

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS INC.,

Petitioner

IPR2026-00206

U.S. Patent No. 9,027,032

**DECLARATION OF BRUCE MCNAIR,
UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION
FOR *INTER PARTES* REVIEW**

TABLE OF CONTENTS

I.	INTRODUCTION	4
II.	QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	8
III.	LEVEL OF ORDINARY SKILL IN THE ART	11
IV.	RELEVANT LEGAL STANDARDS	12
V.	BACKGROUND	14
VI.	OVERVIEW OF THE '032 PATENT	19
VII.	CLAIM CONSTRUCTION	23
VIII.	IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE....	23
A.	Ground 1: Claim 1-7 and 9-20 would have been Obvious over Abuan in view of Lawson and Guzman.....	24
1.	Summary of Abuan	24
2.	Summary of Guzman	30
3.	Summary of Lawson	33
4.	Claim 1	35
5.	Claim 2	77
6.	Claim 3	79
7.	Claim 4	85
8.	Claim 5	86
9.	Claim 6	89
10.	Claim 7	92
11.	Claim 9	93
12.	Claim 10	108
13.	Claim 11	111
14.	Claim 12	116
15.	Claim 13	118
16.	Claim 14	119

17.	Claim 15	120
18.	Claim 16	123
19.	Claim 17	124
20.	Claim 18	124
21.	Claim 19	125
22.	Claim 20	125
B.	Ground 2: Claims 8 and 21 would have been Obvious over Abuan in view of Lawson, Guzman, and Ludwig.	126
1.	Summary of Ludwig	126
2.	Claim 8	129
3.	Claim 21	136
IX.	CONCLUSION.....	136

I, Bruce McNair, do hereby declare as follows:

I. INTRODUCTION

1. I am making this declaration at the request of Cisco Systems, Inc. in the matter of the *Inter Partes* Review of U.S. Patent No. 9,027,032 (“the ’032 patent”) to Chaturvedi et al.

2. I am being compensated for my work in this matter at my standard hourly rate. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this proceeding. My compensation is not contingent on the outcome of this matter or the specifics of my testimony.

3. I have been asked to provide my opinions regarding whether the subject matter of claims 1-21 (“the Challenged Claims”) of the ’032 patent would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. It is my opinion that the Challenged Claims would have been obvious to a POSITA.

4. In the preparation of this declaration, I have studied:

- a. the ’032 patent, Ex.1001;
- b. the prosecution history of the ’032 patent (“’032 File History”), Ex.1002;
- c. U.S. Patent Publication No. US 2011/0249077 to Abuan et al. (“Abuan”), Ex.1005;

d. U.S. Patent Publication No. US 2012/0092438 to Guzman Suarez et al. (“Guzman”), Ex.1006;

e. U.S. Patent Publication No. US 2013/0019297 to Lawson et al. (“Lawson”), Ex.1007; and

f. U.S. Patent No. US 5,689,641 to Ludwig et al. (“Ludwig”), Ex.1008.

5. In forming the opinions expressed below, I have considered: the documents listed above; the relevant legal standards, including the standard for obviousness; and my own knowledge and experience based upon my work in the field of wireless network communications as described below, as well as portions of the following additional materials:

a. D. E. Knuth, Selected Papers on Computer Science (Cambridge Univ. Press), 1996, Ex.1009;

b. B. W. Boehm, Software Engineering Economics (Prentice-Hall, Inc.), 1981, Ex.1010;

c. U.S. Patent No. 5,339,431, Ex.1011;

d. U.S. Patent No. 5,615,400, Ex.1012;

e. U.S. Patent No. 5,946,486, Ex.1013;

f. U.S. Patent No. 6,243,764, Ex.1014;

g. U.S. Patent No. 5,802,367, Ex.1015;

Declaration of Bruce McNair
Inter Partes Review of U.S. 9,027,032

- h. U.S. Patent No. 5,218,697, Ex.1016;
- i. U.S. Patent No. 5,498,003, Ex.1017;
- j. U.S. Patent No. 6,163,858, Ex.1018;
- k. U.S. Patent Publication No. 2012/0019610, Ex.1019;
- l. U.S. Patent Publication No. 2008/0178264, Ex.1021;
- m. U.S. Patent Publication No. 2005/0235044, Ex.1022;
- n. U.S. Patent Publication No. 2006/0037064, Ex.1023;
- o. U.S. Patent Publication No. 2008/0201438, Ex.1024;
- p. U.S. Patent Publication No. 2012/0170722, Ex.1025;
- q. U.S. Patent Publication No. 2005/0265322, Ex.1026;
- r. U.S. Patent No. 6,314,402, Ex.1027;
- s. U.S. Patent Publication No. 2005/0060183, Ex.1028;
- t. U.S. Patent Publication No. 2009/0193394, Ex.1029;
- u. U.S. Patent Publication No. 2008/0022327, Ex.1030;
- v. U.S. Patent Publication No. 2002/0199019, Ex.1031;
- w. U.S. Patent No. 6,101,607, Ex.1032;
- x. U.S. Patent No. 6,311,283, Ex.1033;
- y. U.S. Patent No. 6,295,048, Ex.1034;
- z. U.S. Patent Publication No. 2009/0310023, Ex.1035;
- aa. U.S. Patent Publication No. 2008/0127183, Ex.1036;

- bb. U.S. Patent No. 7,574,202, Ex.1037;
- cc. U.S. Patent No. 7,461,144, Ex.1038;
- dd. U.S. Patent Publication No. 2007/0185998, Ex.1039;
- ee. A. V. Aho et al., Principles of Compiler Design, (Addison-Wesley), 1977, Ex.1040;
- ff. IEEE 100, The Authoritative Dictionary of IEEE Standard Terms, 7th Ed. (IEEE Press), 2000, Ex.1041;
- gg. A. Silberschatz et al., Operating System Concepts, 8th Ed. (John Wiley & Sons, Inc.), 2009, Ex.1042;
- hh. U.S. Patent No. 6,820,235, Ex.1043;
- ii. U.S. Patent Publication No. 2011/0249073, Ex.1044;
- jj. U.S. Patent No. 5,835,435, Ex.1045;
- kk. U.S. Patent No. 5,742,641, Ex.1046;
- ll. U.S. Patent Publication No. 2009/0106830, Ex.1047;
- mm. RFC3261, SIP: Session Initiation Protocol, 2002, Ex.1048; and
- nn. U.S. 2014/0184723; Ex.1049.

6. Unless otherwise noted, all **emphasis** in any quoted material has been added. Claim terms are *italicized*.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

7. My complete qualifications and professional experience are described in my *Curriculum Vitae*, a copy of which can be found in Exhibit 1004. The following is a brief summary of my relevant qualifications and professional experience.

8. I received my Bachelors of Engineering and Masters of Engineering—Electrical Engineering from the Stevens Institute of Technology in 1971 and 1974, respectively. I also completed qualifiers and course work for a Ph.D. in computer science.

9. I have over 55 years of industry experience in the areas of computer systems and electronic circuits, including substantial time spent working with software, security and access control technologies. Before starting high school, I obtained an FCC amateur radio license and commercial radio telephone license and began learning about electronic circuits. While in high school in the 1960s, I was first exposed to computer programming and continued to add to my knowledge in both areas in college. My work experience includes seven years as a GS-0855 Electronic Engineer working for the US Army Electronics Command, a year as a Junior Member of Technical Staff at ITT Defense Communications Division, 24 years as a Member of Technical Staff and Technical Manager at AT&T/Bell

Laboratories and 15 years as a full-time faculty member of the Electrical and Computer Engineering Department at Stevens Institute of Technology.

10. While I was working for the US Army Electronics Command at Fort Monmouth, NJ, I was responsible for support of TSEC/KG-27 bulk encryptor, a classified encryption device used by the Army for protecting multi-channel digital voice communications lines. Also, while I was at Fort Monmouth, my responsibilities included research and development for the cryptographic aspects of SINCGARS, the Single Channel Ground/Airborne Radio System. As part of this work, I wrote computer simulation code for speech processing and system simulations and programmed microcontrollers to create real-time prototypes.

11. During my employment at AT&T/Bell Labs, in the late 1970s and early 1980s, I designed protocols for wide area public data networks, similar to the Internet. I also designed software and hardware for analog modems and secure voice terminals. This work included all of the user interface and real-time hardware driver software for a secure voice terminal and signal processing software to perform real-time speech compression using software development kits (SDKs) and application program interfaces (APIs) for standard microprocessors in the x86 family and various custom digital signal processors (DSPs).

12. From 1985 until 1987, I supervised a group at AT&T Bell Labs involved in the research and design of speech recognition and speaker verification

technologies. This work resulted in the design and implementation of one of the earliest cellular phones with speech recognition technology, the AT&T 3050/3450.

13. From 1987 until 1994, I led the AT&T Bell Labs Security and System Reliability Architecture Group. In the course of this activity, I investigated security in all aspects of AT&T products and services, particularly focused on access control and user authorization techniques. As part of this effort, I also led efforts to employ the newly developed AT&T Smart Card as a personal authenticator for use in the AT&T networks operations centers providing operations support to personnel controlled access to the network support terminals. I was also involved in video processing for a planned AT&T service providing personal video conferencing much like today's Zoom, FaceTime, or WebEx as well as early work with General Magic on multiapplication PDAs.

14. From 1994 until 2002, I was a member of the AT&T Bell Labs and AT&T Labs research organization investigating emerging wireless technologies including extensions to the capability of 2G wireless systems and exploration of technology for what we now know as 4G and 5G systems. To perform these investigations, I was responsible for digital, analog and RF hardware design and extensive real-time signal processing software design with a multiprocessor system I created.

15. I am named on multiple domestic and international patents related to wireless systems, geolocation, encryption devices, voice-based security systems, and computer/network security.

16. Since 2002, I have taught undergraduate and graduate courses at Stevens Institute of Technology including courses in digital system design and computer architecture. Since 2017, I have continued to teach graduate courses in the Stevens on-line graduate program. Between 2002 and 2017, I advised numerous undergraduate design projects including several involving web based systems and computer interfaces and drivers.

17. Also, since 2002, I have consulted on computer and network systems for various technical organizations and for intellectual property matters with technologies similar to those of the '032 patent.

III. LEVEL OF ORDINARY SKILL IN THE ART

18. I understand there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

19. 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 '032 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.

20. For purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a POSITA would have understood or known generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the '032 patent.

IV. RELEVANT LEGAL STANDARDS

21. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '032 patent, I am relying on certain basic legal principles that Cisco's counsel has explained to me.

22. I understand that prior art to the '032 patent includes patents and printed publications in the relevant art that predate the claimed priority date of the '032 patent. For purposes of this Declaration, I am applying July 16, 2013, as the priority date of the '032 patent.

23. I have been informed by Cisco's counsel that a claimed invention is unpatentable under 35 U.S.C. § 103 if the differences between the claimed

invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a POSITA. I have also been informed by Cisco's counsel that the obviousness analysis considers factual inquiries, including the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter.

24. I have been further informed by Cisco's counsel that there are several recognized rationales for combining references or modifying a reference to show obviousness. These rationales include: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device (method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art or to combine prior art teachings to arrive at the claimed invention.

V. BACKGROUND

25. As of the July 2013 priority date of the '032 patent, computer science and software programming had been developing as technical fields of study for at least forty years—and, same say, for several millennia. Ex.1009, 5-6. 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”).

26. 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.

27. As computers grew in their capabilities, 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.

28. 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.US1011, 1:24-28; Ex.1012, 1:25-29. This second step 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.

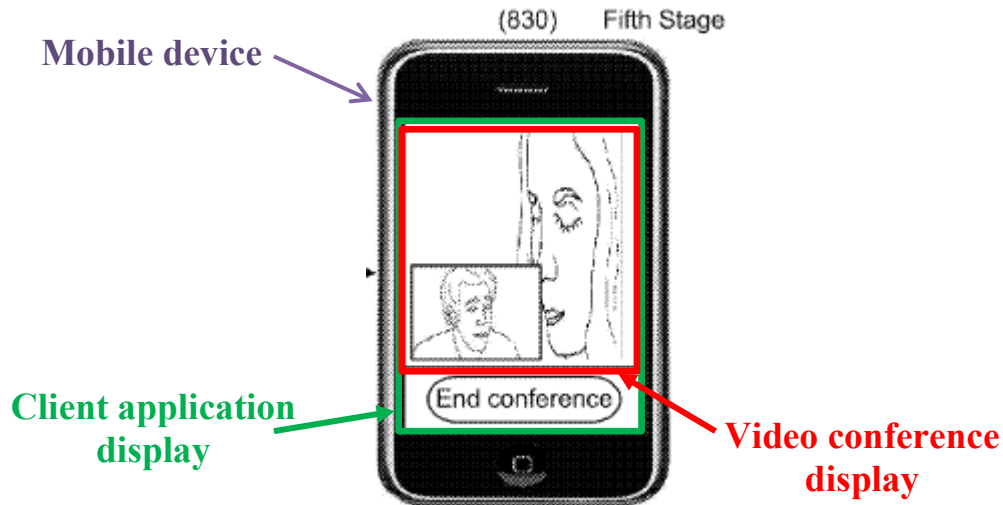
29. As software became even more complex, programmers recognized the value of allowing commonly used functions to be shared among multiple programs. Sharing allows multiple programs to call the same functions but avoids having each program incorporate its own redundant copy of the call functions. Technology 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 ’032 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.”).

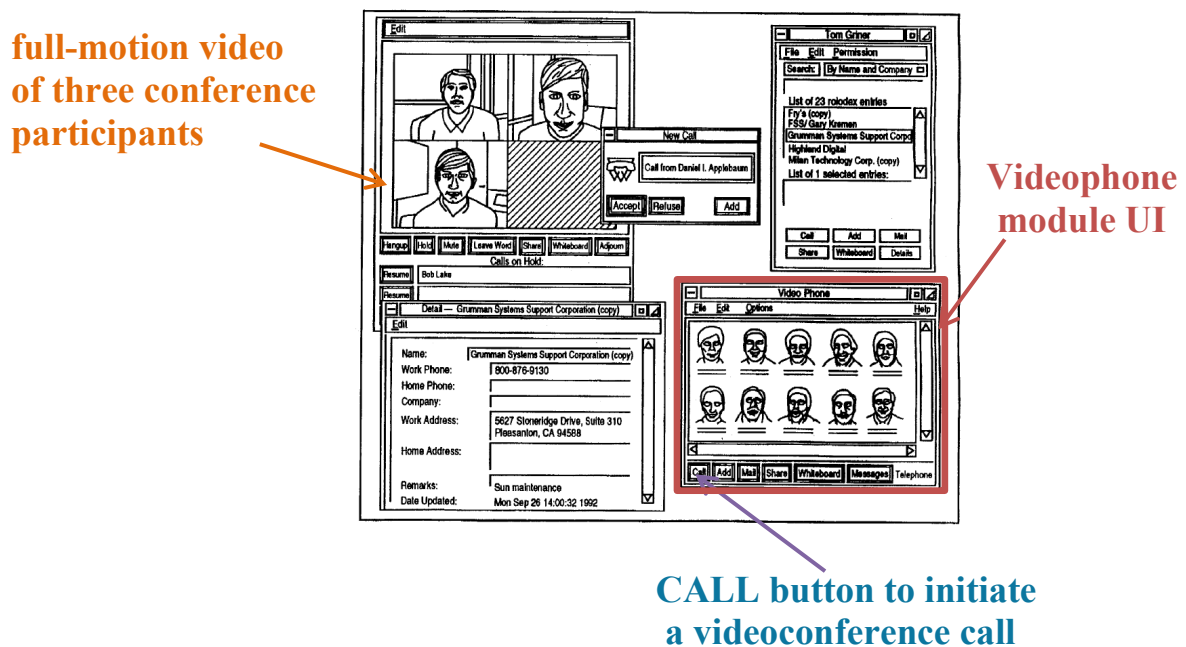
30. Also well-known before the ’032 patent’s earliest priority date was a modular approach for providing the functions to the calling programs, where functionalities, such as video conferencing functionalities, were made accessible to calling programs in the form of self-contained components or modules. 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.

31. 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 '032 patent:



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



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

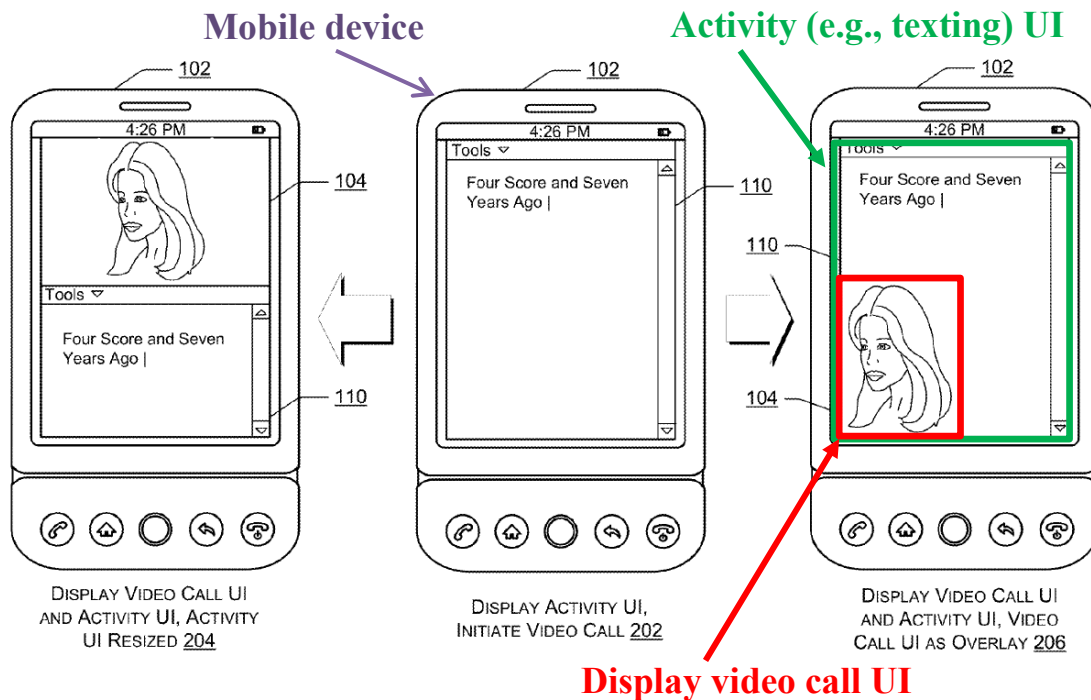


FIG. 2

Ex.20140184723, Fig. 2 (annotated).

32. All of the features and functionality described in this section were well-known for decades before the priority date of the '032 patent, and a POSITA would have been familiar with them.

VI. OVERVIEW OF THE '032 PATENT

33. The '032 patent purports to address “system and method ... for improving functionality in software applications.” Ex.1001, Abstract. For example, “the functionality of [a] superblock application is constrained to some degree by [a] device 100 and by the instructions of the superblock itself.” Ex.1001, 2:22-24. The '032 patent explains that “additional functionality” can be provided to the superblock 104 by a “function block 200 [that] includes instructions for providing

the superblock 104 with one or more functions (e.g., capabilities) that are not otherwise possessed by the superblock 104.” Ex.1001, 2:56, 3:53-56. “[T]he function block 200 may be provided as a set of instructions that are included in the superblock 104.” Ex.1001, 4:5-7. An example of “the function block 200 may be ... a software developer's kit (SDK).” Ex.1001, 4:7-8. “The function block is configured to provide functions that are accessible to the superblock application via an application programming interface (API).” Ex.1001, Abstract.

34. The '032 patent describes “compil[ing] or otherwise includ[ing] the function block instructions in the superblock 104 [to] ensure[] that the function block 200 will occupy the same memory space as the superblock 104.” Ex.1001, 4:11-16. Figure 3A of the '032 patent below shows a superblock application incorporating a function block therein, where “an application programming interface (API)” “enabl[es] the superblock 104 to interact with the function block 200.” Ex.1001, 5:26-6:3. For instance, “the superblock 104 may make API calls to the function block 200 to access the function block's capabilities. The function block 200 then provides services to the superblock 104 in response to the API calls.” Ex.1001, 6:3-6.

