(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 November 2009 (05.11.2009)

PCT

(10) International Publication Number
WO 2009/134548 A2

(51) International Patent Classification:
G06F 17/24 (2006.01) *G06F 17/30* (2006.01) *G06F 17/21* (2006.01)

(21) International Application Number: PCT/US2009/037920

(22) International Filing Date: 23 March 2009 (23.03.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/111,174 28 April 2008 (28.04.2008) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; Attn: Sharon Rydberg, (sharonr), 8/2321, LCA, International Patents Department, One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: **BAILOR, Jonathan, Beckett**; C/o Microsoft Corporation, Lca, International Patents Department, One Microsoft Way, Redmond, WA 98052-6399 (US). **BERNSTEIN, Ethan, Joseph**; c/0 Microsoft Corporation, LCA, International Patents Department, One Microsoft Way, Redmond, Washington 98052-6399 (US). **KROUT, Kelly, Michael**; c/o Microsoft Corporation, LCA, International Patents Department, One Microsoft Way, Redmond, Washington 98052-6399 (US). **MIZULO, Matthew, Eric**; c/o Microsoft Corporation, LCA, International Patents Department, One Microsoft Way, Redmond, Washington98052-6399 (US). **GORDNER, Jonathan, Ian**.

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

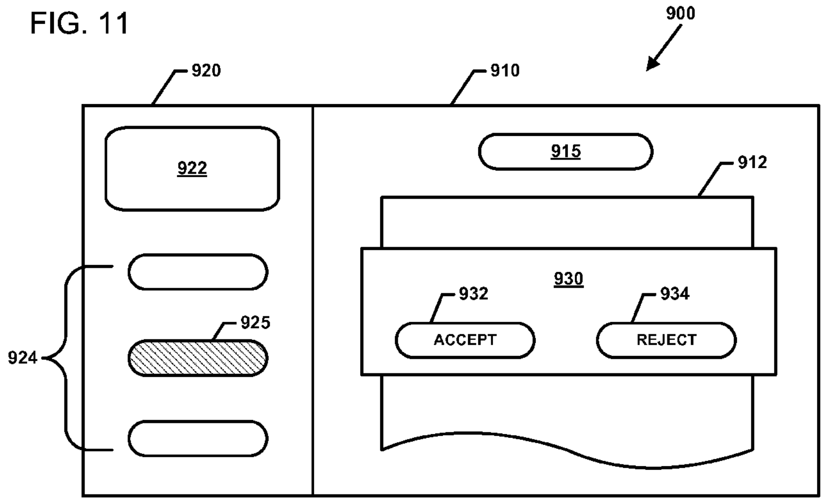
(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

[Continued on next page]

(54) Title: CONFLICT RESOLUTION

FIG. 11



(57) **Abstract**: Embodiments of a collaborative authoring environment enable a user to resolve editing conflicts arising when synchronizing a user copy of a data file with a master copy of the data file. Content updates may be synchronized separately from metadata updates. Metadata updates may be synchronized automatically, whereas content updates may be synchronized only when any identified editing conflicts are resolved. When an editing conflict is identified, the user interface of the authoring application may be configured to toggle between displaying and hiding the identified editing conflicts.

WO 2009/134548 A2

CONFLICT RESOLUTION

BACKGROUND

[0001] Traditional collaborative editing tends to be performed serially. Users take turns accessing a document, editing the document, and storing their edits. To inhibit editing conflicts, the accessing user may place a lock on the file to inhibit other users from editing the document when the accessing user is editing the document. The iterative editing process can cause delays since each user may wait for a turn at editing the document. In addition, the iterative editing process may be difficult to manage. For example, each user may need to keep track of who is editing which portions of the document, which version of the document is the most recent, and when the user will have a turn.

[0002] In other types of traditional collaborative editing, each user can edit a different copy of a document. Subsequently, all of the edited copies may be merged into a single document. This large scale merge also may cause delays, lead to numerous editing conflicts, and/or be difficult to manage. For example, the user responsible for merging the documents may be required to track the relationship between the documents. The user also may be responsible for resolving conflicts among two or more of the edited copies.

[0003] It is with respect to these and other considerations that the present disclosure has been made.

SUMMARY

[0004] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended as an aid in determining the scope of the claimed subject matter.

[0005] Embodiments of the present disclosure are generally directed to enabling a user to resolve editing conflicts arising when synchronizing a data file in a collaborative environment. Each user authoring a user copy of a data file may resolve editing conflicts between a master copy of the data file and the user copy. Updates from the user copy of the data file may be incorporated into the master copy after editing conflicts have been resolved.

[0006] According to aspects of the disclosure, an authoring application enables a user to selectively show and hide editing conflicts within a user copy of a data file. The authoring application enables free editing of the user copy regardless of whether or not editing conflicts are shown or hidden. According to other aspects, authoring application provides a contextual user interface that enables a user to resolve the displayed editing conflicts.

[0007] In some embodiments, showing editing conflicts includes annotating conflicting content. In one embodiment, annotating conflicting content indicates how the content conflicts. In another embodiment, only conflicting content inserted, revised, and/or deleted within the user copy is annotated.

