig. 8 (annotated); Ex.1003, ¶162.

A POSITA would have found it obvious that *the function block*, in response to receiving the instructions from *the superblock application*, notif[ied] *the superblock application* about the activation of the video conference tool (“*that the communication session is being requested*”) and the wait for the establishment of the video conference call such that *the superblock application* displayed the hold stage indicator (e.g., as shown in Figure 8 above (third stage)). Ex.1003, ¶165.

First, because the video conference manager 1304 communicates with the client application to handle the client application's requests to establish video conferences as well as to cause the client application to update its user interfaces, the video conference manager 1304 (part of "*the function block*") would have been an obvious way to notify the client application (part of "*the superblock application*") about the status of the video conference request from the client application (i.e., about the wait for the establishment of the video conference call) so that the client application can update its user interface accordingly (e.g., by transitioning from second stage 815 to third stage 820). Ex.1003, ¶¶165-167.

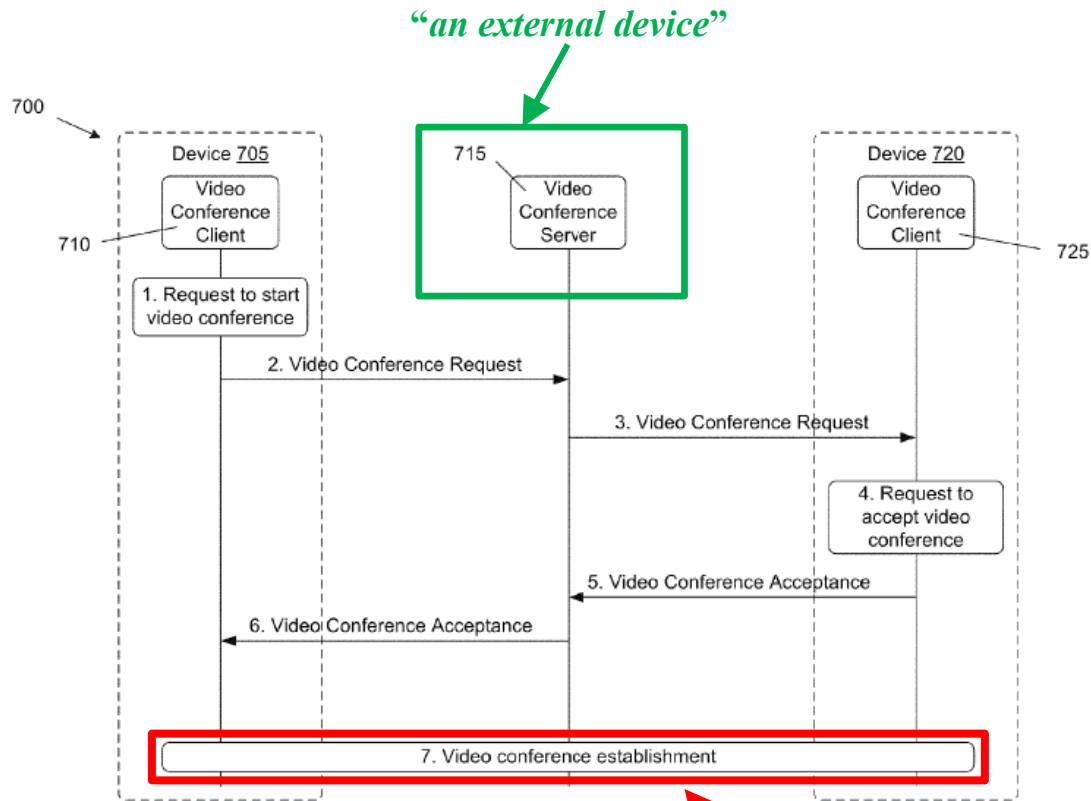
Second, a user interface of a client application displaying a notification when a module, triggered by an event, informed the client application about the event was a routine and expected design in event-driven systems—common in mobile video conferencing. Ex.1003, ¶168. For example, a POSITA would have understood that the transition from second stage 815 to third stage 820 was triggered by an event, such as the user's selection of the Video Conference option 854 on the user interface of the client application 1365. Ex.1005, [0120] ("The third stage 820 displays the UI 805 after... the selection of the Video Conference option 854."). In response, a functional module, such as the video conference manager, informed the client application about the event, causing the client application to update its user interface to display a notification about the event, the

hold stage indicator shown in Figure 8 above. Ex.1005, [0121]-[0123]. A POSITA would have understood that in event-driven systems like Abuan (*see* Ex.1005, [0140], [0243]), modules signal state transitions to the user interface via callbacks, queues, or observer notifications to inform users of backend. Ex.1041, 137 (defining “a callback” as “[a] function written as part of the application, associated with a specific widget resource, that is invoked as a result of a specific change of state associated with that widget). Ex.1003, ¶¶168-169, 164.

[7.7] establish a connection with an external device via the network interface to support the communication session; and

As discussed at [1.7.1], Abuan describes how the various modules comprising *the function block* on a mobile device use the *network interface* of that device to create videoconferencing connections between (“*establish a connection with*”) the client application on that mobile device and the videoconference server (“*an external device*”) to establish an audio/video conference call between the client application and another client application on another mobile device (“*to support the communication session*”), as shown in Figure 7 of Abuan below.

Ex.1003, ¶¶170-171.

**Figure 7***“a connection with an external device”***Ex.1005, Fig. 7 (annotated) Ex.1003, ¶170.****[7.8] provide the communication session to the superblock application,**

See [7.7]. Ex.1003, ¶172.

[7.9] wherein the steps of receiving, notifying, establishing, and providing are performed without switching context from the superblock application to another application on the data processing system.