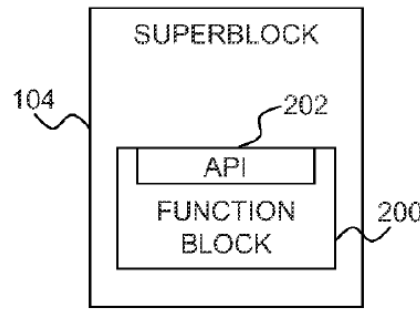


FIG. 3A

Ex.1001, Fig. 3A

35. The '032 patent provides “audio, video, ... [and] conferencing” as example functions provided by the function block. Ex.1001, 3:56-61. For example, “the function block 200 ... connect[s],” “us[ing] one or more external services 204,” “to another device in order to establish [an audio/video call] session.” Ex.1001, 4:55-57, 46. “The external services 204 may be provided via a server.” Ex.1001, 4:46-47. When “provid[ing] ... video capabilities for the superblock 104,” the function block 200 “provid[es] those capabilities within the superblock application,” as illustrated in Figure 2B below. Ex.1001, 3:62-65. The '032 patent describes “visual functions [being] presented without leaving the superblock application display. In other words, there is no need to switch context ... to any other application in order to access the additional functionality provided by the function block 200.” Ex.1001, 3:67-4:4.

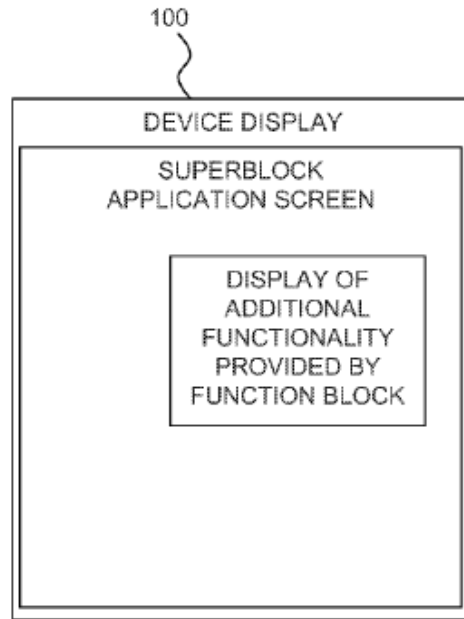


FIG. 2B

Ex.1001, Fig. 2B

36. In summary, the '032 patent merely combines known elements—application programming interfaces (APIs), an application utilizing APIs to call SDKs/libraries, static-linked libraries with functions that can be executed without switching from the context of a calling application, a server that provides an audio/video session—into one system. This agglomeration of known ideas did not advance human knowledge or understanding, and the '032 patent claims should not have been allowed to issue.

37. As I will explain below, these concepts were well-known as of the priority date of the '032 patent.

VII. CLAIM CONSTRUCTION

38. It is my understanding that in order to properly evaluate the '032 patent, the terms of the claims must first be interpreted. It is my understanding that for the purposes of this *inter partes* review, the claims are to be construed under the so-called *Phillips* standard, under which claim terms are given their ordinary and customary meaning as would have been understood by a POSITA in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. I have also been informed that claim terms only need to be construed to the extent necessary to resolve the obviousness inquiry.

39. I have reviewed the entirety of the '032 patent, as well as its prosecution history. It is my opinion that for purposes of applying the prior art presented herein to evaluate the patentability of the Challenged Claims, no terms require formal construction.

VIII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

40. The discussion in this Declaration provides a detailed analysis of how the asserted prior art references teach each limitation of the Challenged Claims.

41. As part of my analysis, I have considered, and discuss in detail, the scope and content of the prior art and any differences between the alleged invention and the prior art.

42. It is my opinion that the alleged invention recited in the Challenged Claims would have been obvious in view of the teachings of the asserted prior art and the knowledge of a POSITA.

A. Ground 1: Claim 1-7 and 9-20 would have been Obvious over Abuan in view of Lawson and Guzman.

1. Summary of Abuan

43. U.S. Patent Publication No. US 2011/0249077 to Abuan et al. (Ex.1005, “Abuan”) was filed on June 6, 2010, and published on October 13, 2011. Ex.1005, 1 (fields 22 (filing date) and 43 (publication date)). Abuan is titled “Video Conference Network Management for a Mobile Device.” Ex.1005, 1(field 54). Abuan was not before the Examiner during prosecution of the ’032 patent.

44. 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. Ex.1005, [0043], [0099], [0056]. The modules’ functionalities are accessed via “a set of application program interfaces (APIs).” Ex.1005, [0044]. Figures 13 and 14 of Abuan below show “a video conferencing and processing module 1300” that “includes a client application 1365, a video conference module

1302, a media exchange module 1320, a buffer 1325, [and] a captured image processing unit (CIPU) driver 1330,” as well as “frame buffers 1405 and 1410, audio processing manager 1415, and audio driver 1420.” Ex.1005, [0170], [0183].

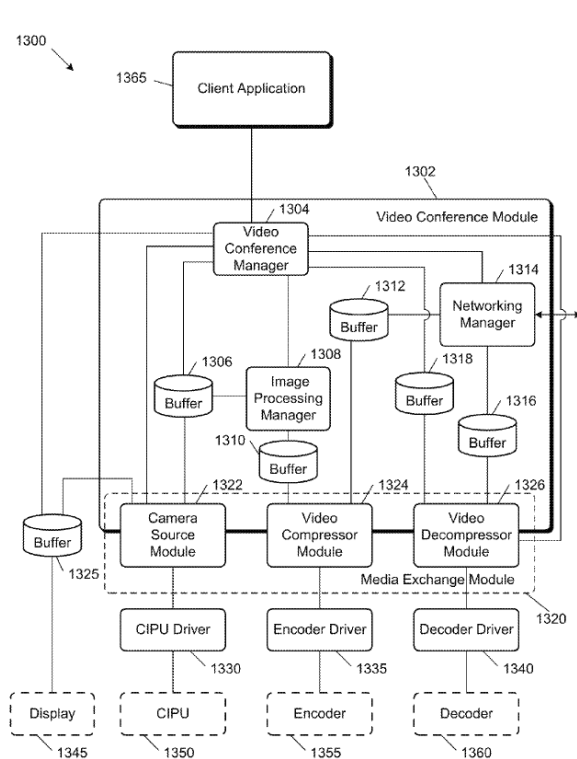


Figure 13

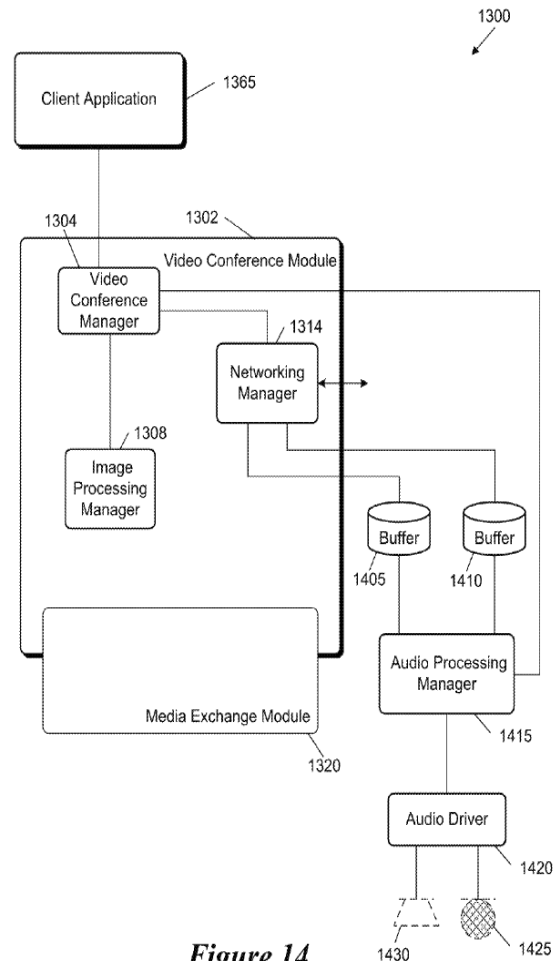


Figure 14

Ex.1005, Figs. 13 and 14

45. Abuan describes “the client application 1365,” which “may be ... implemented as a stand-alone application,” as “an application that uses the video conferencing functions of the video conference module 1302, such as a video conferencing application, a voice-over-IP (VoIP) application (e.g., Skype), or an

instant messaging application.” Ex.1005, [0171]. “[T]o start a conference and end a conference,” the client application 1365 “sends instructions to the video conference module 1302.” Ex.1005, [0172]. “The client application 1365 ... generates user interfaces that are displayed on the dual camera mobile device.” Ex.1005, [0172].

46. Abuan discloses that “the video conference manager 1304 is responsible for initializing some or all of the other modules of the video conference module 1302 (e.g., the image processing manager 1308 and the networking manager 1314) when a video conference is starting.” Ex.1005, [0175]. “The image processing manager 1308 ... processes images captured by the cameras of the ... mobile device” and “the networking manager 1314 manage[s] one or more connections between the dual camera mobile device and the other device participating in the video conference.” Ex.1005, [0178]-[0179].

47. Figure 7 of Abuan shown below illustrates a “video conference request messaging sequence 700 [for establishing a video conference] among a video conference client 710 running on a device 705, a video conference server 715, and a video conference client 725 running on a device 720.” Ex.1005, [0108]. “[T]he video conference clients 710 and 725 are the same as” the client application 1365. Ex.1005, [0108] (“the video conference clients 710 and 725 are the same as the video conference client 645”), [0171] (“the client application 1365 is the same as the video conference client 645”).

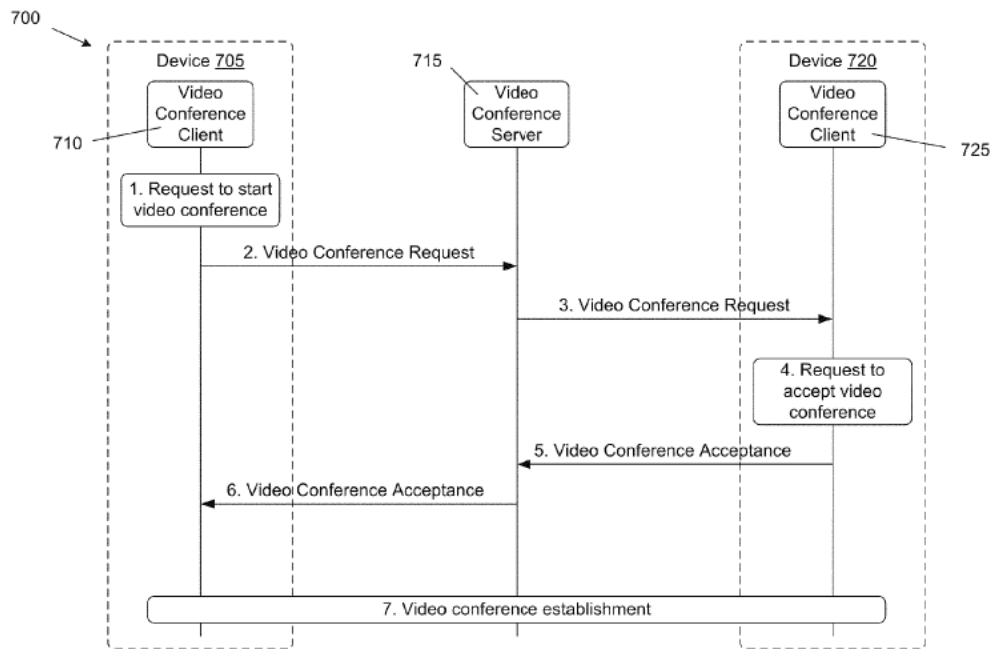


Figure 7

Ex.1005, Fig. 7

48. In Figure 7, “the video conference client 710 receives (at operation 1) a request from a user of the device 705 to start a video conference with the device 720” and in response “sends (at operation 2) a video conference request, which indicates the device 720 as the recipient based on input from the user, to the video conference server 715.” Ex.1005, [0110]-[0111]. “The video conference server 715 ... routes messages among [the] video conference clients,” for example, “[t]he video conference server 715 forwards (at operation 3) the video conference request to the video conference client 725 of the device 720” and also “forwards (at operation 6) the video conference acceptance [from the video conference client 725] to the video conference client 710.” Ex.1005, [0109], [0111], [0113]. “[A]

video conference between the device 705 and the device 720” is then established.

Ex.1005, [0114]. The network layer of the video conference modules “performs some or all of the networking functionalities for [the] video conferencing.”

Ex.1005, [0104], [0102] (“The video conference module 625 includes ... a network layer”). Figure 8 shows the “user interface (“UI”) 805 of the device 800” “illustrat[ing] the start of such a video conference by a ... mobile device 800.”

Ex.1005, [0117]. Figure 9 illustrates “the sequence of operations for presenting and accepting a video conference invitation at the remote user's device.” Ex.1005, [0129].

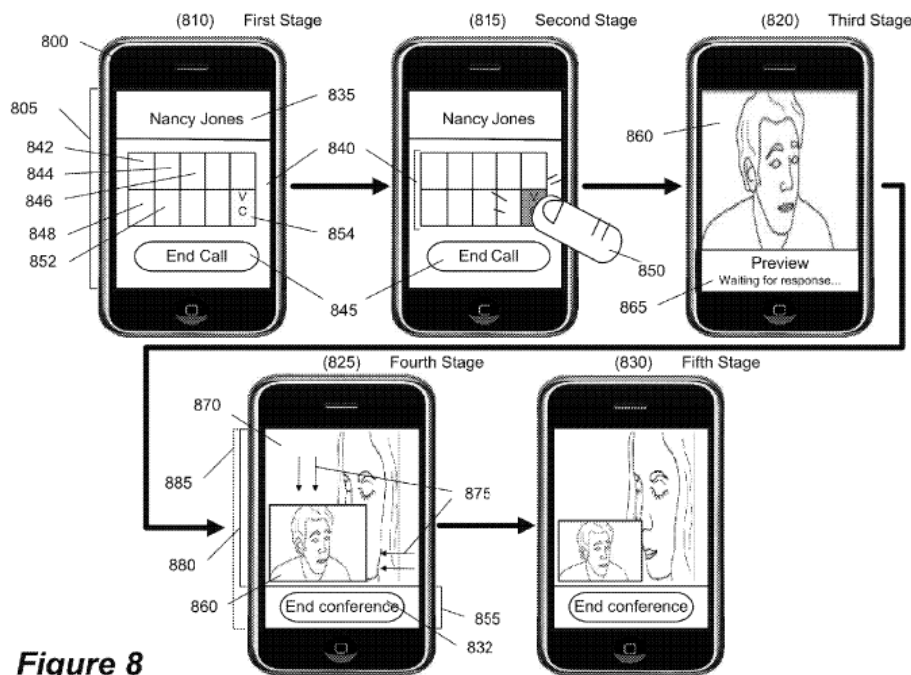
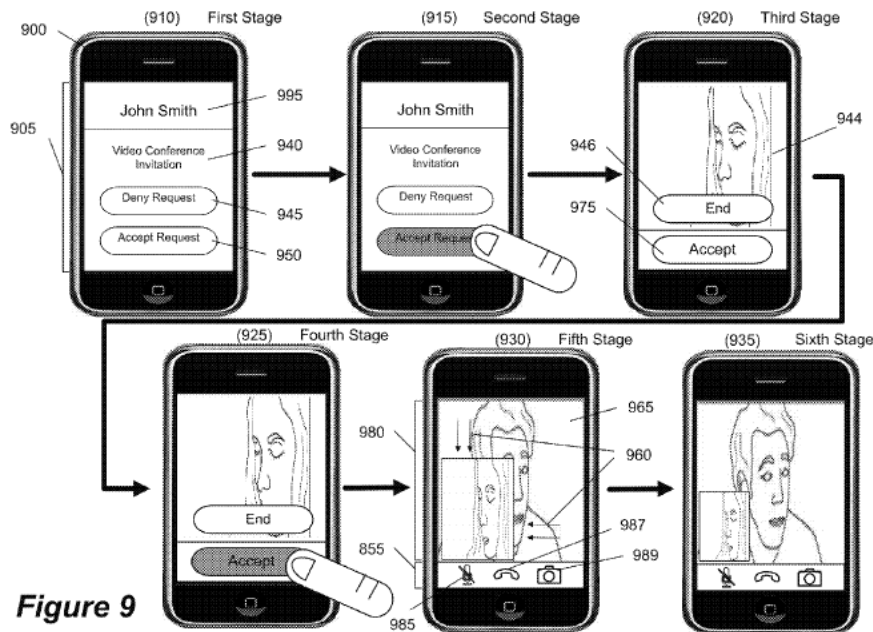


Figure 8

Ex.1005, Fig. 8



Ex.1005, Fig. 9

49. Abuan is analogous art to the '032 patent. First, it is directed to the same field of endeavor as the '032 patent, namely, the “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:15-16; *see* Ex.1005, [0001]-[0002] (describing ways that “functionality” is accessed on “portable devices, such as smartphones”). Abuan and the '032 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 '032 patent both describe at length features for facilitating videoconferencing. Ex.1001, Abstract, 3:53-65 (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”).

50. Second, Abuan is analogous art to the '032 patent because it is reasonably pertinent to a problem the inventors of the '032 patent allegedly attempted to solve. The '032 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, 3:33-36. The '032 patent’s proposed solution is to provide functionality—such as communication functions—to a superblock application. Ex.1001, 3:53-4:4. 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].

2. Summary of Guzman

51. U.S. Patent Publication No. US 2012/0092438 to Guzman Suarez et al. (“Guzman”) was filed on Oct. 18, 2010, and published on Apr. 19, 2012. Ex.1006, 1 (fields 22 (filing date) and 43 (publication date)). Guzman is titled “Overlay for a Video Conferencing Application.” Ex.1006, 1 (field 54). Guzman was not before the Examiner during prosecution of the '032 patent.

52. 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].

53. Guzman is analogous art to the ’032 patent. First, it is directed to the same field of endeavor as the ’032 patent, namely, the “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:15-16; *see* Ex.1006, [0001]–[0002] (describing how “mobile phones[] allow users to engage in video conferences”). Guzman and the ’032 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 ’032 patent both describe at length features for facilitating videoconferencing.

Ex.1001, Abstract, 3:53-65 (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”).

54. Second, Guzman is analogous art to the ’032 patent because it is reasonably pertinent to a problem the inventors of the ’032 patent allegedly attempted to solve, namely, the need to provide functionality—such as communication functions—to a superblock application. Ex.1001, 3:53-4:4. 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].

3. Summary of Lawson

55. U.S. Patent Publication No. US 2013/0019297 to Lawson et al. (“Lawson”) was filed on Sept. 21, 2012, and published on Jan. 17, 2013. Ex.1006, 1 (fields 22 (filing date) and 43 (publication date)). Lawson is titled “System and Method for Communicating with a Client Application.” Ex.1007, 1 (field 54). Lawson was not before the Examiner during prosecution of the ’032 patent.

56. 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]. An authorization token bestows “privileges for its bearer,” i.e., an “authorization token can preferably be used to verify if a bearer can perform a particular action.” Ex.1007, [0048], [0052]. “Example privileges conferred by a token can include, but are not limited to, a user's abilities to make and/or receive calls.” Ex.1007, [0048]. For example, “an authorization token can permit a communication request to establish a communication session that supports communications selected from audio, video, chat, and/or other suitable communication mediums.” Ex.1007, [0048].

57. An authorization token can also be used to limit or prohibit the bearer from performing certain actions, i.e., an authorization token can be used to indicate “which operations to limit.” Ex.1007, [0052]. The “restriction on the communications ... associated with the authorization token” include restrictions on “number of communications, timing of communications, communication endpoints, and communication mediums.” Ex.1007, [0052].