[0008] These and other features and advantages will be apparent from a reading of the following detailed description and a review of the associated drawings. It is to be understood that both the foregoing general description and the following detailed description are explanatory only and are not restrictive of aspects as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 illustrates an example authoring system having features that are examples of inventive aspects of the disclosure;

[0010] FIG. 2 is a schematic block diagram illustrating an example authoring system including a storage device, which stores a master copy of a data file, communicatively coupled to a user device, which stores a user copy of the data file, in accordance with the principles of the disclosure;

[0011] FIG. 3 is a flowchart illustrating an operational flow for an example authoring process by which a user copy of a data file may be authored in accordance with the principles of the disclosure;

[0012] FIG. 4 is a schematic block diagram of a user computing system configured to implement an authoring environment in accordance with the principles of the disclosure;

[0013] FIG. 5 is a flowchart illustrating an operational flow for an example synchronize process by which the synchronize operation of FIG. 3 may be implemented in accordance with the principles of the disclosure;

[0014] FIG. 6 is a flowchart illustrating an operational flow for an example editing process by which the continuation operation of FIG. 5 may be implemented in accordance with the principles of the disclosure;

[0015] FIG. 7 is a flowchart illustrating an operational flow for an example review process by which one or more editing conflicts may be reviewed and optionally resolved in accordance with the principles of the disclosure;

[0016] FIG. 8 illustrates an example user interface that is displayed to a user when no unresolved editing conflicts have been identified in accordance with the principles of the disclosure;

[0017] FIG. 9 illustrates an example user interface displayed to a user when at least one editing conflict has been identified in accordance with the principles of the disclosure;

[0018] FIG. 10 illustrates an example user interface that is displayed to a user when at least one editing conflict has been identified and the user has chosen to review the identified editing conflict in accordance with the principles of the disclosure;

[0019] FIG. 11 is a schematic diagram of a user interface including a resolution interface that is displayed when an editing conflict is selected from the listing in the summary window in accordance with the principles of the present disclosure;

[0020] FIG. 12 is a flowchart illustrating an operational flow for an example implementation process by which the authoring application implements resolution instructions provided by the user in accordance with the principles of the disclosure;

[0021] FIG. 13 is a flowchart illustrating an operational flow for an example accept process by which an accepted editing operation may be instantiated into the merged version of the data file in accordance with the principles of the disclosure;

[0022] FIG. 14 is a flowchart illustrating an operational flow for an example reject process by which a rejected editing operation may be removed from or undone within the merged version of a data file in accordance with the principles of the disclosure;

[0023] FIGS. 15-22 illustrate changes to an example user interface displayed by an authoring application as a first user is editing a user copy of a data file online in accordance with the principles of the disclosure; and

[0024] FIGS. 23 - 29 illustrate changes to an example user interface displayed by an authoring application as a first user is editing a user copy of a data file offline in accordance with the principles of the disclosure.

DETAILED DESCRIPTION

[0025] In the following detailed description, references are made to the accompanying drawings that form a part hereof, and in which are shown by way of illustrations specific embodiments or examples. While the disclosure will be described in the general context of program modules that execute in conjunction with an application program that runs on an operating system on a computer system, those skilled in the art will recognize that the disclosure also may be implemented in combination with other program modules. The embodiments described herein may be combined and other embodiments may be utilized without departing from the spirit or scope of the present disclosure. The following detailed description is therefore not to be taken in a limiting sense, and the scope of the invention is defined by the appended claims and their equivalents.

[0026] In accordance with the principles of the present disclosure, a collaborative authoring application provides an authoring environment in which one or more users can edit one or more data files (e.g., word processing documents, presentation documents, spreadsheet documents, pictures or other images, sound files, software applications, executable code, etc.) via editing operations (e.g., insertion, revision, and/or deletion of content and/or metadata). Each user obtains a user copy of the data file based on a version of the data file represented by a master copy. A user may edit the user copy of the data file to create a new version of the data file and periodically synchronize the new version with the master copy.

[0027] Synchronization, as the term is used herein, refers to the sending and/or receiving of one or more version updates between the master copy of the data file and a user copy of the data file to create a common version of the data file. For example, each user periodically may send to the master copy a version update representing the new version of the data file and periodically may receive from the master copy a version update representing a current version of the master copy (e.g., which may reflect edits performed by other users).

[0028] As briefly described above, embodiments of the present disclosure are directed to enabling a user to resolve editing conflicts arising when synchronizing a data file in a collaborative environment. In general, editing conflicts may arise when the master copy of the data file changes (e.g., editing operations are performed on the master copy) between when a user copy is obtained and the user copy is synchronized with the master copy or between synchronizations. Such changes to the master copy will be referred to herein as "intervening changes."

[0029] An editing operation performed on a user copy of the data file results in an editing conflict, as the term is used herein, when the editing operation interferes with an intervening change made to the master copy (i.e., or vice versa). For example, if a user performs an editing operation in a user copy of a data file to revise a first data unit that was deleted in a master copy of the data file by an intervening change, then the editing operation resulting in the revision of the first data unit would conflict with the editing operation resulting in the deletion of the first data unit in the master copy.