Abuan’s client application is a “stand-alone application” that employs application programming interfaces (APIs) to invoke functionality provided by the video conference module and other modules of *the function block*. Ex.1005, [0105], [0171]; *also see generally*, Ex.1005, [0236]-[0248]. Abuan does not

reference making any context switch as part of the API-calling process, nor does it show any context switch in its Figures 8-11, which show the client application continuing to run with the user interfaces open on the device while function calls (e.g., to request a video conference via the server 715) are performed throughout various stages of video conferencing establishment. *See generally*, Abuan, [0118]-[0128]. In fact, Abuan describes some embodiments in which function calls (e.g., accepting video conference requests) are performed without any user input and without any action by the client application's user interface. *See, e.g.*, Abuan, [0187], [0115]; Ex.1003, ¶¶173-177.

Petitioner's analysis is consistent with the '092 patent specification, which describes "switch[ing] contexts" as "switch[ing] between applications." Ex.1001, 2:66-67; *see also* 3:37-46, 3:62-4:2; Ex.1003, ¶178.

[8.0] The computer program product of claim 7 wherein the function block is further operative when executed by the processor to

See [7.0] and [7.3]. Ex.1003, ¶179.

[8.1] determine whether the function block has a capability to perform any rendering needed by the communication session.

As discussed at [1.7.2], Abuan explains how a client application interacts with various software modules via application programming interface (API) calls. "[A]n API call can report to an application the capabilities or state of a hardware device, including those related to aspects such as input capabilities and state,

output capabilities and state, processing capability, power state, storage capacity and state, communications capability, etc.” Ex.1005, [0243]. Abuan explains that the client hardware device includes common output devices such as “speaker 1430” and “a display.” Ex.1005, [0184], [0252]. It would have been obvious to a POSITA for Abuan’s various software modules (which correspond to “*the function block*”) to determine their own output capabilities based at least in part on the output capabilities of output devices such as the speaker and display. Ex.1003, ¶¶180-183.

For example, Abuan describes example “API-implementing component[s]” such as “device driver[s].” Ex.1005, [0246]. A POSITA would have been familiar with device drivers for output devices and would have recognized that they commonly included APIs for reporting on an output device’s capabilities. Material Fact #6; Ex.1003, ¶¶184-185.

The determination by Abuan’s software modules of the output capabilities of a client device renders obvious this limitation. Ex.1003, ¶¶180-185; *see* Ex.1001, 9:29-57 (describing checking for “an available speaker output” or “whether there is an available screen”). Ex.1003, ¶186.

Petitioner’s analysis is consistent with the ’092 patent specification, which describes the function block “checking ... to see whether there is an available microphone (mic) input and an available speaker output” and “checking ... to see

whether there is an available screen” for displaying the video. Ex.1001, 9:30-57.

These checks relate to “*determin[ing] whether the function block has a capability to perform any rendering*” of audio or video. Ex.1001, 9:37-40 & 9:50-53;

Ex.1003, ¶186.

[9.0] The computer program product of claim 7 wherein the function block is further operative when executed by the processor to

See [7.0] and [7.3]. Ex.1003, ¶187.

[9.1] determine whether the superblock has a capability to perform any rendering needed by the communication session.

As discussed at [8.1], Abuan describes determining the output capabilities of a client device. It would have been obvious to a POSITA that where a client device lacks a speaker, neither the *function block* nor the *superblock application* would have a capability of rendering the audio portion of a videoconference. Ex.1003, ¶188. Similarly, where a client device lacks a display, neither the *function block* nor the *superblock application* would have a capability of rendering the video portion of a videoconference. Thus, Abuan’s discussion of determining the output capabilities of a client device renders this limitation obvious. Ex.1003, ¶¶188-189.

[10.0] The computer program product of claim 7 wherein the function block is further operative when executed by the processor to

See [7.0] and [7.3]. Ex.1003, ¶190.

[10.1] determine that the superblock application is authorized to access functionality provided by the function block for the communication session.

See [1.5]. Ex.1003, ¶191.

8. Independent claim 12

Claim 12 recites the same functionality as claim 7 in the form of a method claim. As discussed above for claim 1, Abuan describes how a video conference module 1302 and related other modules (“*function block*”) perform a variety of “video conferencing functions,” including “video processing and encoding functionalities” and “audio processing operations.” Ex.1005, [0171], [0237], [0183]-[0184]. These functions and operations render obvious perform “*a method*.” Ex.1003, ¶¶192-201.

Further, as discussed at [1.2] and [1.1], the conference module 1302 and related other modules are modules of Abuan’s mobile device that includes a network interface, teaching “*a function block associated with a computing entity having a network interface*.” The mobile device includes the one or more processing unit(s) (“*a data processing system*”), i.e., the *computing entity* recited in claim 12 includes *the data processing system* recited in claim 7. Thus, claim 12 is obvious for the same reasons discussed above for computer program product claim 7. Ex.1003, ¶¶192-201.

Dependent claims 16, 17, and 19 recite the same functionality as claims 8-10, respectively. Accordingly, claims 16, 17, and 19 are obvious for the same reasons discussed above for computer program product claims 8-10, respectively.

Ex.1003, ¶¶211-214, 217-218.

[13.0] The method of claim 12 wherein providing the communication session includes

See [12.0] and [12.5]. Ex.1003, ¶202.

[13.1] operating as a bridge between the superblock application and the external device.

See [2.1]. Ex.1003, ¶203.

[14.0] The method of claim 12 wherein

See [12.0]. Ex.1003, ¶204.

[14.1] the request is an application programming interface (API) call from the superblock application to the function block.