58. Lawson is analogous art to the '032 patent. First, it is directed to the same field of endeavor as the '032 patent, namely, the “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:15-16; 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 '032 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 '032 patent both describe features for facilitating audio and video communications. Ex.1001, Abstract, 3:53-65; Ex.1007, [0034], [0048] (“communication session that supports communications selected from audio, video, chat, and/or other suitable communication mediums”), [0057].

59. Second, Lawson is analogous art to the '032 patent because it is reasonably pertinent to a problem the inventors of the '032 patent allegedly attempted to solve, namely, the need to provide functionality—such as communication functions—to a superblock application. Ex.1001, 3:53-4:4. 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].

4. Claim 1

a. [1.0] *A mobile device comprising:*

60. To the extent that the preamble is limiting, Abuan discloses it or renders it obvious.

61. Abuan is directed to “Video Conference Network Management for a **Mobile Device**” and discloses “**a mobile device** with two cameras that can take pictures and videos” and is capable of “transmit[ing] the captured images to one or more devices during a real-time communication session between the users of the devices.” Ex.1005, Title, [0003]. For example, Figure 7 of Abuan “illustrates an example video conference request messaging sequence 700” between client devices 705 and 720. Ex.1005, FIG. 7, [0108]. “Examples of [Abuan’s] **mobile devices** include mobile phones, smartphones, personal digital assistants (PDAs), laptops, tablet personal computers, or any other type of mobile computing device.” Ex.1005, [0032]; *see also* [0001]. Abuan’s client device or mobile device teaches “*a mobile device*.”

b. [1.1] *a network interface*;

62. Abuan discloses that its mobile device “has *a network interface* that allows the device to transmit the captured images to one or more devices during a real-time communication session between the users of the devices.” Ex.1005, [0003]. Abuan describes an “implementation of a mobile computing device” that includes “a wireless communication subsystem(s) 1925,” which “may include radio frequency receivers and transmitters, and optical receivers and transmitters.” Ex.1005, [0249]-[0251]. A POSITA would have understood that Abuan’s network

interface is implemented as a wireless communication subsystem(s), teaching “*a network interface.*”

c. [1.2] *a processor coupled to the network interface; and*

63. Abuan discloses that an “implementation of [its] mobile computing device can include **one or more processing units 1905.**” Ex.1005, [0249]. The “one or more processing units 1905” “can [] be **coupled**” to “a peripherals interface 1915,” which in turn “can be **coupled to**” the “wireless communication subsystem(s) 1925.” Ex.1005, [0249]-[0250], FIG. 19. As discussed at [1.1], the wireless communication subsystem(s) 1925 teaches “*a network interface.*” Thus, Abuan discloses one or more processing units 1905 (“*a processor*”) coupled to the wireless communication subsystem(s) 1925, teaching “*a processor coupled to the network interface.*”

64. Further, Abuan discloses that “[t]he video conference module 625 ... sends data to and receives data from networks ... through the network interface 650.” Ex.1005, [0101]. Since the video conference module 625 can be implemented as a software and “software processes” can be “executed by one or more processing unit(s),” as discussed at [1.0] above, a POSITA would have found it obvious for the *processor to be coupled to the network interface* to facilitate such sending and receiving of data. Ex.1005, [0233]; Ex.1019, [0032] (“programmatic

elements, often referred to as modules” “may include a program, a subroutine, a portion of a program, or a software component”).

d. [1.3] *a memory coupled to the processor and configured to store a plurality of instructions executable by the processor, the instructions including instructions for:*

65. Abuan discloses that an “implementation of [its] mobile computing device can include ... memory interface 1910,” which “can be coupled to memory 1970, which can include high-speed random access memory and/or non-volatile memory such as flash memory.” Ex.1005, [0249], [0253]. The “memory interface 1910” “can also be coupled” to “one or more processing units 1905.” Ex.1005, [0249]. Thus, Abuan discloses a memory 1970 (“*memory*”) coupled to one or more processing units 1905 (“*the processor*”), teaching “*a memory coupled to the processor.*”

66. Further, Abuan discloses that the “[m]emory can [] include communication instructions 1974 to facilitate communicating with one or more additional devices; graphical user interface instructions 1976 to facilitate graphic user interface processing; image/video processing instructions 1978 to facilitate image/video-related processing and functions; phone instructions 1980 to facilitate phone-related processes and functions; media exchange and processing instructions 1982 to facilitate media communication and processing-related processes and functions; camera instructions 1984 to facilitate camera-related processes and

functions; and video conferencing instructions 1986 to facilitate video conferencing processes and functions.” Ex.1005, [0254]. Abuan also discloses that the “set of instructions recorded on a computer readable storage medium (also referred to as computer readable medium) ... are executed by one or more processing unit(s) (e.g., one or more processors, cores of processors, or other processing units).” Ex.1005, [0233]. The “machine-readable medium includes magnetic disks, optical disks, random access memory; read only memory, flash memory devices, etc.” Ex.1005, [0248].

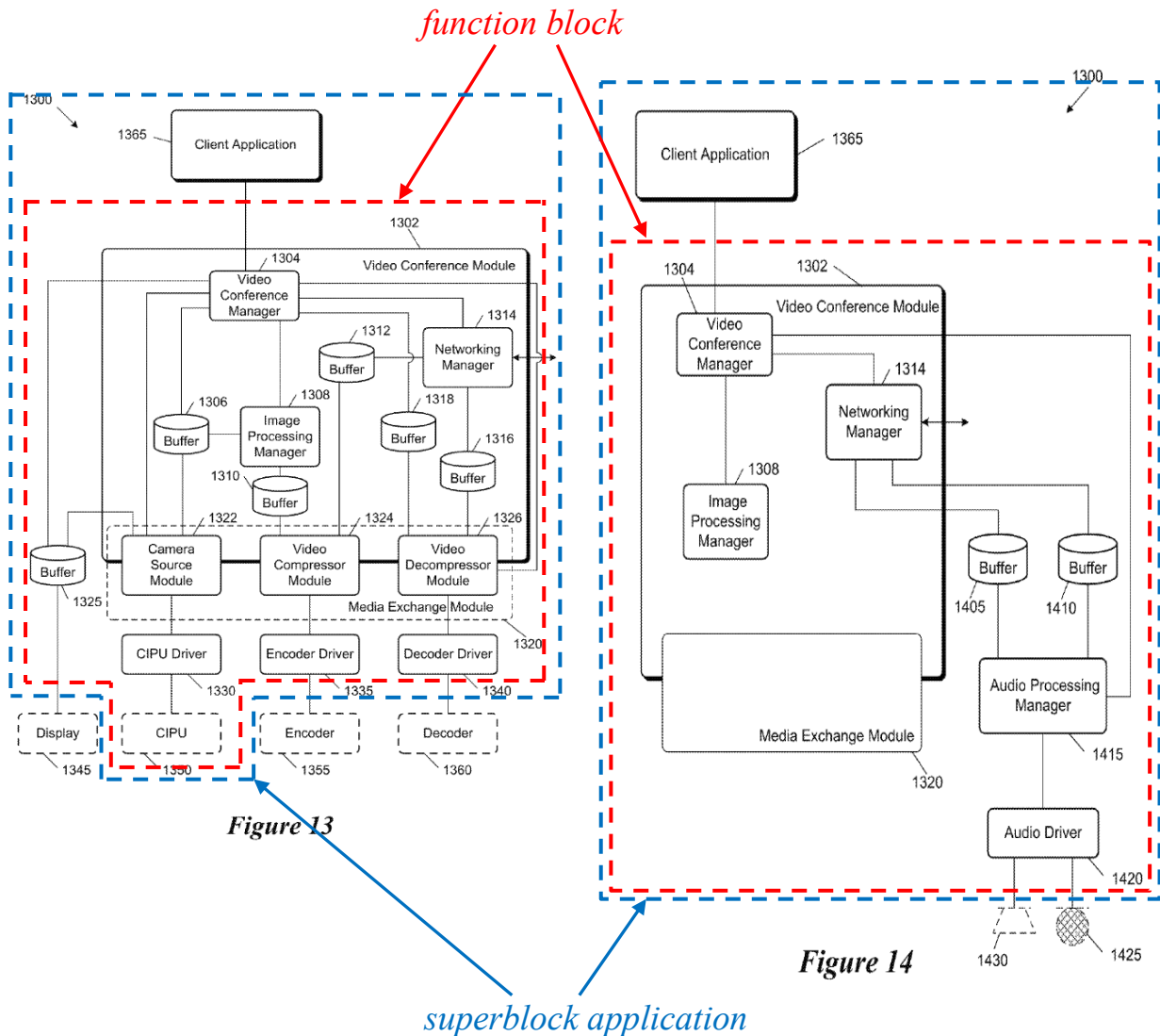
67. Thus, Abuan teaches “*a memory ... configured to store a plurality of instructions executable by the processor, the instructions including instructions for.*”

- e. **[1.4] *receiving, by a function block stored in the memory of the mobile device, a request for an audio/video (A/V) call from a superblock application stored in the memory of the mobile device;***

68. Abuan discloses that the “video conferencing and processing module 1300 includes a client application 1365, a video conference module 1302, a media exchange module 1320, a buffer 1325, [and] a captured image processing unit (CIPU) driver 1330,” as well as “frame buffers 1405 and 1410, audio processing manager 1415, and audio driver 1420.” Ex.1005, [0170], [0183], FIG. 13, FIG. 14. The video conferencing and processing module 1300 also includes “a captured

image processing unit (CIPU),” where “all of the modules of the CIPU 400 are implemented in software.” Ex.1005, FIG. 13, [0055]-[0056].

69. The video conferencing and processing module 1300 as described above teaches “*a superblock application*.” The video conferencing and processing module 1300 excluding the client application 1365 teaches “*a function block*.” Figures 13 and 14 of Abuan are annotated below to show the above-described “*superblock application*” and “*function block*.”



Ex.1005, Figs. 13 and 14 (annotated).

70. As discussed further below at [1.6], Guzman also describes software modules for performing certain functions, including an “an account setup and verification module.” Ex.1006, [0222]. It would have been obvious to incorporate the functionality of Guzman’s modules into Abuan as additional modules, which are therefore also included in the identified “*function block*.” See *infra* [1.6].

71. The video conferencing and processing module 1300 can be implemented as software and “software processes” can be “specified as a set of instructions recorded on a computer readable storage medium (also referred to as computer readable medium).” Ex.1005, [0233]; Ex.1019, [0032] (“programmable elements, often referred to as modules” “may include a program, a subroutine, a portion of a program, or a software component”). Software also includes “firmware residing in read-only memory.” Ex.1005, [0233]. Thus, a POSITA would have found it obvious for the video conferencing and processing module 1300 (“*a superblock application*”) and the video conferencing and processing module 1300 excluding the client application 1365 (“*a function block*”) each to be recorded or stored in a memory or computer readable medium of the mobile device (“*stored in the memory of the mobile device*”).

72. Abuan discloses that the “client application 1365 may be an application that uses the video conferencing functions of the video conference module 1302.” Ex.1005, [0171]. For example, the client application 1365 “sends instructions to the video conference module 1302 such as instructions to start a video conference and end a video conference.” Ex.1005, [0172]. The video conference would include “the video conference module 625 ... send[ing] data (e.g., images captured by cameras of the dual camera mobile device) to another device on the Internet.” Ex.1005, [0106]. Abuan discloses that the video

conference manager 1304 of the video conference module 1302 (part of “*a function block*”) “receiv[es] (at 1505) [the] instructions to start a video conference” “from the client application 1365.” Ex.1005, [0186]-[0187].

73. It would have been obvious to a POSITA for the video conference to also include audio, i.e., the video conference is an audio/video conference. *See, e.g.*, Ex.1005, [0138] (“muting the audio of the other user during the video conference”), [0149] (“during a video conference, audio data is transmitted”), [0153] (“mobile devices in a video conference communicate data (e.g., audio and video images”), [0185] (“multi-participant audio conference”). The audio/video conference to be started by the video conference module 1302 is an example of “the video conferencing functions of the video conference module 1302” that are used by the client application 1365. Ex.1005, [0171]. Abuan also discloses that “the video conference module 625 [which is the same as the video conference module 1302] receives instructions from ... the video conference client 645 [which is the same as the client application 1365].” Ex.1005, [0101], [0171] (“[T]he client application 1365 is the same as the video conference client 645”), [0174] (“the video conference module 1302 is the same as the video conference module 625”).

74. Thus, Abuan discloses that the video conference module 1302 (part of “*the function block*”) receives instructions (“*a request*”) from the client application 1365 (part of “*a superblock application*”) to start an audio/video conference call

(“an audio/video (A/V) call”) for use by the client application 1365, teaching
“receiving, by a function block ..., a request for an audio/video (A/V) call from a
superblock application.”

f. [1.5] *determining, by the function block, that the superblock application is authorized to access A/V call functionality provided by the function block;*

75. Abuan, in combination with Lawson, render obvious this limitation.

76. As discussed at [1.4], Abuan teaches that the video conference module 1302 (part of “the function block”) receives instructions from the client application 1365 (part of “the superblock application”) to start an audio/video conference call (“A/V call functionality”) for use by the client application 1365.

77. Figure 7 of Abuan illustrates an audio/video conference call established after a “video conference client 710 [running on a device 705] sends ... a video conference request, which indicates the device 720 as the recipient ..., to the video conference server 715,” indicating that the video conference client 710 is authorized, using the video conference module 1302, to communicate with the video conference server 715 (as discussed at [1.7.1] below) and the user of device 705 is already logged into the video conference server 715. Ex.1005, [0111], [0108].

78. 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].

79. Lawson discloses a “mobile device 14 of the system 10” “for communicating with a client application.” Ex.1007, [0046], [0042]. “[T]he communication is preferably an open media channel (e.g., voice/audio, video, [] or any suitable communication session) between the client application and a device of the communication destination.” Ex.1007, [0057]. The mobile device 14 includes a cloud interface module 46 that “further include[s] one or more security and/or resource management tools.” Ex.1007, [0047]-[0048]. An example of a security and/or resource management tool is “an authorization token, which can be defined as an electronic listing of possible privileges for its bearer.” Ex.1007, [0048].

80. Lawson explains that “[t]he authorization token can be used to authorize actions when included in communication requests.” Ex.1007, [0048]. “Example privileges conferred by a[n] [authorization] token can include ... a user’s abilities to make and/or receive calls.” Ex.1007, [0048]. For instance, “an authorization token can permit a communication request to establish a communication session that supports communications selected from audio, video, chat, and/or other suitable communication mediums.” Ex.1007, [0048].

81. Because an “authorization token is [] created for a client application,” the permission to establish a communication session, which the authorization token

grants, belongs to the client application, the bearer of the authorization token.

Ex.1007, [0052]. The “authorization token can be verified against an account key” “[a]t the communication platform or at a service communicatively coupled to the communication platform.” Ex.1007, [0052]. The communication platform can be “one or more separate servers and/or services configured for trafficking and connecting communications between [] users.” Ex.1007, [0051].

82. Accordingly, the combination of Abuan and Lawson renders it obvious that the client application 1365 includes an authorization token in the instructions it sends to the video conference module 1302. The authorization token, when verified, authorizes the client application 1365 to “make and/or receive” audio/video conference calls using the video conference module 1302. Ex.1007, [0048]. That is, when the video conference module 1302 (part of the “*function block*”) receives the instructions and the authorization token, it verifies the received authorization token (e.g., using a communication platform such as a server) and determines that (“*determining, by the function block, 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,” i.e., “make and/or receive” audio/video conference calls using the video conference module 1302 (“*is authorized to access A/V call functionality provided by the function block*”). Ex.1005, [0171].

83. A POSITA would have been motivated to combine the teachings of Lawson with Abuan for multiple reasons.

84. Abuan generally describes client applications for facilitating videoconference communications. Ex.1005, 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” (and preventing the attendant waste of resources and bandwidth of the videoconference server) and providing “secure[] access [to] application services and resources.” Ex.1007, [0028], [0035]. Lawson’s techniques are imminently “adapted to function for any suitable type of communication,” including “video calls.” Ex.1007, [0028].

85. A POSITA would have found it obvious to incorporate Lawson’s authentication and authorization security measures in the videoconferencing system of Abuan. Abuan’s combination with Lawson is simply the combination 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’s videoconferencing system to authorized users). Further, the combination is merely the use of known techniques

(e.g., logging into a server) to improve similar devices (Abuan's videoconference clients and server) in the same way ("prevent[ing] malicious parties from posing" as someone else). Ex.1007, [0028].

86. 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].

g. [1.6] *determining, by the function block, that the A/V call requires access to a resource that is external to the mobile device;*

87. Abuan, in combination with Guzman, render obvious this limitation.

88. As discussed at [1.4], Abuan discloses that the video conference module 1302 (part of “*the function block*”) receives instructions from the client application 1365 to start an audio/video conference call (“*the A/V call*”).

89. Figure 7 of Abuan shown below illustrates the “video conference request messaging sequence 700” that occurs for the video conference module 1302 at a mobile device to start an audio/video conference call. Specifically, Abuan discloses that “a video conference request, which indicates the device 720 as the recipient based on input from the user,” is sent by mobile device 705 “to the video conference server 715.” Ex.1005, [0111]. A POSITA would have understood video conference server 715 to be a server that is separate from mobile device 705 for several reasons. First, the video conference server 715 is shown in Figure 7 to be a separate device from the device 705 running the video conference client 710. Ex.1005, [0108]. Second, because a video conference requires at least two separate devices, as shown in Figure 7 of Abuan, a POSITA would have understood that the video conference server 715 is separate from device 705 and/or device 720, and having the video conference server 715 be separate from device 705 would have been an obvious design choice. Ex.1005, FIG. 7. Third, Abuan teaches that “the video conference client[] 710,” “running on a 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.),”

indicating that the video conference server 715 is separate from device 705.

Ex.1005, [0108]-[0109].

90. Thus, video conference server 715 is “a resource that is external to the mobile device.”

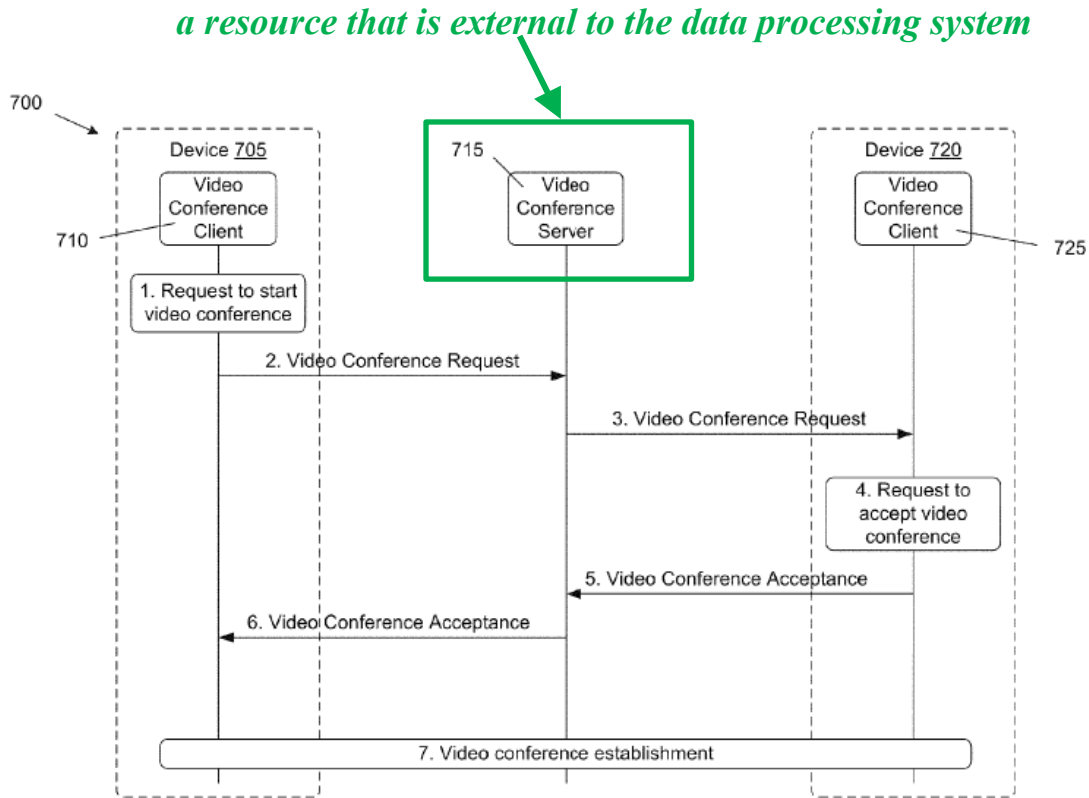


Figure 7

Ex.1005, Fig. 7 (annotated).