[0030] Referring now to the drawings, FIG. 1 illustrates an example authoring system 100 having features that are examples of inventive aspects of the disclosure. The authoring system 100 includes a storage device 120 storing a master copy 150 of a data file (e.g., word processing documents, presentation documents, spreadsheet documents, pictures or other images, sound files, software applications, executable code, etc.). In one embodiment, the storage device 120 can include one or more storage devices (e.g., a network of storage devices). In another embodiment, the storage device 120 can include one or more computing devices.

[0031] The authoring system 100 also includes at least one user computing device 110 that may be communicatively coupled to the storage device 120. As the term is used herein, a user computing device 110 includes any device configured to obtain and author a user copy 155 of a data file from a master copy 150 of the data file. As the term is used herein, authoring a data file may include creating the data file and/or editing the data file via editing operations. Each of the user computing devices 110 can author the data file by creating a user copy 155 of the data file based on the master copy 150. The user device 110 may edit the user copy 155 when the user device 110 is communicatively coupled to the storage device 120 (i.e., online) or when the user device 110 is disconnected from the storage device 120 (i.e., offline).

[0032] The user copy 155 of the data file may be synchronized when the user computing device 110 communicatively couples to the storage device 120 (i.e., is online) and periodically sends to the storage device 120 one or more updates to be incorporated into the master copy 150 and, thereby, shared with other user computing devices. Synchronization of the user copy 155 also includes periodically obtaining from the storage device 120 updates from the master copy 150 that originated from other user computing devices. When a user computing device 110 is offline, the user computing device does not synchronize with the storage device 120 and, hence, the other user computing devices.

[0033] Additional details pertaining to synchronization of a user copy of a data file with a master copy can be found in co-pending application Serial No. 11/938,082, filed November 9, 2007, and entitled "Collaborative Authoring," the disclosure of which is hereby incorporated herein in its entirety. Additional details pertaining to synchronization when the user computing device is offline can be found in co-pending application Serial No. 11/957,010, filed December 14, 2007, and entitled "Collaborative Authoring Modes," the disclosure of which is hereby incorporated herein in its entirety.

[0034] In the example shown in FIG. 1, four user computing devices 110A, 110B, 110C, and 110D are communicatively coupled to a storage device 120. In other embodiments, however, any number of user computing devices 110 may be coupled to the storage device 120. In the example shown, each user computing device 110A, 110B, 110C, 110D can send to the storage device 120 updates generated by the user of the user computing device and can request from the storage device 120 updates generated by the users of the other user computing devices.

[0035] The user computing devices 110A, 110B, 110C, 110D can be a different device from the storage device 120 or can include different user accounts implemented on the storage device 120. In one embodiment, a device that acts as a storage device 120 for one data file may act as a user computing device 110 for a different data file and vice versa. In one embodiment, the storage device 120 can be a server computing device and the user computing devices 110A, 110B, 110C, 110D can be client computing devices.

[0036] According to aspects of the disclosure, updates to the data file include content updates and/or metadata updates. As the term is used herein, content updates refer to any editing operation made to the substantive content of a data file. For example, content updates for a word processing document can include added paragraphs (i.e., or sections thereof), deleted paragraphs (i.e., or section thereof), revised paragraphs (i.e., or sections thereof), and additions, deletions, and/or changes to tables, charts, images, or other such objects. In another embodiment, content updates for a presentation document can include added, deleted, and/or revised pictures, text, animations, sounds, and other such data objects.

[0037] As the term is used herein, metadata updates refer to any editing operation made to metadata of the data file. Non-limiting examples of metadata include content locks, presence information, and other such data. Presence information indicates which users have indicated an intention to edit the document. Content locks inhibit editing of any content within the lock by users that do not own the lock. For example, content locks may inhibit editing conflicts by indicating which portions of a document or other data file have been claimed by another user. In some embodiments, the content locks can prevent (i.e., bar) a user from editing a portion of a document that has been claimed by another user. In other embodiments, however, the user can choose to break the content lock and edit the portion of the data file. In such cases, the authoring application can warn the user that conflicts may arise when editing the locked portion.

[0038] As shown in FIG. 2, content 152 and metadata 154 of a data file can be stored in memory 125 of the storage device 120. In some embodiments, the metadata 154 of the data file can be stored separately from the content 152. For example, the content 152 can be stored in the data file 150 and the metadata 154 can be stored in a table (not shown) separate from the data file 150. In other embodiments, however, the metadata 154 can be stored within the data file 150. Content 152' and metadata 154' of the user copy 155 of the data file can be stored in a cache (see cache 426 in FIG. 4) on a user computing device 110. One or more authoring applications 130 on the user computing device 110 process and manipulate the content 152' and/or the metadata 154' of the user copy 155 of the data file.

[0039] In general, the user computing devices 110 can synchronize content updates separately from metadata updates. In some embodiments, metadata updates are automatically synchronized among the storage device 120 and user computing devices 110, whereas content updates from each user computing device 110 are synchronized at the request of the respective user. In one embodiment, the authoring environment 100 may synchronize content updates only when editing conflicts do not exist (i.e., or have been resolved), but may synchronize metadata updates regardless of existing editing conflicts.