See [3.1]. Ex.1003, ¶205.

[15.0] The method of claim 12 further comprising

See [12.0]. Ex.1003, ¶206.

[15.1] determining whether the function block has a capability to provide the communication session.

Abuan at least renders obvious limitation [15.1] because Abuan renders obvious the media transport manager 1735 determining whether the buffer 1304 has images that the video conference manager 1304 can retrieve to generate PIP or other composite views for an audio/video conference call. Ex.1003, ¶207.

Abuan discloses that “the media transport manager 1735” of “the networking manager 1700,” which is a component of the video conference module 1302 (part of “*the function block*”), “may [] monitor other variables of the device

such as power consumption and thermal levels that may affect how the operational power modes of the cameras are configured.” Ex.1005, [0227], [0209]. The power consumption level affects the operational power mode configuration of the cameras “of the dual camera mobile device” because “how much power the camera is allowed to consume” “[d]etermin[es] the operational mode in which to leave the unused camera.” Ex.1005, [0054], [0061]. In other words, the power consumption level of a camera of a mobile device determines whether the camera is, for example, in “the first operational power mode, [i.e.,] the camera is powered off,” or whether it is “in the fourth operational power mode,” i.e., “the camera is powered on, the camera's sensor is configured, and” “the camera is now sending images.” Ex.1005, [0058]; Ex.1003, ¶208.

The images “captured by a camera of a device” are “image[s] of a video (i.e., ... a sequence of images)” “that are to be transmitted to another device participating in the video conference.” Ex.1005, [0198]-[0199]. The images are “processed by CIPU 1350” and stored in the buffer 1306. Ex.1005, [0176], [0199]. “[T]he video conference manager 1304 ... retrieves [the] images processed by CIPU 1350 (i.e., images captured by the dual camera mobile device) from the buffer 1306” and “scales and composites the images before displaying the images on the dual camera mobile device,” for example, as “PIP or other composite views.” Ex.1005, [0176]; Ex.1003, ¶209.

Accordingly, Abuan discloses that the media transport manager 1735 (part of “*the function block*”) monitors the power consumption level of a camera of the mobile device, which determines whether the camera is powered off and as such is not capturing images, or is capturing and sending images to the buffer 1304 (part of “*the function block*”), which the video conference manager 1304 (part of “*the function block*”) then retrieves to generate PIP or other composite views for an audio/video conference call. Thus, a POSITA would have found it obvious that the media transport manager 1735 (part of “*the function block*”) determines whether the buffer 1304 (part of “*the function block*”) has images that the video conference manager 1304 (part of “*the function block*”) can retrieve to generate PIP or other composite views for an audio/video conference call (“*the communication session*”), rendering obvious “*determining whether the function block has a capability to provide the communication session.*” Ex.1003, ¶210.

[18.0] The method of claim 12 further comprising

See [12.0]. Ex.1003, ¶215.

[18.1] initializing the function block when the superblock application is loaded and executed.

See [4.1]. Ex.1003, ¶216.

[20.0] The method of claim 12 further comprising

See [12.0]. Ex.1003, ¶219.

[20.1] providing at least one configuration parameter for the function block,

[20.2] wherein the configuration parameter enables the function block to be configured after the function block is distributed as part of the superblock application.

As discussed at [1.7.2] and [3.1], Abuan's client application interacts with the videoconference module and other software modules via application programming interfaces (APIs). "[A]n API-calling component ... passes data and control information using parameters via the API calls or invocations." Ex.1005, [0241]. Abuan's disclosure of passing parameters from an API-calling component (such as the client application) to an API-implementing component (such as the videoconference module) renders obvious "*providing at least one configuration parameter for the function block.*" Ex.1003, ¶¶220-223.

Abuan describes various forms of *configuration parameter[s]*. For example, Abuan's user "may select a profile setting for configuring a camera" to use "high resolution video," "low resolution video," "still images," or various power levels. Ex.1005, [0068], [0058]. Abuan explains that the camera is controlled via a "sensor module" or "camera source module," either of which is part of *the function block*. Ex.1005, [0057], [0180]; *see supra*, [1.2]. The camera profile settings may be selected based on a running application: "[I]f the user selects a video conferencing application, a profile that configures both cameras to capture video is automatically selected." Ex.1005, [0068]. Thus, it would have been obvious to a POSITA for the sensor module or camera source module to receive from the client application, as a

configuration parameter, information about how the client application intends to use the camera. Ex.1003, ¶224. This configuration step would be performed as part of running the client application, showing that the camera’s configuration is “*configured after the function block is distributed as part of the superblock application.*” Ex.1003, ¶224; *see also supra*, [1.4] and [7.9] (the client application is a “stand-alone application” and thus incorporates the *function block* modules). Ex.1003, ¶¶224-226.

9. Independent claim 22

Claim 22 recites substantially the same functionality as claim 1 in the form of a method claim. As discussed above for claim 1, Abuan describes how a video conference module 1302 and related other modules (“*function block*”) perform a variety of “video conferencing functions,” including “video processing and encoding functionalities” and “audio processing operations.” Ex.1005, [0171], [0237], [0183]-[0184]. These functions and operations render obvious perform “*a method.*” Ex.1003, ¶¶227-235.

Further, as discussed at [1.2] and [1.1], the conference module 1302 and related other modules are modules of Abuan’s mobile device that includes a network interface, teaching “*a function block associated with a computing entity having a network interface.*” The mobile device includes the one or more processing unit(s) (“*a data processing system*”), i.e., the *computing entity* recited in

claim 22 includes *the data processing system* recited in claim 1. Thus, claim 22 is obvious for the same reasons discussed above for computer program product claim 1. Ex.1003, ¶¶227-235.