91. As discussed at [1.5], Abuan’s Figure 7 assumes that the video conference client 710 is authorized to use the video conference module 1302 to communicate with the video conference server 715 and that the user of device 705 is already logged into the video conference server 715. 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 it was common for a server to require a user login to access the server's services at the time. *See* 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"). 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].

92. Guzman discloses a "video conferencing application [] for conducting a video conference between [a] first electronic device and a second electronic device." Ex.1006, Abstract. Like Abuan's video conference module 1302 that provides video conferencing functions to the client application 1365 on a mobile device, Guzman's video conferencing application "enables a user of [an electronic] device to engage in a video conference with a remote user of a second electronic

device that also includes a video conferencing application.” Ex.1005, [0171];
Ex.1006, [0034].

93. Guzman discloses that the “video conferencing application 2200 includes ... an account setup and verification module 2210” that “manages ... the verification of [] user accounts upon login.” Ex.1006, [0222], [0230]. Guzman explains that the verification process checks to see if an account or a client has auto-login (i.e., persistent login) set up:

The process 300 then determines (at 315) whether an auto login option has been previously selected by the user. The selection can be made by checking a box labeled “enable auto-login” (e.g., through a click operation) in some embodiments, or through other user interface mechanisms. Selecting the auto login option 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.

When the process 300 determines that the user has previously selected the auto login option, the process 300 determines (at 370) whether the previously entered user name and password are valid. Various embodiments verify the user name and password 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, [0056]-[0057].

94. Accordingly, Guzman teaches that, before allowing the establishment of a video conference call, the video conferencing application determines whether access to a remote server that stores login information is required to validate user credentials (or whether the login information can be validated “on the device with the video conferencing application”). Ex.1006, [0057]. Thus, the combination of Abuan and Guzman renders obvious that when the video conference module 1302 receives instructions from client application 1365 to start an audio/video conference call, the video conference module 1302 would determine whether access to a remote server that stores login information is needed to validate user credentials and establish the audio/video conference call.

95. 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.

96. As another example, Guzman discloses that, when a “user has not selected to sign in,” the video conferencing application “continues to display (at 320) the initial setup and configuration information in the overlay display area” until “the user chooses to sign in,” rendering it obvious that the video conferencing application confirms that the user has provided username and password values through the user interface. 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.

97. As discussed above in this section, it would have been obvious to a POSITA for the video conference server 715 to be separate from the mobile device on which client application 1365 is running. It would also have been obvious to a POSITA for the video conference server 715, to serve as a remote server that stores login information. First, Guzman contemplates “a centralized server ... associated with the video conferencing application,” rendering it obvious that the video conference server 715 can be the centralized server that “routes messages among video conference clients” and also “stores login information” associated with the video conferencing application. Ex.1006, [0089], [0057]; Ex.1005, [0109]. Second, Guzman teaches that its remote server that stores login information may also provide 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)”). Guzman also 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]. Third, in the context of Abuan’s videoconference system shown in Figure 7, the videoconference server is the only server.

98. Accordingly, the combination of Abuan and Guzman renders obvious that the video conference module 1302 (part of “*the function block*”) determines whether access to the video conference server 715 (“*a resource that is external to the mobile device*”) is needed to validate user credentials and establish the audio/video conference call, rendering obvious “*determining, by the function block, that the A/V call requires access to a resource that is external to the mobile device.*”

99. A POSITA would have been motivated to combine the teachings of Abuan and Guzman for multiple reasons.

100. 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 Figure 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. Allowing unrestricted and unlimited use of a server-provided feature would create the undesirable risk the server becoming overwhelmed and failing, either intentionally (in a denial-of-service attack) or unintentionally. Ex.1037, 19:5-10 (“In 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 (“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.”). Restricting access to authorized users mitigates that risk.

101. 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].

102. 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 combination 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). Further, 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). 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).

103. A POSITA would also have been motivated to combine Abuan with Guzman because 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].

h. [1.7.1] *establishing, by the function block, the A/V call between the superblock application and the resource via a network interface,*

104. Abuan at least renders obvious this limitation because Abuan discloses that a video conference client running on a mobile device uses the mobile device’s network interface to engage in an audio/video conference call that is routed via a remote video conference server. For example, Abuan describes initiating and establishing an audio/video conference call (“*establishing... the A/V call*”), for example, by exchanging call setup messages with a videoconference server (“*the resource*”). Ex.1005, [0108]-[0109], FIG. 7. For instance, Abuan discloses that 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].

105. Specifically, with reference to Figure 7 of Abuan shown below, Abuan discloses “a video conference client 710 running on a device 705” that

“receives (at operation 1) a request from a user of the device 705 to start a video conference with the device 720.” Ex.1005, [0108], [0110]. “[T]he video conference client 710 sends (at operation 2) a video conference request ... to the video conference server 715,” which “forwards (at operation 3) the video conference request to the video conference client 725 of the device 720,” which in return “sends (at operation 5) a video conference acceptance to the video conference server 715.” Ex.1005, [0111]-[0112]. “[T]he video conference server 715 [then] forwards (at operation 6) the video conference acceptance to the video conference client 710,” after which a video conference is “establish[ed] (at operation 7) ... between the device 705 and the device 720.” Ex.1005, [0114]. This messaging sequence is shown in Figure 7 of Abuan below, which shows a “video conference establishment” between the video conference client 710 and the video conference client 725 via the video conference server 715. Ex.1005, FIG. 7. As discussed at [1.4], the video conference call is an audio/video conference call.

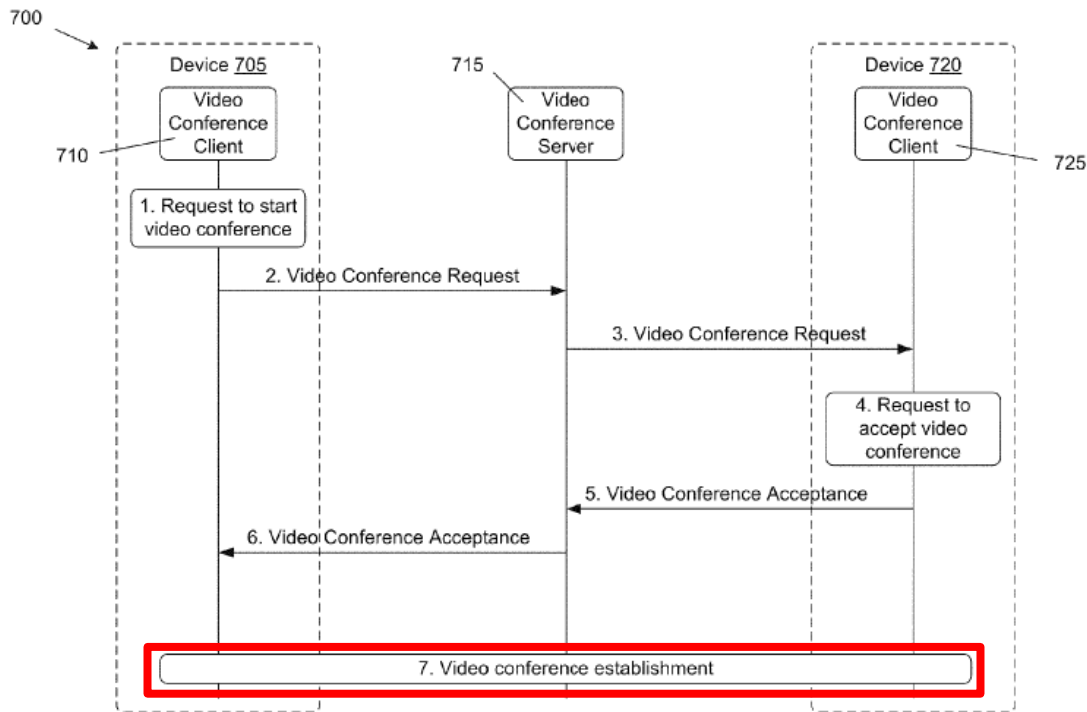


Figure 7

Ex.1005, Fig. 7 (annotated).

106. Abuan further discloses that “[e]ach of the video conference clients 710 and 725 ... 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]. Each of video conference clients 710 and 725 is the same as the client application 1365, because Abuan explains that “video conference clients 710 and 725 are the same as the video conference client 645,” which in turn is the same as “the client application 1365.” Ex.1005, [0108], [0170].

107. Thus, when the audio/video conference call is established between the video conference clients 710 and 725, the audio/video conference call extends from the client application 1365 (part of “*the superblock application*”) on device 705 to its network interface (“*a network interface*”) to a network (e.g., cellular network) to the video conference server 715 (“*the resource*”), and further to video conference client 725 on device 720, rendering obvious “*establishing ... the A/V call between the superblock application and the resource via a network interface.*”

108. The communications between the video conference clients 710 (on device 705) and 725 (on device 720) and the video conference server 715 that established the audio/video conference call are performed by the video conference module 1302 (“*establishing, by the function block, the A/V call*”), as Abuan teaches that i) “[t]he video conference module 1302 is the same as the video conference module 625,” ii) “[t]he video conference module 625 includes ... a network layer 640,” and iii) “the network layer 640 performs ... all of the networking functionalities for video conferencing”). Ex.1005, [0102], [0104], [0174].

109. For example, Abuan discloses that the networking manager 1314 of the video conference module 1302 (or network layer 640 of the video conference module 625) “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]; *see also* [0104]. It would have been obvious for the networking

manager 1314 to establish such network connections using a network interface like network interface 650. For instance, Abuan teaches that the videoconference client device 705 “communicates with the video conference server 715 over a network ... through a network interface such as the network interface 650” and that “the network interface 650 is a communication interface that allows the video conference module 625 and the video conference client 645 to send data and receive data over a network.” Ex.1005, [0109], [0106]; *see also* [0053], [0101]. Accordingly, 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 “*establishing... the A/V call*,” such as sending and receiving request and acceptance messages via the network interface.

i. [1.7.2] *wherein the establishing is not supported by the superbloc application;*

110. Abuan at least renders obvious this limitation for several reasons.

111. First, as discussed at [1.4], the “client application 1365 may be an application that uses the video conferencing functions of the video conference module 1302,” examples of which include “a voice-over-IP (VoIP) application (e.g., Skype) [and] an instant messaging application.” Ex.1005, [0171]. It was well-known before the priority date of the ’032 patent that IM applications were text-based and lacked video calling or conferencing capabilities. 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”). It was also well-known before the priority date of the ’032 patent that VoIP was a technique for exchanging audio communications in real time and did not have video conferencing functionalities. 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)”).

112. Thus, a POSITA would have found it obvious for the client application 1365 (part of “*the superblock application*”) on device 705 to be a text-only IM application or audio-only VoIP application that is not capable of establishing a video conference call (and as such, an audio/video conference call) with the video conference server 715 without “us[ing] the video conferencing functions of the video conference module 1302,” rendering obvious “*wherein the establishing is not supported by the superblock application.*” Ex.1005, [0171].

113. Further, it would have been obvious to a POSITA that a client application 1365 “that uses the video conferencing functions of the video conference module 1302” would not itself have video conferencing capabilities. Ex.1005, [0171]. This is 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. Although we must pay the added expense of linking and loading if we produce relocatable object modules, we gain a great deal of flexibility in being able to compile subroutines separately and call other previously compiled programs from an object module.” Ex.1040, 518-519.

114. For example, Abuan identifies the phone application as another “application that uses the video conferencing functions of the video conference module 1302.” Ex.1005, [0171]. Abuan discloses that “while [a] user of a mobile device is engaged in a phone call with a second user, the user can turn the phone call into a video conference with the permission of the other party.” Ex.1005, [0117]. With reference to Figure 8 shown below, Abuan explains that, after “a phone call has been established between the handheld mobile device user and Nancy Jones” (first stage 810), “the user can turn the phone call into a video conference” by “select[ing] the selectable Video Conference option 854 (e.g., through a single finger tap by finger 850)” and “activat[ing] a video conference

tool.” Ex.1005, [0117], [0119]. In other words, activating the video conference tool
“allows the user to start a video conference during the phone call.” Ex.1005,
[0119].

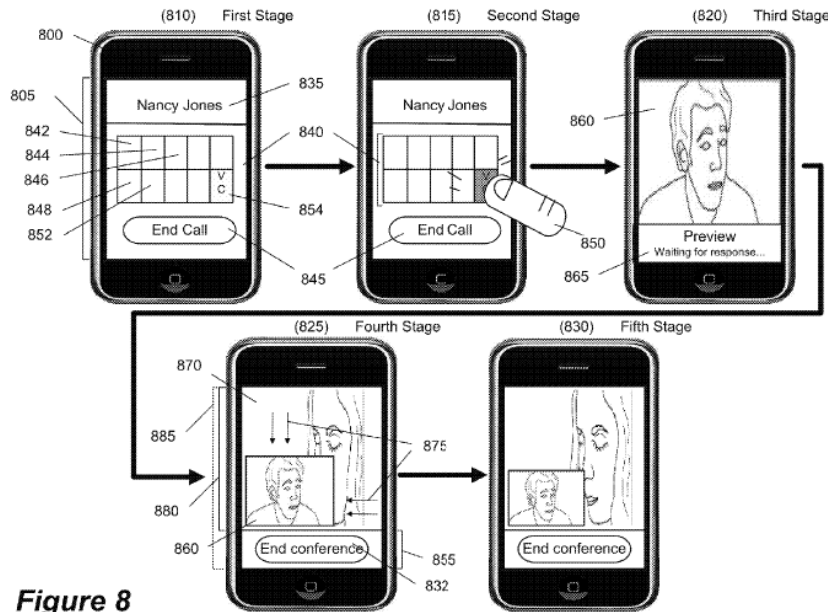


Figure 8

Ex.1005, Fig. 8

115. A POSITA would have understood that i) the phone application of the phone call from which the video conference tool is activated to start a video conference corresponds to a client “application [1365] that uses the video conferencing functions of” the video conference tool and ii) the “video conference tool” that is activated to establish a video conference corresponds to “the video conference module 1302” that provides “video conferencing functions.” Ex.1005, [0171], [0119]. Thus, it would have been obvious to a POSITA for the client application 1365 (part of “*the superbloc application*”) on device 705 to be a

phone application that is not capable of establishing a video conference call (and as such, an audio/video conference call) with the video conference server 715 via the network without having to activate the video conference module 1302 and using its video conferencing functions, rendering obvious “*wherein the establishing is not supported by the superblock application.*”

116. As another example, Abuan identifies the contacts application as another “application that uses the video conferencing functions of the video conference module 1302.” Ex.1005, [0171]. Abuan discloses that “[i]n addition to activating the video conference tool through a selectable option during a phone call, some embodiments allow a user of a dual camera device to initiate a video conference directly without having to make a phone call first.” Ex.1005, [0144]. For example, with reference to Figure 11 shown below, a user, “looking through a contacts list on [the] mobile device,” “selects the person 1155 with whom he would like to have a video conference (e.g., through a single finger tap 1160 on the person's name 1155)” (“second stage 1115”) and then selects “the Video Conference option 1172 (e.g., through a single finger tap)” to activate a “video conference tool” and initiate a video conference (“fourth stage 1125”). Ex.1005, [0145], [0147].

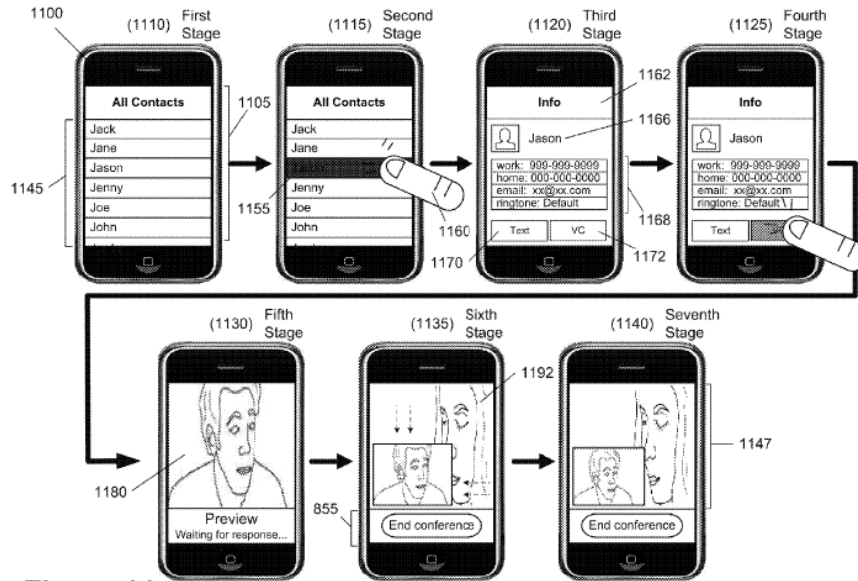


Figure 11

Ex.1005, Fig. 11

117. A POSITA would have understood that i) the contacts application listing the contacts on the user's phone from which the video conference tool is activated to start a video conference corresponds to a client "application [1365] that uses the video conferencing functions of" the video conference tool and ii) the "video conference tool" that is activated to establish a video conference corresponds to "the video conference module 1302" that provides "video conferencing functions." Ex.1005, [0171], [0119]. Thus, it would have been obvious to a POSITA for the client application 1365 (part of "*the superbloc application*") on device 705 to be a contacts application that is not capable of establishing a video conference call (and as such, an audio/video conference call) with the video conference server 715 via the network without having to activate the

video conference module 1302 and using its video conferencing functions, rendering obvious “*wherein the establishing is not supported by the superblock application.*”

118. Second, as discussed at [7.0] below, the client application 1365 is an example of an API-calling application program and the video conference module 1302 is an example of an API-implementing, where “[a]n API defines the language and parameters that API-calling components use when accessing and using specified features of the API-implementing component.” Ex.1005, [0241]. That is, 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].

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].

119. 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”). Thus, a POSITA would have found it obvious for Abuan’s client application 1365 (part of “*the superblock application*”) to access functionalities of various software modules, including the video conferencing functions of the video conference module 1302 that establish (“*the establishing*”) the audio/video conference call with the video conference server 715, that were *not supported* in the client application itself.

120. Third, a POSITA would have known that API-calling components, such as Abuan’s client application 1365 (part of “*the superblock application*”), only included the API calls to the functionalities of API-implementing components, such as the video conference module 1302 (part of “*the function*

block”), and not the functions themselves. Ex.1032, 5:44-48 (“In accordance with the API modularity concept or paradigm, only calls 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”). 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. Maintaining multiple redundant codebases for performing the same functionality would have been seen as wasteful and potentially problematic. That is, it would have been obvious to a POSITA for the client application 1365 (“*the superblock application*”) *not to support* the functionality provided by the video conference module 1302 (including the video conferencing functions of the video conference module 1302 that establish (“*the establishing*”) the audio/video conference call with the video conference server 715)—or the other “*function block*” modules identified in [1.4]—because that would render the modules superfluous and unnecessary and would have gone against the known “API modularity concept or paradigm.” Ex.1032, 5:44-45.