[0040] In one embodiment, an editing conflict may stem from a content update received from the master copy. In such an embodiment, changes to content 152' and/or metadata 154' of the user copy 155 interfere with intervening changes to the content 152 of the master copy 150. Such editing conflicts are referred to herein as mergeable conflicts. For example, in one embodiment, the same data unit may have been edited differently in the user copy and the master copy of the data file in between synchronizations. In another embodiment, the data unit may have been edited in the master copy 150 before a content lock obtained on the data unit in the user copy 155 was synchronized with the master copy 150.

[0041] In another embodiment, an editing conflict may stem from a metadata update received from the master copy 150. In such an embodiment, changes to the content 152' and/or metadata 154' of the user copy 155 interfere with intervening changes to the metadata 154 of the master copy 150 (e.g., the addition of content locks). Such editing conflicts are referred to herein as unmergeable conflicts. For example, the user device 110 may receive a metadata update from the master copy 150 of a data file indicating that content revised in the user copy 155 has already been locked by another user.

[0042] In one embodiment, changes to the metadata 154' of a user copy 155 of a data file that interfere with intervening changes to the master copy 150 are overridden by the intervening changes to the master copy 150. For example, if the storage device 120 receives a metadata update from a first user device 110A (FIG. 1) requesting a content lock on a first data unit of a data file and determines the corresponding first data unit of the master copy 150 is locked already to a second user device 110B (FIG. 1), then the storage device 120 will deny the lock request of the first user device 110A.

[0043] FIG. 3 is a flowchart illustrating an operational flow for an example authoring process 300 by which an authoring application may author a user copy of a data file, such as user copy 155 of FIG. 2. The authoring process 300 initializes and begins at a start module 302 and proceeds to an author operation 304. In general, the author operation 304 edits the user copy of the data file. In one embodiment, the author operation 304 obtains a user copy of the data file based on a master copy of an existing data file (e.g., from a storage device). In another embodiment, the author operation 304 creates and edits a new data file, generates a master copy of the data file (e.g., periodically or when editing is completed), and stores the master copy of the data file (e.g., on a storage device).

[0044] A receive operation 306 obtains at the authoring application updates indicating intervening changes made to the master copy of the data file. For example, in one embodiment, the receive operation 306 obtains a content update indicating any intervening changes made to the content of the master copy by one or more other users authoring the data file. In another embodiment, the receive operation 306 obtains a metadata update indicating any intervening changes made to the metadata of the master copy by one or more other users authoring the data file. In another embodiment, the receive operation 306 obtains both content and metadata updates.

[0045] In one embodiment, the receive operation 306 receives updates from the master copy at predetermined intervals. In another embodiment, the receive operation 306 receives an update from the master copy when a threshold amount of editing has been performed on the master copy. In another embodiment, the receive operation 306 receives an update from the master copy in response to a request for the update. For example, the receive operation 306 may request an update from the master copy in order to update the data file before saving the data file. In such an embodiment, the receive operation 306 requests an update from the master copy when instructions to synchronize the data file are received from the user.

[0046] An update operation 308 instantiates the intervening changes into the user copy of the document (e.g., by merging the intervening changes into the user copy of the document). In one embodiment, the update operation 308 instantiates intervening metadata changes differently from intervening content changes. In one embodiment, the update operation 308 may instantiate content updates and metadata updates automatically. In another embodiment, the update operation 308 may instantiate metadata updates automatically and may instantiate content updates at the request of the user. For example, the update operation 308 may present a button or other interface tool to the user indicating the availability of updates that may be instantiated by selecting the button or other interface tool.

[0047] The update operation 308 also may determine whether any editing conflicts exist. In some embodiments, the update operation 308 may instantiate the intervening changes differently depending on whether editing conflicts are identified. For example, in one embodiment, the update operation 308 may inhibit instantiation of content updates when editing conflicts are identified, but may continue to instantiate metadata updates automatically despite the existence of editing conflicts. Advantageously, synchronizing the metadata updates despite the existence of editing conflicts may mitigate the creation of further editing conflicts. For example, synchronizing lock data may inhibit concurrent editing of the same data unit by different users.

[0048] A synchronize operation 310 attempts to synchronize the user copy with the master copy by forwarding to the master copy updates indicating changes made to the user copy of the data file. In one embodiment, the synchronize operation 310 forwards the updates for distribution to other users collaboratively authoring the data file. The synchronize operation 310 only stores the user copy as the master copy (i.e., overwrites the master copy) if no editing conflicts between the user copy and the master copy are identified. In one embodiment, the synchronize operation 310 obtains the most recent version of the master copy and determines whether editing conflicts exist between the user copy and the most recent version of the master copy.