Dependent claims 23-25 recite the same functionality as claims 2, 4, and 5, respectively. Accordingly, claims 23-25 are obvious for the same reasons discussed above for computer program product claims 2, 4, and 5. Ex.1003, ¶¶236-241.

[26.0] The method of claim 22 further comprising

See [22.0]. Ex.1003, ¶242.

[26.1] providing at least one configuration parameter for the function block,

See [20.1]. Ex.1003, ¶243.

[26.2] wherein the configuration parameter enables the function block to be configured after the function block is distributed as part of the superblock application.

See [20.2]. Ex.1003, ¶244.

[27.0] The method of claim 22 further comprising

See [22.0]. Ex.1003, ¶245.

[27.1] determining whether the function block has a capability to perform any rendering needed by the A/V service.

As discussed at [8.1], it would have been obvious to a POSITA to *determine whether the function block has a capability to perform any rendering needed by the* communication session. As discussed at [7.4], the communication session is taught by an audio/video conference call, which also teaches *the A/V service* (as discussed

at [1.4]). Thus, a POSITA would have found limitation [27.1] obvious. Ex.1003, ¶246.

[28.0] The method of claim 22 further comprising

See [22.0]. Ex.1003, ¶247.

[28.1] determining whether the superblock has a capability to perform any rendering needed by the communication session.²

See [9.1]. Ex.1003, ¶248.

[29.0] The method of claim 22 further comprising

See [22.0]. Ex.1003, ¶249.

[29.1] providing at least one configuration parameter for the function block,

See [20.1]. Ex.1003, ¶250.

[29.2] wherein the configuration parameter enables the function block to be configured after the function block is distributed as part of the superblock application.

See [20.2]. Ex.1003, ¶251.

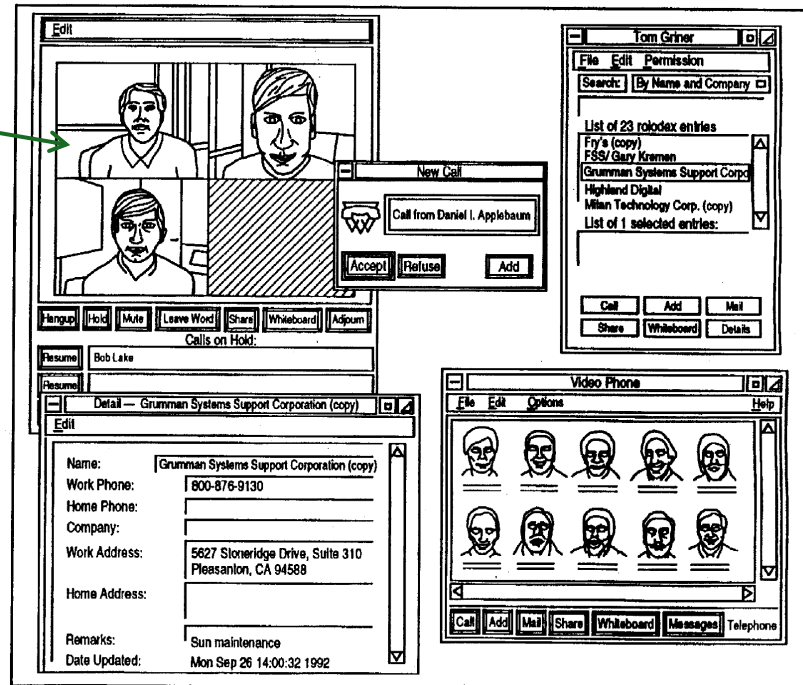
² Petitioner notes that there is no earlier introduction of a “communication session” in claim 28 or independent claim 22 on which it depends. For the purposes of this petition, Petitioner assumes that the term “the communication session” recited here is supposed to refer to “a communication session.”

C. Ground 2

1. Ludwig

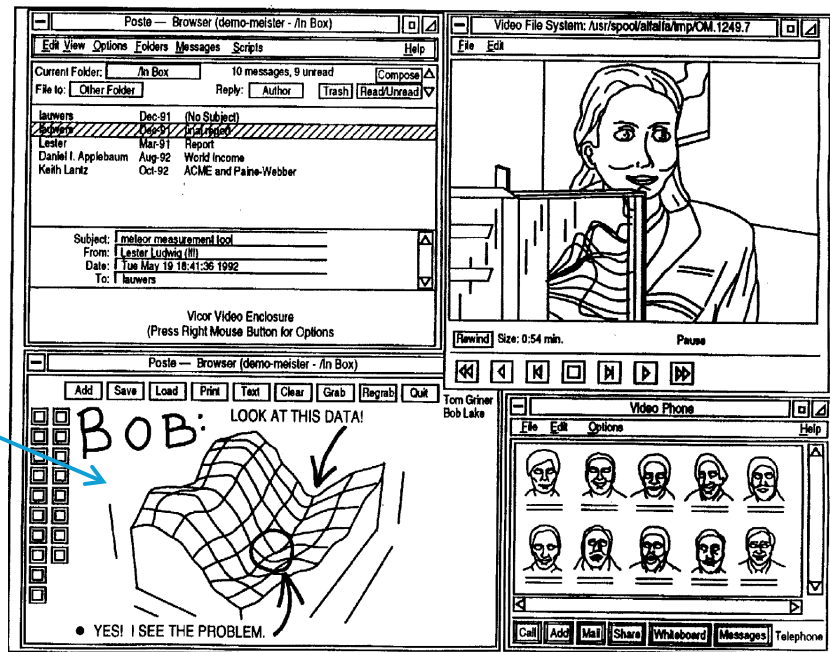
Ludwig describes a multimedia collaboration system that integrates real-time audio/video teleconferencing with data conferencing to replicate face-to-face meetings across distributed networks and heterogeneous platforms. Ex.1008, Abstract. Beyond high-quality, multi-party video and audio, the system emphasizes advanced collaboration tools, including “snapshot” or screen sharing of windows or regions, application sharing with shared control, and a shared whiteboard that supports telepointing and rich, multi-user annotations in real time. Ex.1008, 2:57-3:12. Ludwig illustrates exemplary mid-meeting screens in Figures 2A and 2B, including “full-motion video of three conference participants” and “data shared and annotated by those conferees.” Ex.1008, 6:53-57; Ex.1003, ¶253.

full-motion video of
three conference
participants



Ex.1008, Fig. 2A (annotated); Ex.1003, ¶253.