121. The above analysis is consistent with the '032 patent's description that “[t]he function block 200 includes instructions for providing the superblock 104

with one or more functions (e.g., capabilities) that are not otherwise possessed by the superblock 104. Such functions ... may include ... video, ... conferencing, meetings, and/or other functions.” Ex.1001, 3:53-61. The ’032 patent also describes “[t]he function block 200 ... as part of the superblock application.” Ex.1001, 4:37-38. Therefore, the ’032 patent describes the function block 200, which is a part of the superblock 104, providing its one or more functions or capabilities to the part of the superblock 104 that excludes the function block 200.

j. [1.8.1] *providing, by the function block, video output for the A/V call to a video window within a display window of the superblock application on the mobile device,*

122. As discussed at [1.7.1], the audio/video conference call extending from the client application 1365 (part of “*the superblock application*”) on device 705 to device 720 teaches “*the A/V call.*”

123. Abuan discloses that its mobile devices are capable of “display[ing] two videos ... captured by two cameras of two different devices that are in a video conference.” Ex.1005, [0035]. Specifically, Abuan describes that “when a user of the dual camera mobile device accepts ... a video conference request or when a user of another device accepts a request sent by the user of the dual camera mobile device,” the video conference manager 1304 of the video conference module 1302 (part of “*the function block*”) “generates (at 1520) composite images for displaying on the dual camera mobile device (i.e., local display). These composite images

may include ... various combinations of images from the cameras of the local dual camera mobile device and images from cameras of the other device participating in the video conference.” Ex.1005, [0186], [0190]. Abuan describes “composite display [as] a PIP display.” Ex.1005, [0036]. A POSITA would have understood PIP as an acronym for “picture-in-picture.” Ex.1006, [0132] (“picture-in-picture (PIP) display”). A POSITA would also have understood that the images are video images. Ex.1005, [0034] (“One example of a real-time communication session that involves the transmission of the captured video images is a video conference”).

124. Figure 8 of Abuan, which illustrates a video conference being established from an ongoing phone call, as discussed at [1.7.1], shows “a PIP display 880 that is presented by the UI 805 [of a mobile device 800] during the video conference.” Ex.1005, [0126], [0117]. The “PIP display 880 [] has a foreground inset display 860 of the local user’s video and a background main display 870” “that contains the video from a camera of the remote device” “of the remote user.” Ex.1005, [0124].

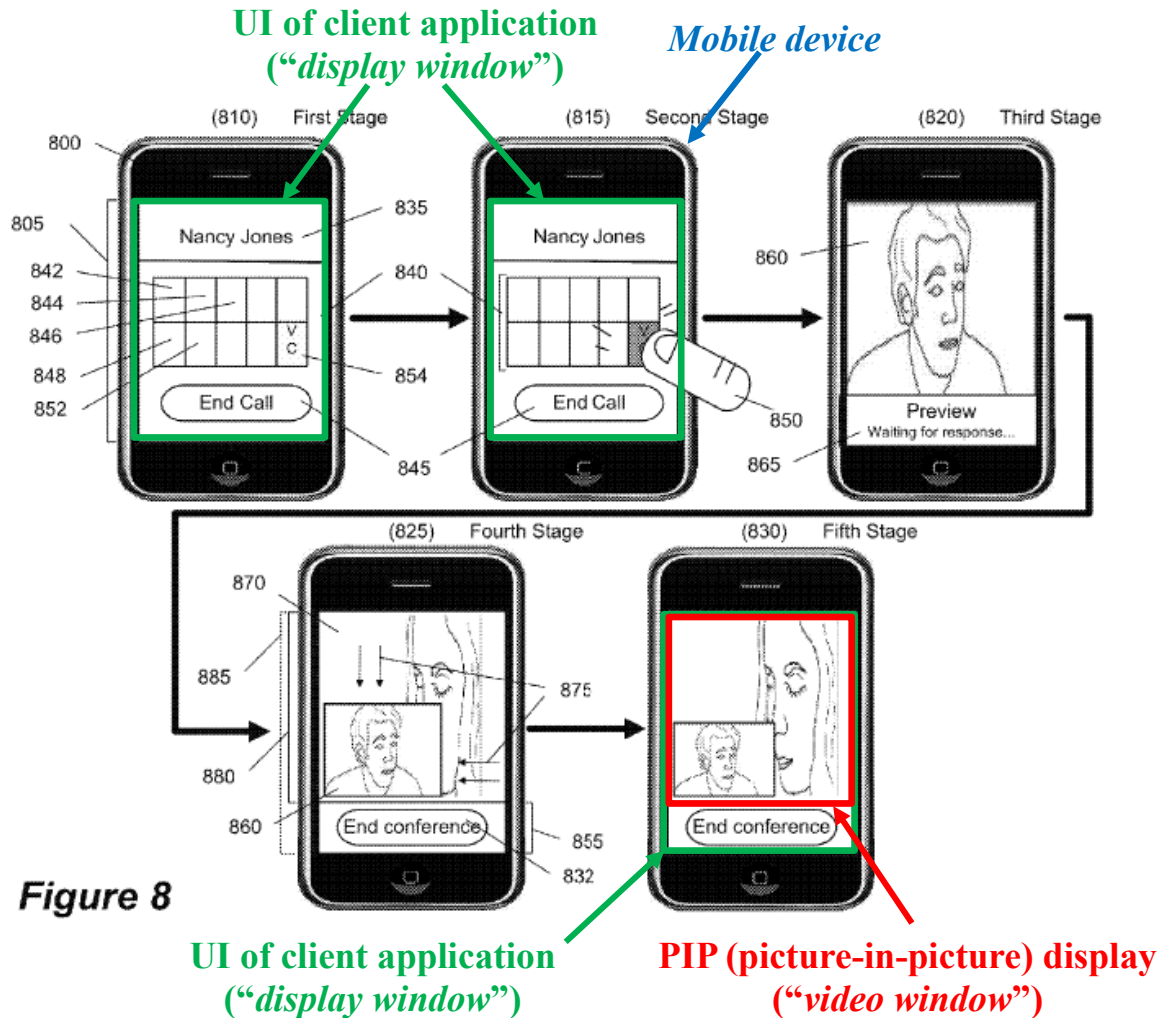


Figure 8

Ex.1005, Fig. 8 (annotated).

125. Abuan discloses that the user interface 805 (shown transitioning through five stages UI 810 to UI 830 in Figure 8 above) belongs to the client application 1365, explaining that the “video conference client 710 of some embodiments receives the request to start the video conference when the user of the device 705 selects a user interface (UI) item of a user interface displayed on the device 705” and that “such user interfaces are illustrated in FIG. 8,” which shows “five operational stages 810, 815, 820, 825, and 830 of a user interface (“UI”)

805.” Ex.1005, [0110], [0117]. In other words, the UI 805 (and UI 810 to UI 835) in Figure 8 of Abuan are UIs of the video conference client 710. As discussed at [1.7.1], the video conference client 710 is the same as the client application 1365. Abuan also teaches that “[t]he client application 1365 ... generates user interfaces that are displayed on the dual camera mobile device,” rendering obvious that the UI 805 of the device 800 displaying the client application 1365 is the UI of the client application 1365. Ex.1005, [0172].

126. For example, as illustrated in Figure 8 above, the “user interface (“UI”) 805 of the device 800” displays the phone call application from which the video conference is activated (“first stage 810” and “second stage 815”). Ex.1005, [0117]; *see also* [0118] (“the UI 805 includes a name field 835” “display[ing] the name of the person on the other end of the phone call” and “a selectable UI item 845” that “provides a selectable End Call option for the user to end the phone call”).

127. Accordingly, Abuan teaches that when the audio/video conference call extending from the client application 1365 (part of “*the superblock application*”) on device 705 to device 720 is established as discussed at [1.7.1] and [1.7.2], the video conference manager 1304 of the video conference module 1302 (part of “*the function block*”) on device 705 outputs a PIP display (“*a video window*”) of videos from the audio/video conference call (“*video output for the*

A/V call”) between devices 705 and 720 to be displayed on a user interface 805 of the client application 1365 (e.g., the user interface of the phone application) on device 705 (“*within a display window of the superblock application on the mobile device*”), rendering obvious “*providing, by the function block, video output for the A/V call to a video window within a display window of the superblock application on the mobile device.*”

- k. **[1.8.2] *wherein the video window is displayed simultaneously with the display window of the superblock application on a screen of the mobile device; and***

128. As discussed at [1.8.1], a PIP display (“*the video window*”) of videos from the audio/video conference call between devices 705 and 720 is displayed in a user interface 805 of the client application 1365 (e.g., the phone application) on device 705 (“*is displayed*” in “*the display window of the superblock application on ... the mobile device*”). A POSITA would have understood that the user interface 805 is displayed on a screen of the device 705 (“*a screen of the mobile device*”), as Abuan discloses that “[t]he mobile device ... has a display screen for displaying the captured picture images and video images,” such as “two videos ... captured by two cameras of two different devices that are in a video conference.” Ex.1005, [0003], [0035].

129. Abuan discloses that, upon establishment of the audio/video conference call between devices 705 and 720, the user interface 805 (“*the display*

window”) of the client application 1365 (e.g., the phone application), which corresponds to “the entire display area 885” of the device 705, contains the PIP display 880 (“*the video window*”) and “a lower display area 855 (e.g., a tool bar or a menu bar)” having a “selectable UI item” 832/985 such as a “mute button,” an “end conference button,” and a “switch camera button.” Ex.1005, FIG. 8 (“first stage 810” and “fourth stage 825”), [0125], [0138], FIG. 9 (“fifth stage 930”). That is, Abuan discloses that the user interface 805 (“*the display window*”) of the client application 1365 displays within it the PIP display 880 (“*the video window*”), along with a display area 855 of the user interface 805 (“*the display window*”) of the client application 1365 having selectable UI items for controlling aspects of the audio/video conference call, rendering obvious “*the video window is displayed simultaneously with the display window.*” Ex.1005, [0140].

I. [1.9] *providing, by the function block, audio for the A/V call via the mobile device.*

130. As discussed at [1.7.1], the audio/video conference call from device 705 to device 720 teaches “*the A/V call.*”

131. Abuan discloses an “audio processing manager 1415 [that] processes audio data captured by the dual camera mobile device for transmission to the other device in the video conference. For instance, the audio processing manager 1415 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

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

A POSITA would understand that the speaker 1430 is a hardware component of the dual camera mobile device. *See, e.g.*, Figure 19 showing “a dual camera mobile computing device architecture 1900” having “audio subsystems 1930 [that] is coupled to a speaker” ([0249], [0251]).

132. Accordingly, Abuan discloses that the audio processing manager 1415 (part of “*the function block*,” as discussed at [1.4]) outputs audio of the audio/video conference call (“*providing ... audio for the A/V call*”) via the speaker of mobile device 705 (“*via the mobile device*”).

5. Claim 2

a. [2.0] *The device of claim 1 wherein the instructions further include*

133. See limitations [1.0] and [1.3].

b. [2.1] *determining, by the function block, whether the function block has a capability to provide the A/V call.*

134. Abuan at least renders obvious limitation [2.1] 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.

135. 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].

136. 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].

137. 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 powered on and as such 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 A/V call*”), rendering obvious “*determining, by the function block, whether the function block has a capability to provide the A/V call.*”

6. Claim 3

a. [3.0] *The device of claim 2 wherein the instructions further include*

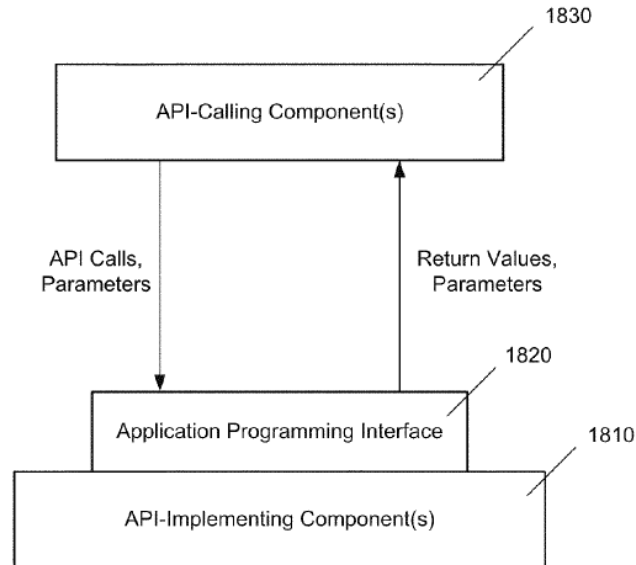
138. Claim limitation [3.0] is substantially the same as [2.0] and would have been obvious for the reasons discussed above.

b. [3.1] *determining, by the function block, whether the function block has a capability to render the A/V call.*

139. Abuan at least renders obvious limitation [3.1] because Abuan describes various software modules (which correspond to “*the function block*”) that determine their own output capabilities based at least in part on the output capabilities of output devices such as the speaker and display.

140. As discussed at [1.7.2], Abuan discloses “[o]ne or more Application Programming Interfaces (APIs)” such as “[a]n API” “implemented by a program code component or hardware component (hereinafter ‘API-implementing component’) that allows a different program code component or hardware component (hereinafter ‘API-calling component’) to access and use one or more functions, methods, procedures, data structures, classes, and/or other services provided by the API-implementing component.” Ex.1005, [0237]-[0238]. An example of an “API-implementing component 1810” includes “an operating system, a library, a device driver, an API, an application program, software or other module[] that implements the API 1820.” Ex.1005, [0246]. An example of an “API-calling component 1830” includes “an operating system, a library, a device driver, an API, an application program, software or other module[that], makes API calls through the API 1820 to access and use the features of the API-implementing component 1810 that are specified by the API 1820.” Ex.1005, [0246]. Figure 18

of Abuan below shows an API architecture including an API-implementing component 1810, an API 1820, and an API-calling component 1830.



Ex.1005, Fig. 18

141. Abuan teaches that “an 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]. 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. *See* 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”).

142. Thus, with Abuan’s description of the client hardware device including common output devices such as “speaker 1430” and “a display,” 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, thereby rendering obvious “*determining, by the function block, whether the function block has a capability to render the A/V call.*” Ex.1005, [0184], [0252].

143. For example, it would have been obvious to a POSITA that the audio driver 1420 is an API-implementing component 1810 and the audio processing manager 1415, which “is implemented as a separate software module ... [or] is implemented as part of the media exchange module 1320,” is an API-calling component 1830. Ex.1005, [0183], [0246] (“the API-implementing component 1810 (e.g., ... a device driver...)” and “The API-calling component 1830 (e.g., ... module)”). Abuan discloses that “the audio driver 1420” provides “audio data ...

captured by microphone 1425” to the “audio processing manager 1415” and “output[s] ... [decoded audio data] to the speaker 1430”. Ex.1005, [0184].

Accordingly, it would have been obvious to a POSITA that, for the audio processing manager 1415 to output audio of the audio/video conference call via the speaker of the mobile device, as discussed at [1.9], the audio processing manager 1415 (i.e., the API-calling component 1830) (part of “*the function block*”) makes an API call to the audio driver 1420 (i.e., the API-implementing component 1810), to determine the “output capabilities and state” of the speaker of the mobile device, rendering obvious “*determining, by the function block, whether the function block has a capability to render the A/V call.*”

144. As another example, Abuan describes the video conference manager 1304 “scal[ing] and composit[ing] the images [into PIP or other composite views] before displaying the images on the dual camera mobile device.” Ex.1005, [0176]. The display of the mobile device can be a touchscreen that is coupled to a touchscreen controller 1955. Ex.1005, FIG. 19 (“touch screen 1965”), [0252] (“input/output peripheral devices, such as a display, a touch screen, etc.” and “Touch-screen controller 1955 can be coupled to the touch screen 1965”). It would have been obvious to a POSITA for the video conference manager 1304, which is part of the video conference module 1302 (part of “*the function block*”), to determine the “output capabilities and state” of the display or touch screen (e.g., by

performing an API call to the touch screen controller 1955) before scaling and compositing the images into a PIP display for displaying at the touch screen or display of the local mobile device. Ex.1005, [0243].

145. The above analysis is consistent with the '032 patent's description that the function block's capability to render an audio corresponds to the function block "checking ... to see whether there is an available microphone (mic) input and an available speaker output" and the function block's capability to render a video corresponds to the function block "checking ... to see whether there is an available screen" for displaying the video:

the function block 200 may determine whether (1) the function block 200 has a particular capability, (2) whether the function block 200 can render that capability, and (3) whether the superblock 104 can render that capability. For example, the function block 200 may determine that the function block 200 has the capability to capture audio and play that captured audio through a speaker.

Next, the function block 200 may determine whether it can render that capability by checking, for example, to see whether there is an available microphone (mic) input and an available speaker output. If one or both of the mic and speaker are not present, then the function block 200 has the capability to handle the audio, but not the capability to render the audio.

...

In another example, the superblock 104 may desire to display video. Again, the function block 200 may determine that the function block 200 has the capability to handle the video. Next, the function block 200 may determine whether it can render that capability by checking, for example, to see whether there is an available screen. If a screen is not present, then the function block 200 has the capability to handle the video, but not the capability to render the video

Ex.1001, 10:9-37.

7. Claim 4

a. [4.0] *The device of claim 3 wherein the instructions further include*

146. Claim limitation [4.0] is substantially the same as [3.0] and would have been obvious for the reasons discussed above.

b. [4.1] *determining, by the function block, whether the superblock has a capability to render the A/V call.*

147. As discussed at [3.1], a POSITA would have found it obvious that 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 and state" of output devices such as the speaker and display. Ex.1005, [0243]. It would have been obvious to a POSITA that where a client device lacks a speaker, neither the *function block* nor the *superblock* would have a capability of

rendering the audio portion of a videoconference. Similarly, where a client device lacks a display, neither the *function block* nor the *superblock* would have a capability of rendering the video portion of a videoconference. Thus, Abuan's description of determining the output capabilities of a client device, as discussed at [3.1], renders this limitation obvious.

148. Additionally, the '032 patent specification notes that the *superblock application's* "capability to render" a video corresponds to "whether there is an available screen" and whether the *superblock application* is able to "provide a video window to render the video" and "place [] video data in a video window" on the screen. Ex.1001, 10:23-25, 30-42. Thus, it would be obvious to a POSITA that when the *function block* determines the output capabilities and state of the display, the *function block* also determines the availability of the display and as such determines *whether the superblock has a capability to render the video of the A/V call*.

8. Claim 5

a. [5.0] *The device of claim 1 wherein the instructions further include*

149. See limitations [1.0] and [1.3].

- b. **[5.1] *determining, by the function block, that the resource is currently available prior to establishing the A/V call.***

150. As discussed at [1.7.1], Abuan at least renders obvious the video conference module 1302 establishing an audio/video conference call (“*the A/V call*”) between the video conference client 710 running on mobile device 705 and video conference client 725 running on mobile device 720 via the video conference server 715. Abuan discloses that “the video conference module 1302 includes ... a networking manager 1314,” which in turn includes a “session negotiating manager 1705 [that] is responsible for establishing connections between the dual camera mobile device and one or more remote devices participating in the video conference.” Ex.1005, [0174], [0208]-[0209], [0211]. “The session negotiating manager 1705 includes a protocol manager 1710” that “support[s] a number of communication protocols, such as a real-time transport protocol (RTP), a transmission control protocol (TCP), a user datagram protocol (UDP), and a hypertext transfer protocol (HTTP), among others,” including “session initiation protocol (SIP).” Ex.1005, [0210]-[0211]. A POSITA would have understood that the video conference module 1302 establishes the audio/video conference call from mobile device 705 to mobile device 720 via the video conference server 715 using any of the recited number of communication protocols.

151. A POSITA would have been familiar with TCP and SIP, including that SIP itself uses TCP. Ex.1048, 142 (“All SIP elements MUST implement UDP and TCP.”). It was well-known before the priority date of the ’032 patent that establishing a communication session between a client and a server according to the TCP protocol comprises the client sending a SYN message to the server and in return receiving an acknowledgment ACK message. Ex.1047, [0023] (“As is well-known to those skilled in networking technology, a TCP session between a client 10 and a server 12 is set up by a client sending a synchronisation SYN packet to the server 12 and waiting for an acknowledgement ACK packet in return from the server 12”). It would have been obvious to a POSITA that if the videoconference server fails to return the acknowledgment ACK message, the videoconference server is not available; otherwise (the videoconference server returns the acknowledgment ACK message), the videoconference server is available.