[0049] According to aspects of the disclosure, the synchronize operation 310 may enable the user to initiate resolution of the editing conflicts at any time after the editing conflict has been identified at the discretion of the user. The synchronize operation 310 may enable the user to continue freely editing the user copy of the data file even though one or more editing conflicts have been determined to exist. In one embodiment, the synchronize operation 310 may continue to edit the user copy with editing conflicts being hidden from the user. In another embodiment, the synchronize operation 310 may continue to edit the user copy with editing conflicts being presented to the user. If editing of the user copy is continued despite the existence of one or more editing conflicts, then content updates indicating changes to the user copy may be stored locally until the editing conflicts are resolved instead of being forwarded to the master copy for synchronization as will be discussed in greater detail herein.

[0050] The synchronize operation 310 may synchronize metadata updates differently from content updates. For example, in one embodiment, the synchronize operation 310 may forward content updates only if all editing conflicts have been resolved and may forward metadata updates regardless of whether editing conflicts have been resolved. As noted above, synchronizing the metadata updates despite the existence of editing conflicts may mitigate the creation of further editing conflicts. Furthermore, ceasing to synchronize content updates when editing conflicts exist may inhibit introducing the editing conflicts into the master copy of the data file. The authoring process 300 completes and ends at a stop module 312.

[0051] In general, an authoring environment having features that are examples of inventive aspects in accordance with the principles of the disclosure can be implemented on a user computing device (e.g., a personal computer, a server computer, a notebook computer, a PDA, a Smartphone, or any other such computing device). A non-limiting embodiment of a user computing system 400 configured to implement an authoring environment and perform authoring processes, such as authoring process 300 of FIG. 3, is described herein with reference to FIG. 4.

[0052] In FIG. 4, the exemplary computing system 400 for implementing the principles of the disclosure includes a user computing device, such as user computing device 410. In a basic configuration, the user computing device 410 typically includes at least one processing unit 415 for executing applications and programs stored in system memory 420. Depending on the exact configuration and type of computing device 910, the system memory 420 may include, but is not limited to, RAM, ROM, EEPROM, flash memory, CD-ROM, digital versatile disks (DVD) or other optical storage devices, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or other memory technology.

[0053] System memory 420 typically stores an operating system 422, such as the WINDOWS® operating systems from MICROSOFT CORPORATION of Redmond, WA, suitable for controlling the operation of the computing device 410. System memory 420 also may include a data file cache 426 in which a user copy 427 of a document can be stored. Metadata 429 of the data file also can be stored within the user cache 426.

[0054] The system memory 420 also may store one or more software applications, such as authoring applications 424 for creating and editing data files. One non-limiting example of an authoring application 424 suitable for authoring documents in accordance with the principles of the present disclosure is MICROSOFT® OFFICE WORD authoring software from MICROSOFT CORPORATION of Redmond, WA. Other non-limiting examples of authoring applications include POWERPOINT® presentation software and VISIO® drawing and diagramming software, both also from MICROSOFT CORPORATION of Redmond, WA.

[0055] Computing device 410 also may have input device(s) 430, such as a keyboard, mouse, pen, voice input device, touch input device, etc., for entering and manipulating data. Output device(s) 435, such as a display screen, speakers, printer, etc., also may be included. These output devices 435 are well known in the art and need not be discussed at length herein.

[0056] The computing device 410 also may contain communication connections 440 that allow the device 410 to communicate with other computing devices, for example, the storage device 120 of FIG. 1, over a network in a distributed computing environment (e.g., an intranet or the Internet). By way of example, and not limitation, communication device media 440 includes wired media such as a wired network or direct-wired connection, and wireless media, such as acoustic, RF, infrared and other wireless media.

[0057] FIG. 5 is a flowchart illustrating an operational flow for an example synchronize process 500 by which the synchronize operation 310 of FIG. 3 may be implemented. The synchronize process 500 initializes and begins at a start module 502 and proceeds to an obtain operation 504. The obtain operation 504 acquires a version update from the master copy. For example, in one embodiment, the obtain operation 504 may request the version update from the master copy. In another embodiment, the receive operation 504 may receive periodic version updates from the master copy (e.g., at predetermined intervals).

[0058] An identify operation 506 determines whether editing conflicts exist between the current version of the master copy and the user copy. More specifically, the identify operation 506 determines whether any intervening changes to the master copy and any editing operations performed on the user copy since the most recent synchronization (i.e., or since the user copy was obtained) interfere with one other.

[0059] A merge operation 508 combines the user copy and the master copy into a merged version of the data file and presents the merged version to the user. In one embodiment, the merge operation 508 integrates the intervening changes into the user copy of the data file. In another embodiment, the merge operation 508 integrates the user changes into the master copy of the data file. In another embodiment, the merge operation 508 integrates the intervening changes and the user changes into the most recently synchronized version of the master copy (i.e., the version of the master copy that was obtained and edited by the user).

[0060] A first determination module 510 splits the flow of the synchronize process 500 based on whether any editing conflicts were identified by the identify operation 506 and/or whether any identified editing conflicts from previously received updates remain unresolved. If the first determination module 510 determines at least one editing conflict has been identified, then an alert operation 512 indicates the presence of the editing conflict to the user. For example, the alert operation 512 may present a message to the user indicating the existence of one or more editing conflicts. In one embodiment, the alert operation 512 also may indicate consequences of having unresolved, identified editing conflicts (e.g., the inability to fully synchronize the user copy with the master copy until the editing conflict is resolved).