Data shared and
annotated by
conference
participants



Ex.1008, Fig. 2B (annotated); Ex.1003, ¶253.

Ludwig is analogous art to the '092 patent. First, Ludwig is in the same field of endeavor as the '092 patent, namely, the “manner in which functionality is

accessed in certain environments, such as mobile device environments.” Ex.1001, 1:42-43; *see* Ex.1008, 38:56-39:32 (describing a videoconference participant using a “laptop in the field in Central Mexico” equipped with “two wireless modems”).³ Ludwig and the ’092 patent also both generally relate to implementing functionality via software. *See* Ex.1001, Title (“providing additional functionality to existing software”); Ex.1008, Fig. 20, 18:34-20:37 (describing “workstation software” and “server software” for multimedia collaboration). Ludwig and the ’092 patent both describe at length features for facilitating videoconferencing. Ex.1001, Abstract, 4:19-31 (purpose of a function block is to provide functions including “audio, video,... conferencing, meetings, and/or other functions”); Ex.1008, 2:66-3:7 (“As used herein, desktop teleconferencing includes real-time audio and/or video teleconferencing...”). Ex.1003, ¶254.

Second, Ludwig is reasonably pertinent to a problem allegedly addressed by the ’092 patent. The ’092 patent describes purported problems associated with using multiple applications on a mobile device, potentially requiring a user to “switch[] back and forth between the video window of [a] call and [a] superblock

³ Those of skill in the art in the 1990s (when Ludwig was filed and published) considered a laptop to be a mobile device. *See* Ex.1001, 3:2-3, 7-9; Ex.1045, 1:37 (“mobile devices, such as notebook and laptop computers”); Ex.1046, 4:19-24.

application.” Ex.1001, 3:65-4:2. The ’092 patent’s proposed solution is to provide functionality—such as communication functions—to a superblock application.

Ex.1001, 4:19-37. Ludwig similarly describes how communication functions are centrally coordinated and provided via a “Collaboration Initiator”: “All

collaborative functions can be accessed through this module.” Ex.1008, 18:53-55.

Ludwig illustrates in Fig. 20 that the Collaboration Initiator is integrated with other data conferencing modules, such as videophone 169 and application sharing 166.

The Collaboration Initiator and its integrated modules are shown separated from “other applications 170,” indicating that Ludwig’s modules provide their

communication functions to the Collaboration Initiator directly (not via the

“interapplication protocols” used to communicate with other applications 170).

Ex.1008, 18:44-52; Ex.1003, ¶255.

2. Reasons to combine Abuan with Ludwig

A POSITA would have been motivated to combine the teachings of Ludwig with Abuan for multiple reasons. Ex.1003, ¶¶266-269.

Abuan generally describes technology for facilitating videoconference communications and suggests the possibility of also sharing “other types of content.” Ex.1005, Abstract, [0037]. Ludwig explains that traditional videoconferences are “limited in the richness of the data that can be exchanged.”

Ex.1008, 2:9-10. To address these limitations, Ludwig describes a “multimedia

collaboration system that greatly facilitates distributed collaboration, in part by replicating the benefits of face-to-face collaboration.” Ex.1008, 2:59-62. Ludwig’s improved system adds features not found in traditional videoconferencing, such as “sharing of ‘snapshots’ of selected regions of the user’s screen,” “application sharing,” and a “shared whiteboard.” Ex.1008, 3:1-6. The benefits of Ludwig’s multimedia collaboration system are described with a detailed scenario involving how users at a variety of remote locations access an “in-house corporate ‘expert’” regarding trading of financial instruments. Ex.1008, 35:35-41:37. Notably, all the “collaborative activities in this scenario [take] place during a relatively short portion of the expert’s day (e.g., less than an hour of cumulative time) while the Expert remained in his office.” Ex.1008, 41:27-30; Ex.1003, ¶267.

A POSITA would have found it obvious to incorporate Ludwig’s multimedia collaboration features to the videoconferencing system of Abuan. The combination with Ludwig is simply the integration of prior art elements (e.g., videoconferencing, screen sharing, collaborative whiteboarding) according to known methods (implemented using ordinary software programming techniques within the POSITA skill set) to yield predictable results (adding screen-sharing and whiteboarding features to Abuan’s videoconferencing clients). Ex.1003, ¶268. Alternatively, the combination is merely the use of known techniques (e.g., screen sharing, whiteboarding) to improve similar devices (Abuan’s videoconference

clients and server) in the same way (expanding the available collaboration tools made available to users). Ex.1003, ¶268.

Additional reasons to combine are provided in the detailed analysis below.

3. Dependent claims 6, 11, and 21

[6.0] The computer program product of claim 1 wherein the function block is further operative when executed by the processor to

See [1.0] and [1.3]. Ex.1003, ¶256.