152. Accordingly, it would have been obvious to a POSITA that the video conference module 1302 would send a TCP SYN message to, and receive a TCP ACK message from, the video conference server 715 in order to establish the audio/video conference call from mobile device 705 to mobile device 720 via the video conference server 715, i.e., the video conference module 1302 would determine that the video conference server 715 is available for establishing the audio/video conference call before establishing the audio/video conference call,

rendering obvious “*determining, by the function block, that the resource is currently available prior to establishing the A/V call.*”

9. Claim 6

- a. [6.0] *The device of claim 1 wherein the instructions further include***

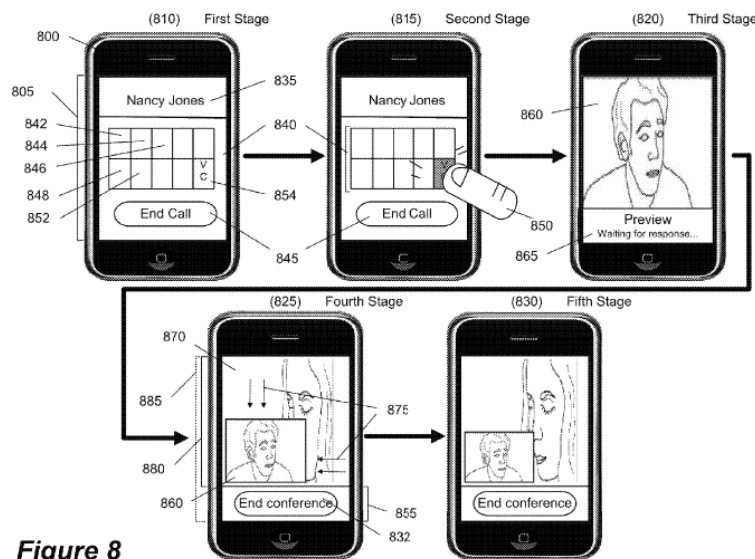
153. See limitations [1.0] and [1.3].

- b. [6.1] *initializing the function block, by the superblock application, when the superblock application is loaded into the memory and executed.***

154. As discussed at [2.0], Abuan discloses “a process 1500 performed by a video conference manager” 1304 of the video conference module 1302 where the video conference manager 1304 receives “instructions to start a video conference” “from the client application 1365” (part of “*superblock application*”) or “from a user through a user interface displayed on the dual camera mobile device and forwarded to the video conference manager 1304 by the client application 1365.” Ex.1005, [0186]-[0187]. “The video conference client 710,” which is the same as the client application 1365 as discussed at [1.7.1], “receives the request to start the video conference when the user of the device 705 selects a user interface (UI) item of a user interface displayed on the device 705. Examples of such user interfaces are illustrated in FIG. 8.” Ex.1005, [0110].

155. Figure 8 of Abuan shows a “second stage 815 [that] displays the UI 805 after the user selects the selectable Video Conference option 854 (e.g., through

a single finger tap by finger 850) to activate a video conference tool. In this example, the Video Conference option 854 (which can be implemented as a selectable icon) allows the user to start a video conference during the phone call. In the second stage, the Video Conference option 854 is highlighted to indicate that the video conference tool has been activated.” Ex.1005, [0119].



Ex.1005, Fig. 8

156. Abuan discloses that when the instructions to start a video conference are received by the video conference manager 1304 from the client application 1365 (part of “*the superblock application*”), “module[s] that interact[] with the video conference manager 1304,” such as “the image processing manager 1308, the audio processing manager 1415, and the networking manager 1314” (each part of “*function block*,” as discussed at [1.4]) are initialized. Ex.1005, [0188]. In other words, when a video conference tool of the client application 1365 (part of

“*superblock application*”) is activated, “the image processing manager 1308, the audio processing manager 1415, and the networking manager” (each part of “*the function block*”) are initialized, teaching “*initializing the function block, by the superblock application, when the superblock application is*” activated.

157. Abuan at least renders obvious that the client application 1365 is “*loaded into the memory and executed*” when it is activated. First, Abuan explains that its applications are implemented as instructions “recorded on a computer readable storage medium (also referred to as computer readable medium),” which “includes magnetic disks, optical disks, random access memory; read only memory, flash memory devices, etc.,” and that “applications stored in magnetic storage [] can be **read into memory** for processing by a processor,” rendering obvious the client application 1365 being “*loaded into the memory.*” Ex.1005, [0233]-[0234], [0248]. Second, Abuan discloses that the instructions are “**executed by one or more processing unit(s)** (e.g., one or more processors, cores of processors, or other processing units)” to “cause the processing unit(s) to perform the actions indicated in the instructions,” teaching “*executed.*” Ex.1005, [0233]. It would have been obvious to a POSITA that activating the client application 1365 corresponds to executing instructions, stored in a computer readable medium and implementing the client application 1365, to send the video conference manager 1304 instructions to start an audio/video conference call, thereby “*initializing the*

function block,” i.e., Abuan renders obvious “when the superblock application is loaded into the memory and executed.”

10. Claim 7

a. [7.0] *The device of claim 1 wherein*

158. Claim limitation [7.0] would have been obvious for the reasons discussed above in [1.0].

b. [7.1] *the request is an application programming interface (API) call from the superblock application to the function block.*

159. As discussed at [1.4], the instructions sent by the client application 1365 to the video conference module 1302 to start an audio/video conference call teach “*a request.*”

160. Abuan discloses that “an API can be used ... to initiate performance of an operation ... provided by the API-implementing component. By way of example, the API-implementing component and the API-calling component may each be any one of an operating system, a library, a device driver, an API, an application program, or other module.” Ex.1005, [0243]. “[A]n API-calling component ... passes data and control information using parameters via the API calls or invocations. The API-implementing component may return a value through the API in response to an API call from an API-calling component.” Ex.1005, [0241].

161. A POSITA would have understood that the client application 1365 is an example of an API-calling application program, the video conference module 1302 is an example of an API-implementing “module,” and the instructions (“*the request*”) from the client application 1365 to the video conference module 1302 are API calls or invocations (“*an application programming interface (API) call*”) from the client application 1365 (“*from the superblock application*”) to the video conference module 1302 (part of “*the function block*”) “to initiate performance of an operation,” i.e. to start a conference. Ex.1005, [0243].

11. Claim 9

a. [9.0] *A mobile device comprising:*

162. Claim limitation [9.0] is the same as [1.0] and would have been obvious for the reasons discussed above.

b. [9.1] *a network interface:*

163. Claim limitation [9.1] is the same as [1.1] and would have been obvious for the reasons discussed above.

c. [9.2] *a processor coupled to the network interface; and*

164. Claim limitation [9.2] is the same as [1.2] and would have been obvious for the reasons discussed above.

- d. [9.3] *a memory coupled to the processor and configured to store a plurality of instructions executable by the processor, the instructions including instructions for*

165. Claim limitation [9.3] is the same as [1.3] and would have been obvious for the reasons discussed above.

- e. [9.4] *a superblock application having instructions for a function block included therein,*

166. As discussed at [1.4], Abuan's video conferencing and processing module 1300 teaches "*a superblock application*," and the video conferencing and processing module 1300 excluding the client application 1365 teaches "*a function block*." A POSITA would have known that modules were implemented as a set of instructions. Ex.1041, 703-704 (A module is defined as "[a] program unit that is discrete and identifiable with respect to compiling, combining with other units and loading" or "a logically separable part of a program"), 703-704 (A computer program is defined as "[a] prepared sequence of instructions to the system to accomplish a defined task"). Accordingly, a POSITA would have understood that the video conferencing and processing module 1300 ("*a superblock application*") *included therein instructions for* the video conferencing and processing module 1300 excluding the client application 1365 ("*a function block*").

167. Further, Abuan describes "the client application 1365 [part of "*the superblock application*"] may be ... implemented as a stand-alone application." Ex.1005, [0105], [0171]. A POSITA would have understood Abuan's reference to

a “stand-alone application” to mean that all the functionality of the application, including its module subcomponents (e.g., the *function block* functionality) is contained in a single, unitary application. *See, e.g.*, Ex.1041, 1094 (“stand-alone” defined as “pertaining to a system that is self-contained and not connected to other systems or system component”). Abuan notes that the computer program includes a variety of code, including API and files that are compiled into the computer program:

The computer-readable media may store a computer program that is executable by at least one processing unit and includes sets of instructions for performing various operations. **Examples of computer programs or computer code include machine code, such as is produced by a compiler,** and files including higher-level code that are executed by a computer, an electronic component, or a microprocessor using an interpreter.

Abuan, [0262]; *see also* [0239] (“An API can be specified in terms of a programming language that can be interpreted or **compiled** when an application is built”).

168. Compiling the various modules, including user-facing (e.g., UI) modules for a client application, and functional modules performing operations requested by the client application, was a commonly-used and well-understood way to implement and deploy applications on electronic devices. *See* Ex.1012,

1:26-29 (“Traditionally, an application's source files are compiled in object modules and then linked together with other object modules, generically called libraries, to form a complete stand-alone application.”). Thus, it would have been obvious to a POSITA for the client application to have incorporated into itself the instructions corresponding to its module subcomponents, rendering obvious “*the superblock application having instructions for a function block included therein.*”

f. [9.5.1] *wherein the function block is configured to provide a plurality of functions*

169. The '032 specification describes the “function block 200 [as] includ[ing] instructions for providing the superblock 104 with one or more functions (e.g., capabilities) Such functions ... may include ... audio, video, ... conferencing, ... and/or other functions.” Ex.1001, 3:53-61.

170. As discussed at [1.4], Abuan’s video conference module 1302, media exchange module 1320, and audio processing manager 1415 constitute “*the function block.*” “The video conference module ... handles various video conferencing functions.” Ex.1005, [0101]. “[T]he media exchange module ... provide[s] ... access[to] various video processing and encoding functionalities.” Ex.1005, [0237]. “The audio processing manager 1415 processes audio data.” Ex.1005, [0184]. Thus, Abuan’s video conference module 1302, media exchange module 1320, and audio processing manager 1415 (each part of “*the function*”

block”) together provide audio, video, and/or conferencing functions (“*a plurality of functions*”), teaching limitation [9.5.1].

- g. **[9.5.2] *that are accessible to the superblock application via an application programming interface (API) of the function block,***

171. As discussed at [1.4], the client application 1365 constitutes “*a superblock application*” and Abuan’s video conference module 1302, media exchange module 1320, and audio processing manager 1415 constitute “*the function block.*” Abuan discloses that “the audio processing manager 1415 is implemented as a separate software module ... [or] is implemented as part of the media exchange module 1320.” Ex.1005, [0183].

172. As discussed at [1.7.2], Abuan discloses “[o]ne or more Application Programming Interfaces (APIs)” such as “[a]n API” “implemented by a program code component or hardware component (hereinafter ‘API-implementing component’) that allows a different program code component or hardware component (hereinafter ‘API-calling component’) to access and use one or more functions, methods, procedures, data structures, classes, and/or other services provided by the API-implementing component.” Ex.1005, [0237]-[0238].

173. An example of an “API-implementing component 1810” includes “... [a] module[] that implements the API 1820.” Ex.1005, [0246]. It would have been obvious to a POSITA that the video conference module 1302, media exchange

module 1320, and audio processing manager 1415 (which is a separate module or part of the media exchange module) (each part of “*the function block*”) are API-implementing components.

174. An example of an “API-calling component 1830” includes “an application program... [that] makes API calls through the API 1820 to access and use the features of the API-implementing component 1810 that are specified by the API 1820.” Ex.1005, [0246]. It would have been obvious to a POSITA that the client application 1365 (part of “*the superblock application*”) is an API-calling component.

175. Accordingly, a POSITA would have found it obvious that the API-implementing components, i.e., the video conference module 1302, media exchange module 1320, and audio processing manager 1415 (each part of “*the function block*”) provide APIs that allows the API-calling component, i.e., the client application 1365 (part of “*the superblock application*”) “to access and use one or more functions” such as the audio, video, and/or conferencing functions (“*a plurality of functions*”) of *the function block*, rendering obvious “[*a plurality of functions of the function block*] that are accessible to the superblock application via an application programming interface (API) of the function block.” Ex.1005, [0238].

h. [9.6.1] *wherein the functions are provided within the superblock application itself and*

176. As discussed at [1.4], Abuan's video conferencing and processing module 1300 teaches "*a superblock application,*" and the video conferencing and processing module 1300 excluding the client application 1365, which includes the video conference module 1302, media exchange module 1320, and audio processing manager 1415, teaches "*a function block.*" Accordingly, it would have been obvious to a POSITA that the audio, video, and/or conferencing functions ("*the functions*") of the video conference module 1302, media exchange module 1320, and audio processing manager 1415 (each "*the function block*") are included within the video conferencing and processing module 1300 ("*are provided within the superblock application itself*").

177. Furthermore, as discussed at [9.4], Abuan describes "the client application 1365 may be ... implemented as a stand-alone application," rendering obvious that the videoconferencing module and other modules such as the video conference module 1302, media exchange module 1320, and audio processing manager 1415 (each "*the function block*") are *provided within* the client application 1365 (part of "*the superblock application*") *itself*. Ex.1005, [0105], [0171].

- i. **[9.6.2] *[wherein the functions] are accessible within the superblock application without switching context to another application on the mobile device, and***

178. The '032 specification describes “switch[ing] contexts” as “switch[ing] between applications.” Ex.1001, 2:36-37; *see also* 3:6-18, 30-36.

179. Abuan’s client application is a “stand-alone application” that employs application programming interfaces (APIs) to invoke functionality provided by the video conferencing module and other modules. Ex.1005, [0171]; *also see generally*, Ex.1005, [0236]-[0248]. Abuan does not reference making any context switch as part of the API-calling process. Thus, it would have been obvious to a POSITA from Abuan’s silence about context switching that there is no context switch performed when an API-calling component (such as Abuan’s client application) invokes a function or feature of an API-implementing component (such as Abuan’s video conference module).

180. This is consistent with how a POSITA would have understood an application to execute; that is, a POSITA would have understood that all the code for a stand-alone application would execute within the same memory context. Ex.1042, 317 (“We first need to make sure that each process has a separate memory space. To do this, we need the ability to determine the range of legal addresses that the process may access and to ensure that the process can access only these legal addresses.”), 321 (“When a routine needs to call another routine,...

the relocatable linking loader is called to load the desired routine into memory and to update the program's address tables to reflect this change..."). For example, "stand-alone" is defined as "pertaining to a system that is self-contained and not connected to other systems or system component." Ex.1041, 1094. As such, a POSITA would have understood that, without connection to other components, there would be no other context to switch to and all the code for a stand-alone application would execute within a single memory context.

181. For example, as discussed at [1.7.2], Abuan teaches that, when a user selects a "Video Conference option" on a UI of a client application such as a phone application or a contacts application (part of "*superblock application*") on a mobile device, a video conference tool (i.e., the video conference module 1302 (part of "*the function block*") is activated to provide video conferencing functions (part of "*a plurality of functions*," as discussed at [9.6.1]) and establish an audio/video conference call between the device and another mobile device. As discussed at [1.8.1], Abuan discloses that, when the audio/video conference call is established, the video conference manager 1304 of the video conference module 1302 (part of "*the function block*") outputs a PIP display of the videos from the audio/video conference call to a user interface of the client application 1365 on the mobile device. In other words, the PIP display is displayed without the client application 1365 being switched with any other application on the mobile device.

182. In general, Abuan has no description of any context switch occurring during the establishment of the audio/video conference call. Further, as mentioned above, context switching is not mentioned in Abuan's extensive description of APIs and API-calling procedures for invoking program code and functional modules. *See generally*, Abuan, [0236]-[0248]. Figures 8-11 of Abuan show that the client application continues to run, with its user interface (UI) 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, e.g., [0118]-[0128] (describing each screen of Fig. 8's five stages as "the UI 805").

183. At no point does Abuan disclose that, in any of the sequences shown in Figs. 8-11, the mobile device or video conferencing application switches context to *another application* on the mobile device. Rather, Abuan's disclosure contemplates that the sequences involved in establishing the video conference, and participating in the video context, are performed with the client application's context. 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 UI. *See, e.g.*, Ex.1005, [0187], [0115].

184. Accordingly, Abuan teaches that a user activates a video conference module 1302 (part of “*the function block*”) that provides video conferencing functions (part of “*a plurality of functions*”) from a client application 1365 (“*superblock application*”) on a mobile device, where the activation of the video conference module 1302 launches an audio/video conference call and displays a PIP display thereof without switching away from the client application 1365 to another application on the mobile device, rendering obvious “[*wherein the functions of the function block*] ... *are accessible within the superblock application without switching context to another application on the mobile device.*”

- j. [9.7] *wherein the superblock application cannot access at least one of the functions of the function block without providing the function block with an authorization code, and***

185. As discussed at [1.5], the combination of Abuan and Lawson renders it obvious that the client application 1365 sends to the video conference module 1302 i) instructions to start an audio/video conference call, and ii) an authorization token authorizing the client application 1365 to make and/or receive the audio/video conference call using the video conference module 1302, i.e., using the video conferencing functions of the video conference module 1302. A POSITA would have understood that the client application 1365 (part of “*the superblock application*”) was not authorized to access the video conferencing functions (“*a plurality of functions*”) of the video conference module 1302 (part of “*the function*

block”) without providing the authorization token (“*an authorization code*”) to the video conference module 1302 (part of “*the function block*”), rendering obvious “*the superblock application cannot access at least one of the functions of the function block without providing the function block with an authorization code*”.

- k. [9.8] *wherein the instructions further include providing, by the function block, video data to the superblock application for rendering by the superblock application,*

186. Abuan at least renders obvious that *the function block* processes and provides video images obtained from the mobile devices in the video conference for display on a user interface of a client application on one of the mobile devices.

187. Specifically, Abuan discloses that a video conference manager 1304 of the video conference module 1302 “processes images received from one or more devices in the video conference and images captured by one of both cameras of the dual camera mobile device for display on the dual camera mobile device.” Ex.1005, [0176]. The “images received from the other device of the video conference” are decoded by the decoder 1360 and “are sent back to the video conference module 1302 through the decoder driver 1340 and the video decompressor module 1326.” Ex.1005, [0182]. That is, the decoder 1360 provides the images from the other device to the decoder driver 1340 and the video decompressor module 1326 (each part of “*the function block*”), which in turn provides the images (part of “*video data*”) to the video conference module 1302

(part of “*the superblock application*”), rendering obvious “*providing, by the function block, video data to the superblock application.*”

188. The “images captured by the dual camera mobile device,” such as “an image of a person’s face,” are retrieved from the buffer 1306. Ex.1005, [0176], [0204]. The video conference manager 1304 “scales and composites the images [as the PIP or other composite views] before displaying the images on the dual camera mobile device.” Ex.1005, [0176]. A frame buffer 1325 “stores images of a video for display on a display 1345 of the dual camera mobile device.” Ex.1005, [0170]. That is, the images from the local mobile device and from the remote device in the video conference are scaled and composited and provided to the frame buffer 1325 (part of “*the superblock application*”) to be displayed by the local mobile device, rendering obvious “*for rendering by the superblock application.*”

189. Further, as discussed at [1.8.1], the PIP display of videos from a video conference is displayed on a user interface of the client application (part of “*the superblock application*”). Thus, the video conference manager 1304 (part of “*the function block*”) providing locally-captured images (part of “*video data*”) as well as images (part of “*video data*”) from remote devices on a video conference for display, on a user interface of the client application (part of “*the superblock application*”), as a PIP display, renders obvious “*providing, by the function block,*

video data to the superblock application for rendering by the superblock application.”