[0061] A second determination module 514 enables the user to choose when to review and resolve the identified editing conflict. In the example shown, the second determination module 514 presents to the user an option of reviewing the editing conflict or continuing to edit the user copy of the data file without viewing the editing conflict. If the second determination module 514 determines the continued editing option has been selected by the user, then a continue operation 516 enables the user to edit the data file freely as will be discussed in greater detail herein.

[0062] If the second determination module 514 determines the review option has been selected by the user, however, then a review operation 518 enables the user to review and optionally to resolve the editing conflict using a resolution process that will be discussed in greater detail herein. When the review operation 518 completes, the synchronize process 500 proceeds back to the first determination module 510 to determine whether any editing conflicts remain unresolved. The operation flow described above repeats until the first determination module 510 determines no editing conflicts exist.

[0063] When the first determination module 510 determines no editing conflicts exist, then a third determination module 520 determines whether all intervening changes from the master copy have been obtained and instantiate into the user copy. For example, the third determination module 520 may determine whether any intervening changes have been made to the master copy since the most recent update. In one embodiment, the third determination module 520 compares a version number of the current version of the master copy with a version number of the version of the master copy represented by the most recently received update.

[0064] If the third determination module 520 determines additional intervening changes exist (i.e., that the most recently received update does not reflect the current state of the master copy), then the synchronize process 500 returns to the obtain operation 504 and the synchronize process 500 begins again. If the third determination module 520 determines no additional intervening changes exist (i.e., that the most recently received update reflects the current state of the master copy), however, then an indicate operation 522 provides an indication the editing conflict has been resolved. For example, in one embodiment, the indicate operation 522 may display a message to the user indicating that all editing conflicts have been resolved. In another embodiment, the indicate operation 522 may display a graphic, icon, or other indicia to the user indicating the editing conflicts have been resolved. A store operation 524 overwrites the master copy of the data file with the user copy. The synchronize process 500 completes and ends at a stop module 526.

[0065] FIG. 6 is a flowchart illustrating an operational flow for an example editing process 600 by which the continuation operation 516 of FIG. 5 may be implemented. The editing process 600 initializes and begins at a start module 602 and proceeds to a receive operation 604. The receive operation 604 obtains editing instructions from the user. For example, the receive operation 604 may receive user input through an input device, such as input device 430 of FIG. 4.

[0066] The editing instructions indicate changes to content and/or metadata of the merged version of the user copy of the data file. For example, the editing instructions may indicate a data unit (e.g., a paragraph, a column, a table, a slide, a graphic, etc.) of the new data file should be added, deleted, or revised. The editing instructions also may indicate a change in metadata (e.g., a change in content locks, etc.). An implement operation 606 performs the editing operation on the new copy of the document.

[0067] A determination module 608 determines whether the user has indicated a desire to stop editing. For example, in a first embodiment, the determination module 608 may determine the user has chosen to resolve editing conflicts. In another embodiment, the determination module 608 determines the user has chosen to close the new copy of the data file or to synchronize the new copy with the master copy of the data file. In another embodiment, the determination module 608 may determine the user has chosen to continue editing the new copy of the data file.

[0068] If the determination module 608 determines the user has chosen to continue editing the data file, then the editing process 600 cycles back to the receive operation 604 and begins again. If the determination module 608 determines the user has chosen to stop editing the data file, however, then a save operation 610 stores any changes made by the user. Generally, the save operation 610 stores the merged version of the data file including any changes made by the user in a location other than the master copy of the data file. In one embodiment, the save operation 610 stores the merged version of the data file in local memory (e.g., a local cache). In another embodiment, the save operation 610 may store the merged version of the data file on a storage device separate from the master copy. The editing process 600 completes and ends at a stop module 612.

[0069] FIG. 7 is a flowchart illustrating an operational flow for an example review process 700 by which one or more editing conflicts may be reviewed and optionally resolved. For example, the review process 700 is one example process for implementing the review process 518 of FIG. 5. The review process 700 initializes and begins at a start module 702 and proceeds to a display operation 704. The display operation 704 presents the editing conflicts to the user. In one embodiment, the editing conflicts are provided within the context of the merged version of the data file.

[0070] In some embodiments, the display operation 704 displays all conflicting edits from both the master copy and the user copy. For example, in one embodiment, the display operation 704 may display content resulting from the conflicting editing operations as merged into the user copy of the data file. In another embodiment, the display operation 704 may display content resulting from the conflicting editing operations as merged into the master copy of the data file. In another embodiment, the display operation 704 may display content resulting from the conflicting editing operations as merged into the version of the data file prior to any intervening changes.

[0071] In one embodiment, if conflicting editing operations are made to the same object (e.g., a character, a word, a paragraph, a graphic, a slide, a cell, a row, a column, etc.) within the data file, then the display operation 704 will display the object transformed by both editing operations. In another embodiment, the display operation 704 will display two copies of the object, one copy representing the version of the object found in the master copy of the data file and the other copy representing the version of the object found in the user copy of the data file.