[6.1] enable a user of the data processing system to collaborate on at least one of a document, a file, or an application with a user on another data processing system.

As discussed at [1.4], Abuan's instruction to start or end a conference are each an example of a "*request for an audio/video (A/V) service.*" In addition to microphone-sourced audio and camera-sourced video communications, Abuan describes sending additional forms of content, such as "(1) files and other content stored on the [mobile] device, (2) the screen display of the device (i.e., the content that is displayed on the device's screen), [or] (3) content received from another device...." Ex.1005, [0037]; Ex.1003, ¶257.

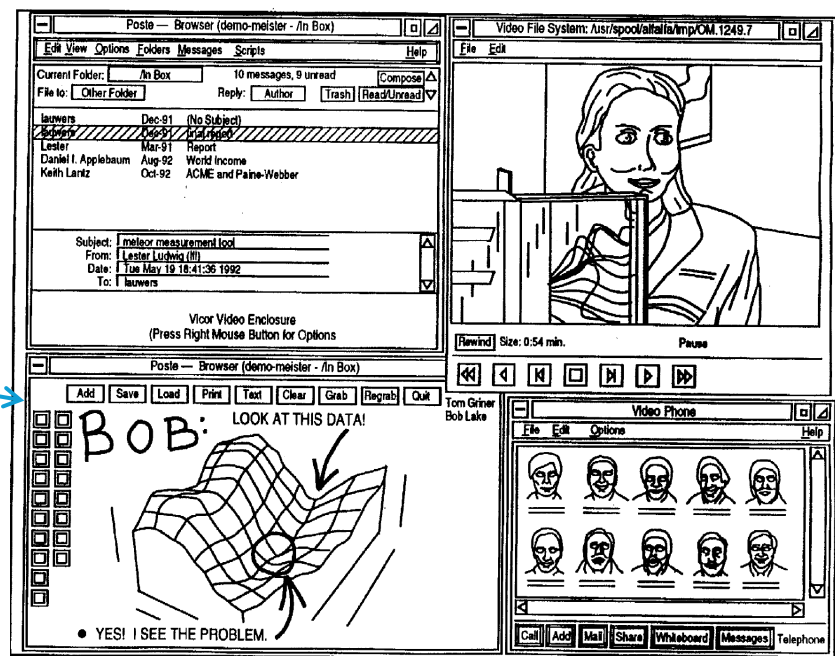
Ludwig provides further explanation and description of improving ordinary videoconferencing with additional collaboration features:

In accordance with the present invention, computer hardware, software and communications technologies are combined in novel ways to produce a multimedia collaboration system that greatly facilitates distributed

collaboration, in part by replicating the benefits of face-to-face collaboration.

Ex.1008, 2:57-62. Ludwig's "multimedia collaboration" techniques allow a user to share their computer display with other conference participants, and allow a remote participant to control their computer. Ex.1008, 9:12-18. Users can also "point at or annotate the shared data." Ex.1008, 9:22-23; *see also id.*, 9:23-29, Fig. 2B. Ludwig illustrates, for example, that multiple users have collaborated on a graph document, with one user writing "BOB: LOOK AT THIS DATA!" and another user responding "YES! I SEE THE PROBLEM":

Conference participants collaborating on a file or document



Ex.1008, Fig. 2B (annotated).

Ex.1003, ¶¶259-261.

Ludwig describes and illustrates additional examples of users collaborating

via shared documents and applications. Ex.1008, 36:49-37:6, 38:34-40, Figs. 36-37, 40; Ex.1003, ¶¶262-263.

It would have been obvious to improve Abuan's videoconferencing services to also include Ludwig's collaboration features, such as screen sharing and screen annotating, because Abuan expressly suggests incorporating such features.

Ex.1005, [0037]. Ludwig also touts these features' advantages over traditional videoconferencing systems that lack them. *See, e.g.*, Ex.1008, 2:11-33, 3:53-59. In the combined system of Abuan-Ludwig, a user's request to start a videoconference would initiate a "data conference" as described by Ludwig, allowing users to not only see and hear one another, but also to use Ludwig's collaboration tools such as screen sharing and screen annotating. Ex.1008, 3:1-6. Thus, Abuan and Ludwig render obvious "*enable a user of the data processing system to collaborate on at least one of a document, a file, or an application with a user on another data processing system.*" Ex.1003, ¶¶257-269.

Ludwig, like Abuan, describes a modular approach to software development, including a Collaboration Initiator component through which "All collaborative functions can be accessed." Ex.1008, 18:34-20:37, Fig. 20. Accordingly, it would have been obvious for Ludwig's collaboration features to be implemented in Abuan by modifying or adding to the modules in Abuan's video conference module ("*the function block*"). Ex.1003, ¶264.

[11.0] The computer program product of claim 7 wherein the function block is further operative when executed by the processor to

See [7.0] and [1.3]. Ex.1003, ¶270.

[11.1] enable a user of the data processing system to collaborate on at least one of a document, a file, or an application with a user on another data processing system.

See [6.1]. Ex.1003, ¶271.

[21.0] The method of claim 12 further comprising

See [12.0]. Ex.1003, ¶272.

[21.1] enabling, by the function block, a user of the computing entity to collaborate on at least one of a document, a file, or an application with a user on another computing entity.

As discussed at [6.1], Abuan, Lawson, Guzman, and Ludwig render obvious “*enable a user of the data processing system to collaborate on at least one of a document, a file, or an application with a user on another*” data processing system.

As discussed at [1.0], the “*data processing system*” is taught by the one or more processing unit(s), which constitute “a mobile computing device” (“*the computing entity*,” as discussed with reference to independent claim 12). Ex.1005, [0248].