190. Additionally, as discussed at [1.8.1], Abuan teaches that, when an audio/video conference call is established between a device 705 and another device 720, the video conference manager 1304 on device 705 generates and outputs a PIP display of videos from the audio/video conference call to be displayed at a user interface 805 of a client application 1365 on the device 705. It would have been obvious to a POSITA that the video conference manager 1304 of the video conference module 1302 (part of “*the function block*”) provides the videos (“*video data*”) of the PIP display to the client application 1365 (“*the superblock application*”) so that the client application 1365 displays the PIP display of the videos (“*video data*”) on a user interface 805 of the client application 1365, rendering obvious “*providing, by the function block, video data to the superblock application for rendering by the superblock application.*”

I. [9.9] wherein the superblock application renders the video data in a video window that is superimposed on a display of the superblock application, and

191. As discussed at [9.8], Abuan renders it obvious that the video conference manager 1304 of the video conference module 1302 provides the videos of the PIP display to the client application 1365 so that the client application 1365 displays the videos (“*video data*”) in a PIP display (“*the video window*”) on a user

interface 805 of the client application 1365 (*“a display of the superblock application”*).

192. Further, as discussed at [1.8.1] and [1.8.2], Abuan illustrates an example of a “picture-in-picture” display, which contains a composite of a user’s local video and another conference participant’s remote video, *“superimposed”* on the client application’s user interface (*“on a display of the superblock application”*). Ex.1005, Figs. 8-11, [0126], [0136], [0148]. For example, Abuan renders it obvious that the PIP display is displayed within the user interface 805 of the client application 1365, along with a display area 855 of the user interface 805 having selectable UI items for controlling aspects of the audio/video conference call. The PIP display *“can include various combinations of images from the cameras of the local dual camera mobile device and images from cameras of the other device participating in the video conference.”* Ex.1005, [0190]. For instance, the user may *“flip the order of videos in a [picture-in-picture] display,”* swapping positions of the foreground inset display and the background main display. Ex.1005, [0040].

193. Accordingly, Abuan renders it obvious that the client application 1365 displays (*“the superblock application renders”*) the videos in a PIP display (*“the video data in a video window”*) that in turn is displayed within (*“that is superimposed on”*) a portion of the user interface 805 (*“a display of the superblock application”*), i.e., the portion excluding the display area 855.

- m. **[9.10] *wherein both the video window and at least a portion of the display of the superblock application are simultaneously viewable.***

194. As discussed at [1.8.2], Abuan renders it obvious that the PIP display (“*the video window*”) is displayed within (“*viewable*”) the user interface 805 of the client application 1365, along with (“*simultaneously*”) a display area 855 of the user interface 805 of the client application 1365 (“*at least a portion of the display of the superblock application*”), rendering obvious “*both the video window and at least a portion of the display of the superblock application are simultaneously viewable.*”

195. Additionally, as discussed at [1.8.1], Abuan illustrates an example videoconferencing user interface that with both a user’s local video (“*the video window*”) and a remote user’s video (“*a portion of the display of the superblock application*”) shown as “*simultaneously viewable.*” Ex.1005, Figs. 8-11, [0126], [0136], [0148]. Alternatively, the remote user’s video also corresponds to the “*video window*,” while the user local’s video corresponds to “*a portion of the display of the superblock application.*”

12. Claim 10

- a. **[10.0] *The mobile device of claim 9 wherein the plurality of functions includes***

196. See limitations [9.0] and [9.5.1].

b. [10.1] *access to an external service not present on the mobile device.*

197. As discussed at [9.5.1], Abuan's video conference module, media exchange module, and audio processing manager (each part of "*the function block*") provide audio, video, and/or conferencing functions ("*the plurality of functions*"). The "video conferencing functions ... [include] networking" and "the network layer 640 [of the video conference module] performs ... all of the networking functionalities for video conferencing," such as "establish[ing] a network connection [] between the dual camera mobile device and a remote device of a video conference." Ex.1005, [0101], [0104], [0102] ("The video conference module 625 includes ... a network layer 640").

198. For example, as discussed at [1.7.1], to establish an audio/video conference call between a device 705 and a device 720, the video conference module 1302 on device 705 performs (via its network layer) all the communications with the video conference server 715, which facilitates the establishment of the audio/video conference call by "rout[ing] messages" between the devices, potentially using "push technology." Ex.1005, [0109], [0111]. For instance, as shown in Figure 7 of Abuan below, device 705 "send[s] (at operation 2) a video conference request, which indicates the device 720 as the recipient" to the video conference server 715 and in return receives "(at operation 6) the video conference acceptance," thereby leading to the "establish[ment] (at operation 7)

[of] a video conference between the device 705 and the device 720.” Ex.1005, [0111], [0113], [0114]. As discussed at [1.7.2], a POSITA would have understood video conference server 715 to be a server that is separate from mobile device 705.

“an external service not present on the mobile device”

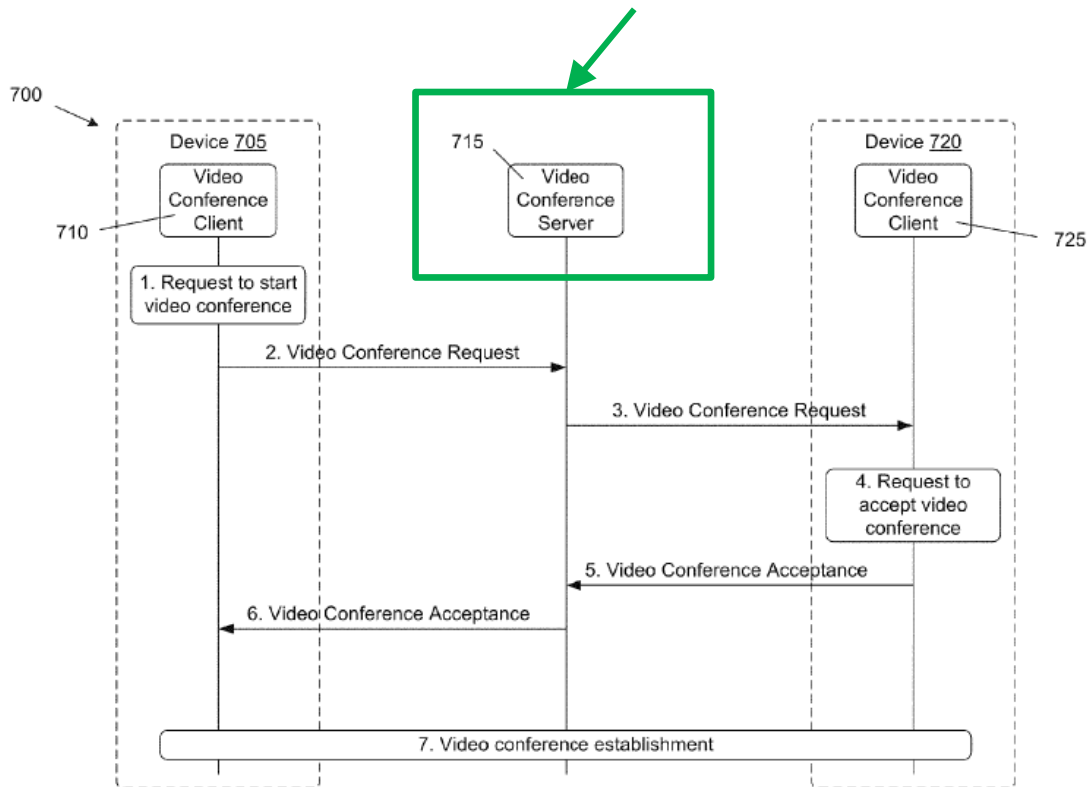


Figure 7

Ex.1005, Fig. 7 (annotated).

199. Accordingly, Abuan teaches that the audio, video, and/or conferencing functions (*“the plurality of functions”*) provided by the video conference module (part of *“the function block”*) on mobile device 705 include communications to and from a separate video conference server 715 (*“access to external service not present on the mobile device”*) to establish an audio/video

conference call with another device, i.e., device 720, rendering obvious “*the plurality of functions includes access to an external service not present on the mobile device.*”

13. Claim 11

a. [11.0] *The mobile device of claim 10 wherein the instructions further include*

200. Claim limitation [11.0] is substantially the same as [10.0] and would have been obvious for the reasons discussed above.

b. [11.1] *providing an event notification by the function block to the superblock application when the function block receives a message from the external service.*

201. Abuan describes multiple examples of “*an event notification*” that occur when a message is received from the videoconference server 715 (“*from the external service*”). Any one of these examples is sufficient to render this limitation obvious.

202. First, Abuan discloses, with reference to Figure 7 shown below, that a “video conference server 715 forwards (at operation 3) [a] video conference request [from device 705] to the video conference client 725 of the device 720” to establish a video conference between device 705 and device 720. Ex.1005, [0110]-[0111]. That is, the video conference client 725 on the device 720 receives a video conference request (“*a message*”) from the remote video conference server 715 (“*from the external service*”).

203. As discussed at [1.7.1], the communication between the video conference client 725 on device 720 and the video conference server 715 is performed by the video conference module 1302 on that device. Thus, a POSITA would have found it obvious that the video conference module 1302 (part of “*the function block*”) receives the video conference request (“*receives a message*”) from the video conference server 715 (“*from the external service*”) and then provides the video conference request to the video conference client 725.

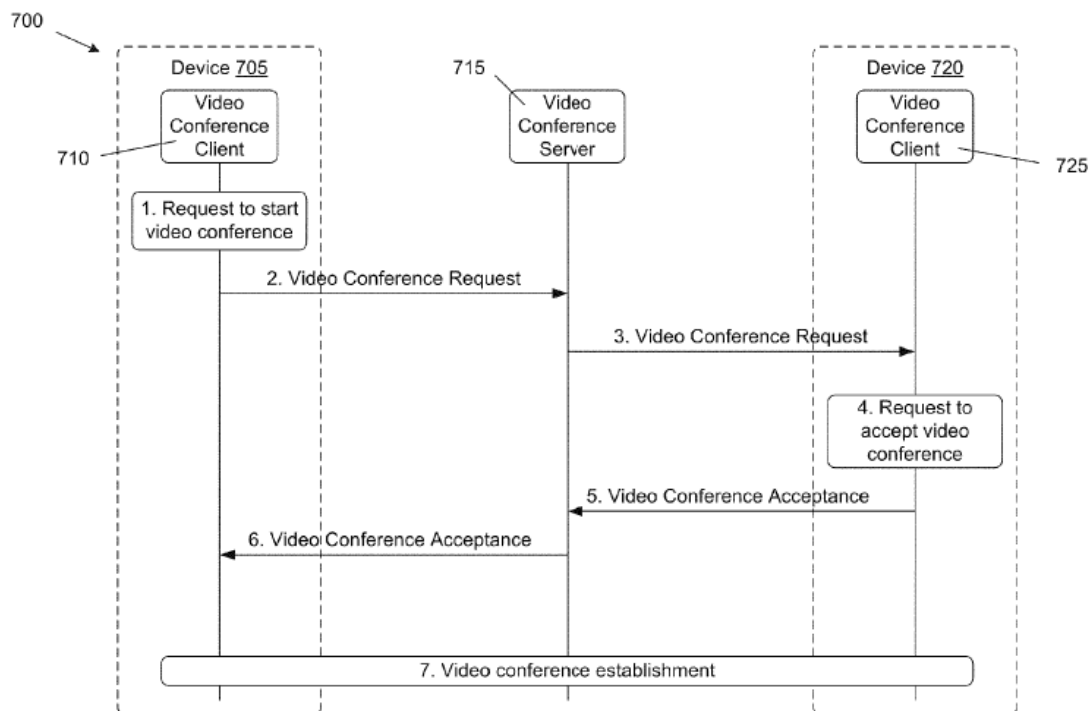


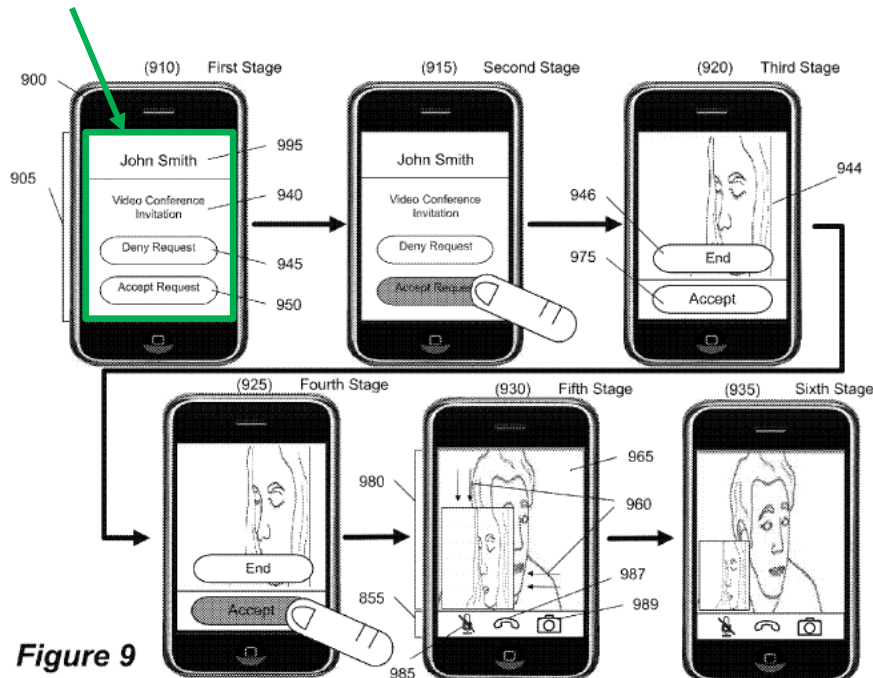
Figure 7

Ex.1005, Fig. 7

204. Abuan discloses that “[w]hen the video conference client 725 ... receives the video conference request, a user interface is displayed on the device

720 to indicate to the user of the device 720 that the user of the device 705 sent a request to start a video conference and to prompt the user of the device 720 to accept or reject the video conference request. An example of such a user interface is illustrated in FIG. 9.” Ex.1005, [0112]. A POSITA would have understood that this user interface corresponds to, for example, first stage (910) of Figure 9 and that the reject and accept prompts correspond to the “selectable Deny Request and Accept Request options 945 and 950,” respectively. Ex.1005, [0130].

UI containing
“event notification”



Ex.1005, Fig. 9 (annotated).

205. Further, it would have been obvious for a module in Abuan’s video conference module 1302, such as networking manager 1314, to send an “event

notification” to the client application upon receipt of a videoconference request from the videoconference server 715. Providing such a notification would facilitate the client application showing the invitation to the user, as Abuan contemplates. See Ex.1005, [0140] (describing the mobile device’s display as dynamic, responding “when a triggering event is received”).

206. Accordingly, Abuan teaches that when a video conference server 715 forwards a video conference request from device 705 to device 720, the video conference module 1302 (part of “*the function block*”) of device 720 receives the video conference request from the video conference server 715 (“*the external service*”) and then provides the video conference request (“*a message*”) to the video conference client 725 (part of “*the superblock application*,” as discussed at [1.7.1]) as a user interface (“*an event notification*”) of the video conference client 725 with options for accepting or rejecting the video conference request, rendering obvious “*providing an event notification by the function block to the superblock application when the function block receives a message from the external service.*”

207. Second, Abuan shows in Figure 9 that a participant may end the videoconference by selecting an “end conference button 987” (see “fifth stage 930” in Figure 9 shown above). Ex.1005, [0138]; *see also id.* [0125]. The videoconference may also end upon receipt of “instructions ... received from the other device participating in the video conference.” Ex.1005, [0194]. Abuan

describes some actions taken at the end of a conference, but Abuan also contemplates the videoconference manager 1304 performing still other actions. Ex.1005, [0194]. It would have been obvious for the videoconference manager 1304 to inform the client application of the end of the conference so that the client application would update the user interface (e.g., to cease displaying the picture-in-picture videoconference images discussed above at [1.8.1]-[1.8.2] and [9.8]-[9.10]).

208. Third, Abuan describes a mobile device “dynamically switch[ing] to transmitting the video captured by its other camera” “while transmitting only one camera’s captured video during a video conference.” Ex.1005, [0038]. “In such situations, the mobile device ... notifies any other device participating in the video conference of this switch so that this other device can provide a smooth transition on its end between the videos captured by the two cameras.” Ex.1005, [0038]. Thus, it would have been obvious for Abuan’s video conference module 1302 to notify the client application (“*providing an event notification by the function block to the superbloc application*”) of receiving such a camera-switching message from the other mobile device via the video conference server 715 (“*when the function block receives a message from the external service*”) so that the client application would facilitate the desired “smooth transition.” See Ex.1005, [0042]; Ex.1044, FIG. 48 and [0525] (illustrating a mobile device “perform[ing] a switch

camera animation to display a transition between the display of image from one camera of” the other device in the video conference and “the display of images from the other camera of” that other device).

14. Claim 12

a. [12.0] *The mobile device of claim 11 wherein the instructions further include*

209. Claim limitation [12.0] is substantially the same as [11.0] and would have been obvious for the reasons discussed above.

b. [12.1] *receiving, by the function block, a response to the event notification from the superblock application.*

210. As discussed at [11.1], Abuan describes multiple examples of an “*event notification.*” Abuan also describes multiple examples of a corresponding “*response.*”

211. First, as discussed at [11.1], Abuan teaches that a video conference module 1302 on a device 720 provides, to a video conference client 725 on the device, a user interface with options for accepting or rejecting a video conference request, as illustrated in Figure 9 of Abuan (first stage (910)).

212. Abuan further teaches that “the invite recipient may deny or accept the request by selecting the Deny Request option 945 or Accept Request option 950 in the UI, respectively.” Ex.1005, [0131]. When the user accepts the videoconference request, the client application (“*superblock application*”) sends an instruction (“*response*”) to video conference manager 1304 of the video conference

module 1302 (part of “*the function block*”) to start a videoconference. Ex.1005, [0187] & [0186] (“when a user of the dual camera mobile device accepts a video conference request, the instructions are received through the user interface and forwarded by the client application,” where the receiving of the instructions is performed by “performed by ... video conference manager 1304”). Thus, a POSITA would have found it obvious that the video conference manager 1304 of the video conference module 1302 (part of “*the function block*”) receives instructions from the client application (“*receiving ... a response ... from the superblock application*”) to start a videoconference when the user selects the Accept Request option 950 on the user interface (“*the event notification*”), rendering obvious “*receiving, by the function block, a response to the event notification from the superblock application.*”

213. Second, Abuan shows in Figure 9 that a participant may provide additional commands during the videoconference, such by selecting an “end conference button” 987 (see “fifth stage 930” in Figure 9 shown in [11.1]). Ex.1005, [0138]; *see also id.* [0125]. When a video conference ends, the client application “sends instructions to the video conference module 1302 such as instructions to... end a conference.” Ex.1005, [0172]. For example, the videoconference manager 1304 of the video conference module 1302 “receives instructions from the client application 1365 to end the video conference (i.e., due

to instructions ... received from the other device participating in the video conference).” Ex.1005, [0194]. That is, when the remote party chooses to end the videoconference, the videoconference manager 1304 receives an instruction from the client application to end the videoconference. Ex.1005, [0194]. The networking manager 1314 of the video conference module 1302 then performs actions such as “tear[ing] down the network connection with the other device” and “stop[ping] processing and encoding images.” Ex.1005, [0195]. It would have been obvious for the instruction to end the conference to be sent in response to the videoconference manager 1304 notifying the client application that the remote party has chosen to end the videoconference, as discussed at [11.1], rendering obvious limitation [12.1].

15. Claim 13

a. [13.0] *The mobile device of claim 10 wherein*

214. Claim limitation [13.0] is substantially the same as [10.0] and would have been obvious for the reasons discussed above.

b. [13.1] *the video window is associated with an audio/video (A/V) call session, and*

215. As discussed at [9.9], the PIP display teaches “*the video window*.” As discussed at [1.8.1] and [9.6.2], the PIP display displays videos from the audio/video conference call. Accordingly, the PIP display (“*the video window*”)

displays videos from the audio/video conference call (*“is associated with an audio/video (A/V) call session”*).