[0072] In one embodiment, the display operation 704 displays the conflicting content as annotations to the non-conflicting content of the merged version of the data file. For example, in one embodiment, conflicting content added to the data file, either in the master copy or in the user copy, may be shown added to the non-conflicting content and annotated to indicate the conflicting content was inserted in one version of the data file. In another embodiment, conflicting content removed from the data file, either in the master copy or in the user copy, may be shown added to the non-conflicting content and annotated to indicate the conflicting content was deleted in one version of the data file. Additional disclosure relating to the display of conflicting edits will be provided herein.

[0073] An obtain operation 706 receives authoring instructions from the user. For example, the obtain operation 706 may receive user input through an input device, such as input device 430 of FIG. 4. A first determination module 708 determines whether the user has provided instructions to return to continuing to edit the data file freely instead of continuing to review editing conflicts. For example, in one embodiment, the first determination module 708 may determine the user has selected a toggle interface, which will be disclosed in greater detail herein. In another embodiment, the first determination module 708 may determine the user has provided instructions to accept or reject one or more of the conflicting editing operations.

[0074] If the first determination module 708 determines the user has provided instructions to return to editing, then the review process 700 completes and ends at a stop module 714 even if identified editing conflicts remain unresolved. If the first determination module 708 determines the user has not provided instructions to return to editing, however, then an implement operation 710 performs the user instructions. In general, the implement operation 710 may accept or reject each editing operation in accordance with the instructions provided by the user. For example, in one embodiment, the implement operation 710 may instantiate a conflicting editing operation into the merged version of the data file if the user provides instructions to accept the editing operation. In another embodiment, the implement operation 710 may remove the effects of a rejected editing operation from the merged version of the data file. Additional details regarding performance of the implement operation 710 are provided herein.

[0075] A second determination module 712 determines whether any identified editing conflicts remain unresolved. In one embodiment, the second determination module 712 does not determine whether any new editing conflicts have been created since the most recent identification of editing conflicts (e.g., through updates received from the master copy, through additional editing operations performed by the user, etc.). Rather, the second determination module 712 determines whether any of the editing conflicts already identified remain unresolved. In another embodiment, the second determination module 712 determines whether any new editing conflicts exist.

[0076] If the second determination module 712 determines at least one identified editing conflict remains unresolved, then the resolution process 700 cycles back to the first determination module 708. If the second determination module 712 determines the identified editing conflicts have been resolved, however, then the resolution process 700 completes and ends at the stop module 714.

[0077] In general, the user interface displayed by the authoring application on the user device, such as authoring application 130 on user device 110 of FIG. 2, may change based on whether one or more editing conflicts have been identified by the authoring application. In some embodiments, the authoring application may display alerts indicating the existence of editing conflicts to the user. For example, in one embodiment, a status bar may indicate the existence of editing conflicts. In another embodiment, a message box providing indicia indicating consequences of the editing conflicts may be displayed to the user. For example, the message box may indicate content cannot be synchronized until the identified editing conflicts are resolved.

[0078] In addition, when an editing conflict is identified, the user interface of the authoring application may be configured to toggle between showing editing conflicts and hiding editing conflicts at the discretion of the user. In general, the user interface of the authoring application enables free editing of the data file both when editing conflicts are shown and when editing conflicts are hidden. The user interface enables additional functionality enabling review and resolution of editing conflicts when the user chooses to view editing conflicts.

[0079] FIGS. 8-10 are schematic block diagrams illustrating example user interfaces that an authoring application may display to a user to enable authoring and synchronization of a data file. FIG. 8 illustrates an example user interface 800 that is displayed to a user when no unresolved editing conflicts have been identified. For example, the authoring application may display user interface 800 when the editing operation 304 of FIG. 3 is being performed.

[0080] The user interface 800 includes an authoring window 810 in which the user copy of the data file may be displayed. In one embodiment, the authoring window 810 includes an editing area 812 in which the content of the user copy is displayed. Metadata of the data file (e.g., content locks) also may be displayed in the editing area 812. The user of the authoring application may interact with the content in the editing area 812 to add, delete, or revise the displayed content.

[0081] FIG. 9 illustrates an example user interface 850 displayed to a user when at least one editing conflict has been identified. For example, the authoring application may display the user interface 850 when the user chooses to continue editing the data file instead of resolving conflicts (e.g., see first determination module 516 of FIG. 5). In one embodiment, the user interface 850 is displayed when the editing process 600 of FIG. 6 is being performed.

[0082] The user interface 850 includes the authoring window 810 of FIG. 8 having the editing area 812 in which the content and metadata of the data file may be displayed and/or edited. The user interface 850 also includes a toggle interface 815. In general, selection of the toggle interface 815 by the user will cause the user interface to display identified editing conflicts. In one embodiment, selection of the toggle interface 815 also will activate a conflict resolution interface.

[0083] FIG. 10 illustrates an example user interface 900 that is displayed to a user when at least one editing conflict has been identified and the user has chosen to review the identified editing conflict. The user interface 900 also may enable resolution of the identified editing conflicts. For example, the authoring application may display the user interface 900 when the user chooses to review editing conflicts (see first determination module 514 of FIG. 5). Accordingly, in one embodiment, the user interface 900 is displayed when the review process 700 of FIG. 7 is being performed.