Thus, a POSITA would have found limitation [21.1] obvious. Ex.1003, ¶273.

4. Dependent claim 30

Dependent claim 30 recites the same functionality as claim 6 in the form of a method claim. Further, as discussed above with reference to independent claim 22, the *computing entity* recited in claim 22 (and also in claim 30, which depends on

claim 22) includes *the data processing system* recited in claim 1 (and also in claim 6, which depends on claim 1). Accordingly, claim 30 is obvious for the same reasons discussed above for computer program product claim 6. Ex.1003, ¶¶274-275.

XIII. Conclusion

Petitioner has established a reasonable likelihood that the Challenged Claims are unpatentable.

Respectfully submitted,

Dated: January 13, 2026
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, Texas 75201
Customer No. 27683

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

XIV. Mandatory notices

A. Real party-in-interest

Pursuant to 37 C.F.R. § 42.8(b)(1), Petitioner certifies that the real party-in-interest is Cisco Systems, Inc.

B. Related matters

Pursuant to 37 C.F.R. § 42.8(b)(2), to the best knowledge of the Petitioner, the '744 Patent is or was involved in the following cases:

Case Heading	Number	Court	Filed
<i>Damaka, Inc. v Cisco Systems, Inc.</i>	2-25-cv-00593	EDTX	May 30, 2025
<i>Damaka, Inc. v. The Cigna Group</i>	2:25-cv-00594	EDTX	May 30, 2025

C. Lead and back-up counsel and service information

Lead Counsel

Theodore M. Foster
HAYNES AND BOONE, LLP
2801 Harwood St. Suite 2300
Dallas, TX 75201

Phone: (303) 382-6205
Fax: (214) 200-0853
ipr.theo.foster@haynesboone.com
USPTO Reg. No. 57,456

Back-up Counsel

Calmann J. Clements
HAYNES AND BOONE, LLP
2801 Harwood St. Suite 2300
Dallas, TX 75201

Phone: (972) 739-8638
Fax: (214) 200-0853
calmann.clements.ipr@haynesboone.com
USPTO Reg. No. 66,910

David L. McCombs
HAYNES AND BOONE, LLP
2801 Harwood St. Suite 2300
Dallas, TX 75201

Phone: (214) 651-5533
Fax: (214) 200-0853
david.mccombs.ipr@haynesboone.com
USPTO Reg. No. 32,271

Brett Bostrom
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75201

Phone: (720) 484-3715
Fax: (214) 200-0853
brett.bostrom.ipr@haynesboone.com
USPTO Reg. No. 73,009

Dagim Tilahun
HAYNES AND BOONE, LLP
2801 Harwood St. Suite 2300
Dallas, TX 75201

Phone: (202) 654-4582
Fax: (214) 200-0853
dagim.tilahun.ipr@haynesboone.com
USPTO Reg. No. 75,722

Matthew Beck
HAYNES AND BOONE, LLP
2801 Harwood St. Suite 2300
Dallas, TX 75201

Phone: (202) 654-4573
Fax: (214) 200-0853
matthew.beck.ipr@haynesboone.com
USPTO Reg. No. 79,791

Please address all correspondence in this proceeding to lead and back-up counsel. Petitioner consents to service in this proceeding by email at the addresses above on condition that Patent Owner likewise consents to email service.

CLAIMS APPENDIX

- [1.0] A computer program product in a non-transitory computer readable medium for use in a data processing system
- [1.1] [a data processing system] that comprises a processor coupled to a network interface,
- [1.2] the computer program product holding computer program instructions that comprise a function block,
- [1.3] the function block having audio/video (A/V) call functionality, the function block operative when executed by the processor to:
- [1.4] receive from a superblock application a request for an audio/video (A/V) service;
- [1.5] determine that the superblock application is authorized to access the A/V service functionality;
- [1.6] determine that the AN service requires access to a resource that is external to the data processing system;
- [1.7.1] establish the A/V service between the superblock application and the resource via the network interface,
- [1.7.2] wherein establishing is not otherwise supported by the superblock application;
- [1.8.1] provide video output for the A/V service to a video window within a

- display window associated with the superblock application,
- [1.8.2] the video window displayable simultaneously with the display window of the superblock application; and
- [1.9] provide audio for the A/V service.
- [2.0] The computer program product of claim 1 wherein the function block is further operative when executed by the processor to
- [2.1] operate as a bridge between the superblock application and the external resource.
- [3.0] The computer program product of claim 1 wherein
- [3.1] the request is an application programming interface (API) call from the superblock application to the function block.
- [4.0] The computer program product of claim 1 wherein
- [4.1] the superblock is configured to initialize the function block when the superblock application is loaded and executed.
- [5.0] The computer program product of claim 1 wherein the function block is further operative when executed by the processor to
- [5.1] determine that the superblock application is authorized to access functionality provided by the function block for the A/V service.
- [6.0] The computer program product of claim 1 wherein the function block is further operative when executed by the processor to

- [6.1] enable a user of the data processing system to collaborate on at least one of a document, a file, or an application with a user on another data processing system.
- [7.0] A computer program product in a non-transitory computer readable medium for use in a data processing system
- [7.1] [a data processing system] that comprises a processor coupled to a network interface,
- [7.2] the computer program product holding computer program instructions that comprise a function block,
- [7.3] the function block having audio/video (A/V) call functionality, the function block operative when executed by the processor to:
- [7.4] receive from a superblock application a request for a communication session,
- [7.5] wherein the function block is configured to provide functionality for the communication session to the superblock application;
- [7.6] notify the superblock application that the communication session is being requested;
- [7.7] establish a connection with an external device via the network interface to support the communication session; and
- [7.8] provide the communication session to the superblock application,