- c. **[13.2] *wherein the function block communicates with the external service to support the A/V call session.***

216. As discussed at [10.1], Abuan teaches that the video conference module 1302 (part of *“the function block”*) on mobile device 705 communicates with a remote video conference server 715 (*“communicates with external service”*) to establish an audio/video conference call (*“to support the A/V call session”*) with another device.

16. Claim 14

- a. **[14.0] *The mobile device of claim 9 wherein***

217. Claim limitation [14.0] is substantially the same as [9.0] and would have been obvious for the reasons discussed above.

- b. **[14.1] *the superblock application is not authorized to use at least one of the functions of the function block.***

218. As discussed at [9.7], the combination of Abuan and Lawson renders it obvious that the client application 1365 sends an authorization token to the video conference module 1302, where the authorization token authorizes the client application 1365 to access the video conferencing functions of the video conference module 1302. Lawson also discloses that, besides authorizing or permitting the bearer to “perform a particular action,” the authorization token can also be used to limit “which operations” the bearer can perform. Ex.1007, [0052].

The “restriction ... may ... be associated with ... number of communications, timing of communications, communication endpoints [(e.g., a client)], and communication mediums.” Ex.1007, [0052], [0032] (“(e.g., a client)”). Example communication mediums include “audio, video, chat, and/or other suitable communication mediums.” Ex.1007, [0048]. Thus, it would have been obvious to a POSITA for the authorization token to prohibit the client application 1365 (part of “*the superblock application*”) from accessing one or more of the video conferencing functions of the video conference module 1302 (part of “*the function block*”), for example, based on the number or timing of the client application 1365’s instructions to the video conference module 1302 to start an audio/video conference call, or based on the type of the conference call (e.g., audio or video), rendering obvious “*the superblock application is not authorized to use at least one of the functions of the function block.*”

219. A POSITA would have been motivated to combine the teachings of Lawson with Abuan and Guzman for the same reasons discussed at [1.5].

17. Claim 15

a. [15.0] *A method comprising:*

220. To the extent that the preamble is limiting, Abuan discloses it or renders it obvious.

221. As discussed at [1.4], the video conferencing and processing module 1300 “includes a client application 1365, a video conference module 1302, a media exchange module 1320, a buffer 1325, [and] a captured image processing unit (CIPU) driver 1330,” as well as “frame buffers 1405 and 1410, audio processing manager 1415, and audio driver 1420.” Ex.1005, [0170], [0183]. Also as discussed at [1.4], the video conferencing and processing module 1300 excluding the client application 1365 teaches “*a function block*.”

222. Abuan describes the functions and operations of the components of *the function block*. For example, Abuan describes the “video conferencing functions of the video conference module 1302,” “video processing and encoding functionalities” of the media exchange module 1320, and “audio processing operations” of an audio processing manager 1415. Ex.1005, [0171], [0237], [0183]-[0184]. Abuan also describes the audio driver 1420 routing audio data to the audio processing manager 1415 and to the speaker of the mobile device. Ex.1005, [0184]. The buffers (buffers 1325, 1405, and 1410) are described as storage for either image/video or audio data. Ex.1005, [0170], [0184]. “The CIPU driver 305 serves as a communication interface between a captured image processing unit (CIPU) 330 and the media exchange module.” Ex.1005, [0046]. The CIPU processes images. Ex.1005, [0176].

223. The functions and operations of *the function block*, examples of which are discussed above, each teach “*a method.*”

- b.** **[15.1]** *receiving, by a function block stored in a memory of a mobile device, a request for an audio/video (A/V) call from a superblock application stored in the memory of the mobile device;*

224. Claim limitation [15.1] is substantially the same as [1.4] and would have been obvious for the reasons discussed above.

- c.** **[15.2]** *determining, by the function block, that the superblock application is authorized to access A/V call functionality provided by the function block;*

225. Claim limitation [15.2] is the same as [1.5] and would have been obvious for the reasons discussed above.

- d.** **[15.3]** *determining, by the function block, that the A/V call requires access to a resource that is external to the mobile device;*

226. Claim limitation [15.3] is the same as [1.6] and would have been obvious for the reasons discussed above.

- e.** **[15.4.1]** *establishing, by the function block, the A/V call between the superblock application and the resource via a network interface, wherein the establishing is not supported by the superblock application;*

227. Claim limitation [15.4.1] is the same as [1.7.1] and would have been obvious for the reasons discussed above.

- f. **[15.4.2] wherein the establishing is not supported by the superblock application;**

228. Claim limitation [15.4.2] is the same as [1.7.2] and would have been obvious for the reasons discussed above.

- g. **[15.5.1] providing, by the function block, video output for the A/V call to a video window within a display window of the superblock application on the mobile device,**

229. Claim limitation [15.5.1] is the same as [1.8.1] and would have been obvious for the reasons discussed above.

- h. **[15.5.2] wherein the video window is displayed simultaneously with the display window of the superblock application on a screen of the mobile device; and**

230. Claim limitation [15.5.2] is the same as [1.8.2] and would have been obvious for the reasons discussed above.

- i. **[15.6] providing, by the function block, audio for the A/V call via the mobile device.**

231. Claim limitation [15.6] is the same as [1.9] and would have been obvious for the reasons discussed above.

18. Claim 16

- a. **[16.0] The method of claim 15 further comprising**

232. Claim limitation [16.0] is substantially the same as [15.0] and would have been obvious for the reasons discussed above.

- b. [16.1] *determining, by the function block, whether the function block has the capability to provide the A/V call.*¹

233. Claim limitation [16.1] is substantially the same as [2.1] and would have been obvious for the reasons discussed above.

19. Claim 17

- a. [17.0] *The method of claim 16 further comprising*

234. Claim limitation [17.0] is substantially the same as [16.0] and would have been obvious for the reasons discussed above.

- b. [17.1] *determining, by the function block, whether the function block has the capability to render the A/V call.*

235. Claim limitation [17.1] is substantially the same as [3.1] and would have been obvious for the reasons discussed above.

20. Claim 18

- a. [18.0] *The method of claim 17 further comprising*

236. Claim limitation [18.0] is substantially the same as [17.0] and would have been obvious for the reasons discussed above.

¹ I note that there is no earlier introduction of “a capability” in claim 16. For the purposes of my analysis, I treat the term “the capability” as referring to “a capability.”

- b. **[18.1] *determining, by the function block, whether the superblock has the capability to render the A/V call.***

237. Claim limitation [18.1] is substantially the same as [4.1] and would have been obvious for the reasons discussed above.

21. Claim 19

- a. **[19.0] *The method of claim 15 further comprising***

238. Claim limitation [19.0] is substantially the same as [15.0] and would have been obvious for the reasons discussed above.

- b. **[19.1] *initializing the function block, by the superblock application, when the superblock application is loaded into the memory and executed.***

239. Claim limitation [19.1] is substantially the same as [6.1] and would have been obvious for the reasons discussed above.

22. Claim 20

- a. **[20.0] *The method of claim 15 wherein***

240. Claim limitation [20.0] is substantially the same as [15.0] and would have been obvious for the reasons discussed above.

- b. **[20.1] *the request is an application programming interface (API) call from the superblock application to the function block.***

241. Claim limitation [20.1] is substantially the same as [7.1] and would have been obvious for the reasons discussed above.

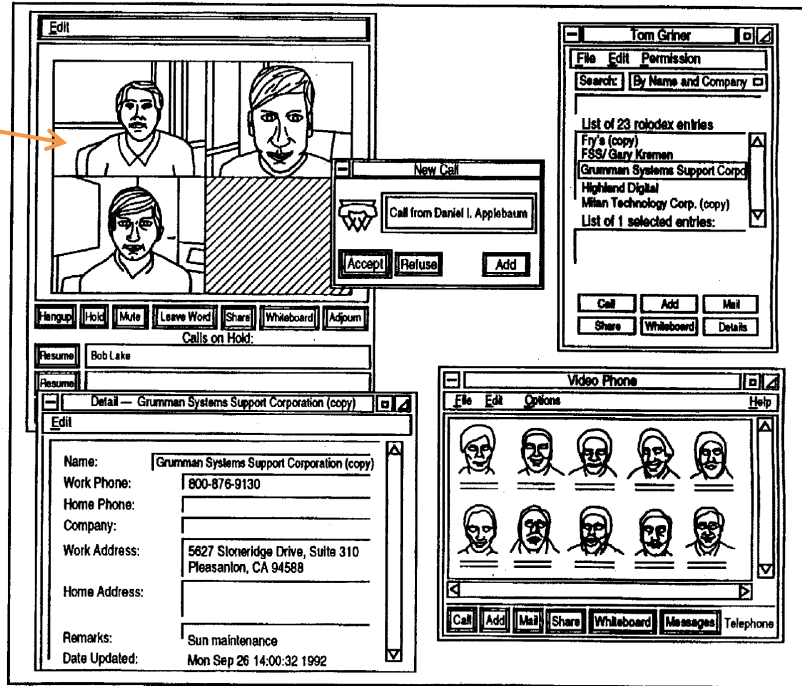
B. Ground 2: Claims 8 and 21 would have been Obvious over Abuan in view of Lawson, Guzman, and Ludwig.

1. Summary of Ludwig

242. U.S. Patent No. US 5,689,641 to Ludwig et al. (“Ludwig”) was filed on Oct. 1, 1993, and issued on Nov. 18, 1997. Ex.1009, 1 (fields 22 (filing date) and 45 (issue date)). Ludwig is titled “Multimedia Collaboration System Arrangement for Routing Compressed AV Signal through a Participant Site Without Decompressing the AV Signal.” Ex.1009, 1 (field 54). Ludwig was not before the Examiner during prosecution of the ’032 patent.

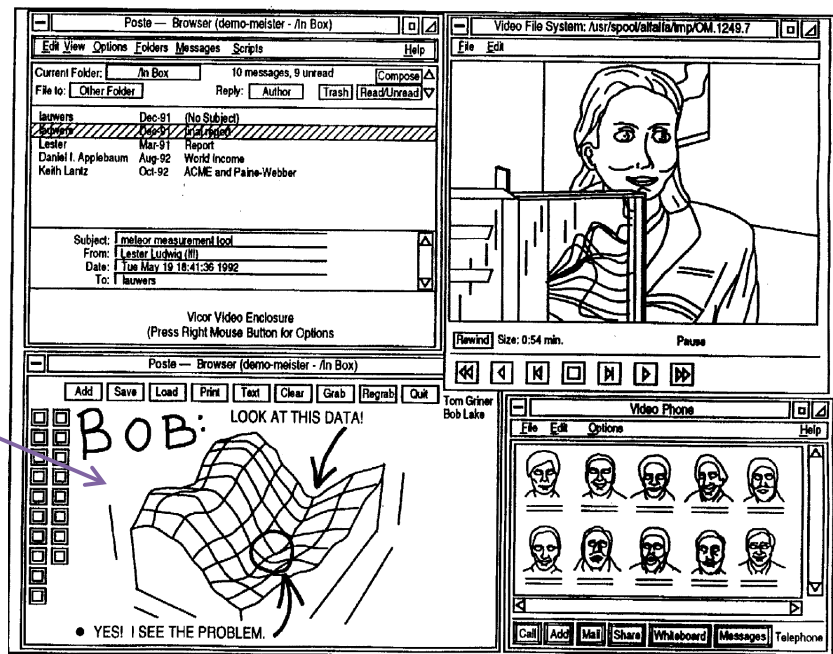
243. Ludwig describes a multimedia collaboration system that integrates real-time audio/video teleconferencing with data conferencing to replicate face-to-face meetings across distributed networks and heterogeneous platforms. Ex.1008, Abstract. Beyond high-quality, multi-party video and audio, the system 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.

full-motion video of
three conference
participants



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

Data shared and
annotated by
conference
participants



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

244. Ludwig is analogous art to the '032 patent. First, it is directed to the same field of endeavor as the '032 patent, namely, the “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:15-16; *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”).² Ludwig and the '032 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 '032 patent both describe at length features for facilitating videoconferencing. Ex.1001, Abstract, 3:53-65 (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...”).

² While the '032 patent distinguishes the claimed “mobile device” from a “laptop,” 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, 2:39-40, 2:45-46; Ex.1045, 1:37 (“mobile devices, such as notebook and laptop computers”); Ex.1046, 4:19-24.

245. Second, Ludwig is reasonably pertinent to a problem the inventors of the '032 patent allegedly attempted to solve. The '032 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, 3:33-36. The '032 patent's proposed solution is to provide functionality—such as communication functions—to a superblock application. Ex.1001, 3:53-4:4. 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.

2. Claim 8

a. [8.0] *The device of claim 1 wherein the instructions further include*

246. See limitations [1.0] and [1.3].

- b. **[8.1] *enabling, by the function block, a user of the device to collaborate on at least one of a document, a file, or an application with a user on another device.***

247. As discussed at [1.7.1], the video conference module 1302 (part of the “*function block*”) establishes an audio/video conference call between the video conference client 710 running on a device 705 and the video conference client 725 running on a device 720. A POSITA would have understood that the audio/video conference call would enable “a user of the device 705” that sends “a request ... to start a video conference with the device 720” and “the user of the device 720 [that] accepts the video conference request” to engage in a video conference once the audio/video conference call is established. Ex.1005, [0112], [0115]. 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].

248. Ludwig teaches “computer-based systems for enhancing collaboration between and among individuals who are separated by distance and/or time.” Ex.1008, 1:8-10. The systems include “a collaborative multimedia workstation (CMW) system” where “personal computers (e.g., ... laptop) ... provide such features as real-time videoconferencing, data conferencing.” Ex.1008, 14:65-15:3.

The videoconferencing can be a “basic two-party videoconferencing” or “multi-party conference calls (involving more than two participants).” Ex.1008, 21:44, 24:12. “Data conferencing, in turn, includes snapshot sharing (sharing of ‘snapshots’ of selected regions of the user's screen), application sharing (shared control of running applications), shared whiteboard (equivalent to sharing a ‘blank’ window), and associated telepointing and annotation capabilities.” Ex.1008, 3:1-6. The shared whiteboard is “for subsequent annotation.” Ex.1008, 36:30-32. Data conferencing can occur simultaneously with videoconferencing, as Ludwig teaches that “a ‘snapshot’ of a selected portion of a participant's CMW screen (such as a window) to be displayed on the CMW screens of other selected participants (whether or not those participants are also involved in a videoconference).” Ex.1008, 26:29-33.

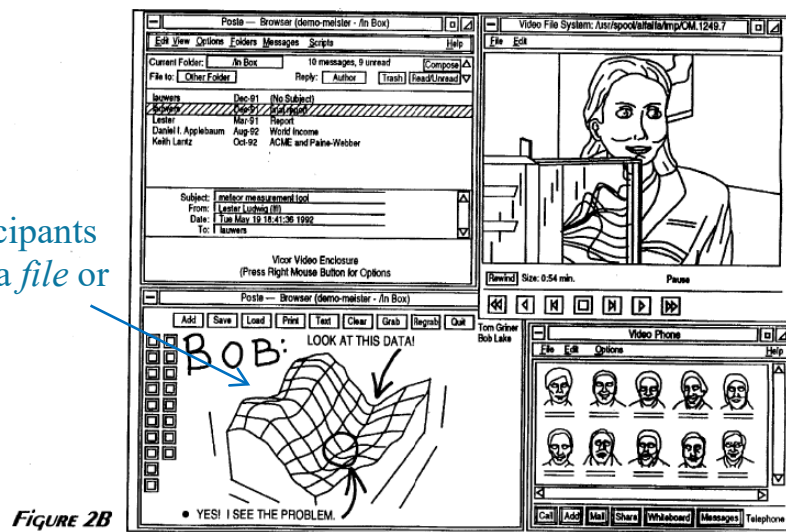
249. Specifically, Ludwig explains that:

The data conferencing component of the above-described system supports the sharing of visual information at one or more CMWs (as described in greater detail below). This encompasses both “snapshot sharing” (sharing “snapshots” of complete or partial screens, or of one or more selected windows) and “application sharing” (sharing both the control and display of running applications). ...

In all cases, any participant can point at or annotate the shared data. These associated telepointers and annotations appear on every participant's CMW screen as they are drawn (i.e., effectively in real time). For example, note FIG. 2B which illustrates a typical CMW screen during a multi-party teleconferencing session, wherein the screen contains annotated shared data as well as video images of the conferees.

Ex.1008, 9:12-28. Ludwig illustrates in Figure 2B, for example, that multiple users have collaborated on a graph document, with one user writing “BOB: LOOK A THIS DATA!” and another user responding “YES! I SEE THE PROBLEM”:

Conference participants
collaborating on a file or
document



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

250. Thus, different participants, at “more [than one] CMWs,” of a “data conferencing” session occurring as part of a videoconferencing session perform

“application sharing,” i.e., “shar[e] both the control and display of running applications.” Ex.1008, 9:12-18.

251. Ludwig also teaches that the shared snapshot that is annotated can be a file, explaining that “instead of sharing a snapshot of data on his current screen, a user may instead choose to share a snapshot that had previously been saved as a file.” Ex.1008, 27:34-36.

252. Ludwig further teaches that the shared data that is annotated can be a document, explaining that “one participant desires to share a long document,” such as “a spreadsheet,” in which case the participant “use[s] the SHARE button to share that page, discuss and perhaps annotate it.” Ex.1008, 27:25-30.

253. Furthermore, 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*”).

254. Accordingly, the combination of Abuan and Ludwig renders obvious that the video conference module 1302 (part of the “*function block*”) establishes an audio/video conference call between the video conference client 710 running on a device 705 and the video conference client 725 running on a device 720, where the

user of device 705 and the user of device 720 share the control and display of running applications, and share and annotate files and documents, rendering obvious “*enabling, by the function block, a user of the device to collaborate on at least one of a document, a file, or an application with a user on another device.*”

255. A POSITA would have been motivated to combine the teachings of Ludwig with Abuan for multiple reasons.

256. 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 video conferencing, 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.

257. A POSITA would have found it obvious to incorporate Ludwig's multimedia collaboration features to the video conferencing system of Abuan. The combination with Ludwig is simply the combination 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). Further, 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).

258. In addition, a POSITA would also have been motivated to combine Abuan with Ludwig because Abuan teaches that "[i]n addition to transmitting video content during a video conference with another device, the mobile device ... can transmit other types of content along with the conference's video content." Ex.1005, [0037]. So motivated, a POSITA would have looked to Ludwig for implementation details because Ludwig describes "[a] multimedia collaboration system that integrates separate real-time and asynchronous networks--the former

for real-time audio and video, and the latter for control signals and textual, graphical and other data--in a manner that ... closely approximates the experience of face-to-face collaboration, while liberating the participants from the limitations of time and distance.” Ex.1008, Abstract. As discussed above, Ludwig’s system includes features such as the “sharing of ‘snapshots’ of selected regions of the user’s screen,” “application sharing,” and a “shared “whiteboard.” Ex.1008, 3:1-6. A POSITA would have had a reasonable expectation of success in modifying Abuan with Ludwig because Ludwig describes the above-noted detailed scenario involving application sharing and whiteboarding. Ex.1008, 35:35-41:37.

3. Claim 21

a. [21.0] *The method of claim 15 further comprising*

259. Claim limitation [21.0] is substantially the same as [15.0] and would have been obvious for the reasons discussed above.

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

260. Claim limitation [21.1] is substantially the same as [8.1] and would have been obvious for the reasons discussed above.

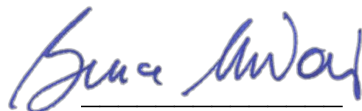
IX. CONCLUSION

261. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements

Declaration of Bruce McNair
Inter Partes Review of U.S. 9,027,032

and the like so made are punishable by fine or imprisonment, or both, under
section 1001 of Title 18 of the United States Code.

Dated: January 10, 2026



Bruce McNair