- [7.9] wherein the steps of receiving, notifying, establishing, and providing are performed without switching context from the superblock application to another application on the data processing system.
- [8.0] The computer program product of claim 7 wherein the function block is further operative when executed by the processor to
- [8.1] determine whether the function block has a capability to perform any rendering needed by the communication session.
- [9.0] The computer program product of claim 7 wherein the function block is further operative when executed by the processor to
- [9.1] determine whether the superblock has a capability to perform any rendering needed by the communication session.
- [10.0] The computer program product of claim 7 wherein the function block is further operative when executed by the processor to
- [10.1] determine that the superblock application is authorized to access functionality provided by the function block for the communication session.
- [11.0] The computer program product of claim 7 wherein the function block is further operative when executed by the processor to
- [11.1] enable a user of the data processing system to collaborate on at least one of a document, a file, or an application with a user on another data processing

system.

[12.0] A method implemented in a function block associated with a computing entity having a network interface, comprising:

[12.1] receiving from a superblock application a request for a communication session,

[12.2] wherein the function block is configured to provide functionality for the communication session to the superblock application;

[12.3] notifying the superblock application that the communication session is being requested;

[12.4] establishing a connection with an external device via a network interface of the computing entity to support the communication session; and

[12.5] providing the communication session to the superblock application,

[12.6] wherein the steps of receiving, notifying, establishing, and providing are performed without switching context from the superblock application to another application associated with the computing entity.

[13.0] The method of claim 12 wherein providing the communication session includes

[13.1] operating as a bridge between the superblock application and the external device.

[14.0] The method of claim 12 wherein

- [14.1] the request is an application programming interface (API) call from the superblock application to the function block.
- [15.0] The method of claim 12 further comprising
 - [15.1] determining whether the function block has a capability to provide the communication session.
- [16.0] The method of claim 12 further comprising
 - [16.1] determining whether the function block has a capability to perform any rendering needed by the communication session.
- [17.0] The method of claim 12 further comprising
 - [17.1] determining whether the superblock has a capability to perform any rendering needed by the communication session.
- [18.0] The method of claim 12 further comprising
 - [18.1] initializing the function block when the superblock application is loaded and executed.
- [19.0] The method of claim 12 further comprising
 - [19.1] determining that the superblock application is authorized to access functionality provided by the function block for the communication session.
- [20.0] The method of claim 12 further comprising
 - [20.1] providing at least one configuration parameter for the function block,

- [21.0] The method of claim 12 further comprising
 - [21.1] enabling, by the function block, a user of the computing entity to collaborate on at least one of a document, a file, or an application with a user on another computing entity.
- [22.0] A method implemented in a function block associated with a computing entity having a network interface, comprising:
 - [22.1] receiving from a superblock application a request for an audio/video (A/V) service;
 - [22.2] determining that the superblock application is authorized to access the A/V service functionality;
 - [22.3] determining that the A/V service requires access to a resource that is external to the computing entity;
 - [22.4.1] establishing the A/V service between the superblock application and the resource via the network interface,
 - [22.4.2] wherein establishing is not otherwise supported by the superblock application;
 - [22.5.1] providing video output for the A/V service to a video window within a display window associated with the superblock application,
 - [22.5.2] the video window displayable simultaneously with the display window of the superblock application; and

- [22.6] providing audio for the A/V service.
- [23.0] The method of claim 22 further comprising
 - [23.1] operating as a bridge between the superblock application and the external resource.
- [24.0] The method of claim 22 further comprising
 - [24.1] initializing the function block when the superblock application is loaded and executed.
- [25.0] The method of claim 22 further comprising
 - [25.1] determining that the superblock application is authorized to access functionality provided by the function block for the A/V service.
- [26.0] The method of claim 22 further comprising
 - [26.1] providing at least one configuration parameter for the function block,
 - [26.2] wherein the configuration parameter enables the function block to be configured after the function block is distributed as part of the superblock application.
- [27.0] The method of claim 22 further comprising
 - [27.1] determining whether the function block has a capability to perform any rendering needed by the A/V service.
- [28.0] The method of claim 22 further comprising
 - [28.1] determining whether the superblock has a capability to perform any

rendering needed by the communication session.

[29.0] The method of claim 22 further comprising

[29.1] providing at least one configuration parameter for the function block,

[29.2] wherein the configuration parameter enables the function block to be configured after the function block is distributed as part of the superblock application.

[30.0] The method of claim 22 further comprising

[30.1] enabling, by the function block, a user of the computing entity to collaborate on at least one of a document, a file, or an application with a user on another computing entity.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioner hereby certifies, in accordance with and reliance on the word count provided by the word-processing system used to prepare this Petition, that the number of words in this paper is 13,825. Pursuant to 37 C.F.R. § 42.24(d), this word count excludes the table of contents, table of authorities, mandatory notices under § 42.8, certificate of service, certificate of word count, appendix of exhibits, and any claim listing.

Dated: January 13, 2026

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. § 42.6(e), the undersigned certifies that a true and correct copy of the foregoing was served on the Patent Owner as detailed below:

<i>Date of service</i>	January 13, 2026
<i>Manner of service</i>	FedEx
<i>Documents served</i>	PETITION FOR <i>INTER PARTES</i> REVIEW UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104; PETITIONER’S POWER OF ATTORNEY; PETITIONER’S EXHIBIT LIST AND EXHIBITS Ex.1001-1043, 1045, 1046, 1049
<i>Persons Served</i>	Munck Wilson Mandala L.L.P 2000 McKinney Ave Ste 1900 Dallas, TX 75201

/Theodore M. Foster/

Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456