[0084] The user interface 900 also includes a toggle interface 915 by which the user may hide identified editing conflicts. In one embodiment, selection of the toggle interface 915 also will deactivate the conflict resolution interface 920. For example, selection of the toggle interface 915 may cause the user interface 850 of FIG. 9 to be displayed to the user instead of the user interface 900.

[0085] The user interface 900 includes an authoring window 910 having an editing area 912 in which the content and metadata of the data file may be displayed and/or edited. The editing area 912 also displays identified editing conflicts or content resulting from identified editing conflicts. For example, in one embodiment, the editing area 912 may display a merged version of the data file generated by merging the user copy with the master copy.

[0086] In some embodiments, the identified editing conflicts displayed within the editing area 912 are indicated via annotations to the content resulting from the conflicting editing operations. In one embodiment, the annotations indicate how of the resulting content conflicts. For example, in one embodiment, a section of content (e.g., a word, a paragraph, a table, a column, a graphic, etc.) may be annotated to indicate the section was inserted, deleted, and/or revised in the user copy and/or the master copy. Example annotations indicating such insertion, deletion, and/or revision may include a predetermined color, a strikethrough, underlining, a predetermined opacity, highlighting, or other such indicia.

[0087] The user interface 900 also can include a summary window 920 in which the authoring application may communicate information about identified editing conflicts to the user. In some embodiments, the summary window 920 can include a summary information area 922. For example, the summary information area 922 may display the number of editing conflicts identified by the authoring application and remaining unresolved. The summary information area 922 also may indicate the number of mergeable and unmergeable editing conflicts. In one embodiment, the summary information area 922 may be refreshed (i.e., updated) as the editing conflicts are resolved.

[0088] In some embodiments, the summary window 920 also may display a listing 924 of any unresolved editing conflicts. For example, in one embodiment, all conflicting content (e.g., content added, deleted, and/or revised in the user copy or the master copy) is listed within the summary window 920. In one embodiment, the listed content may be annotated to indicate an origin of the content (e.g., the user copy or the master copy). In another embodiment, the listed content may be annotated to indicate a type of editing action (e.g., revision, insertion, deletion, etc.) yielding the content.

[0089] The user interface 900 may be configured to enable a user to resolve the displayed editing conflicts. For example, the user interface 900 may enable the user to provide resolution instructions for one or more displayed editing conflicts. In one embodiment, the user interface 900 enables the user to provide resolution instructions for the displayed editing conflicts in accordance with a sequence. In another embodiment, the user interface 900 enables the user to provide resolution instructions for any displayed editing conflict selected by the user.

[0090] Examples of resolution instructions include an accept instruction by which the user indicates a given editing operation should be instantiated into the merged version of the data file and a reject instruction by which the user indicates the editing operation should not be performed on the merged version of the data file. Other examples of resolution instructions may include a next instruction by which the user proceeds to a subsequent editing conflict without resolving the currently selected editing conflict and a back instruction by which the user returns to a previous editing conflict without resolving the currently selected editing conflict.

[0091] In some embodiments, the user may provide a resolution instruction by interacting with one or more resolution interfaces 930. Non-limiting examples of resolution interfaces include buttons (e.g., an accept button, a reject button, a next button, etc.), drop-down menus, tabs, and other such interface tools. In one embodiment, the user interface 900 displays one or more resolution interfaces to the user when the editing conflicts are displayed. In another embodiment, the user interface 900 displays one or more resolution interfaces to the user when conflicting content is selected by the user.

[0092] For example, the user may provide a resolution instruction by selecting (e.g., via a cursor or other input interface) content within the editing area 912 that results from one of the conflicting editing operations and interacting with a resolution interface. In other embodiments, the user may provide a resolution instruction by selecting an editing operation displayed within the summary window 920 and interacting with a resolution interface to provide or select a resolution instruction.

[0093] For example, in FIG. 11, the user interface 900 includes a resolution interface 930 that is displayed when an editing conflict 925 is selected from the listing 924 in the summary window 920. The resolution interface 930 shown in FIG. 11 includes an accept button 932 and a reject button 934. In other embodiments, however, the resolution interface 930 may include any desired interface tool. In the example shown in FIG. 11, the resolution interface 930 is displayed as a pop-up window. In other embodiments, however, the resolution interface 930 may include a toolbar or section thereof, a drop down menu or portion thereof, or other such display interface.

[0094] FIG. 12 is a flowchart illustrating an operational flow for an example implementation process 1000 by which the authoring application implements resolution instructions provided by the user. The implementation process 1000 initializes and begins at a start module 1002 and proceeds to an obtain operation 1004. The obtain operation 1004 receives a resolution instruction from the user for resolving conflicting content. For example, the obtain operation 1004 may determine a given section of conflicting content has been selected by the user and a particular button, menu option, or other resolution interface tool has been selected by the user.

[0095] A determination module 1006 determines whether the resolution instruction indicates the selected conflicting content is to be accepted or rejected. If the determination module 1006 determines the obtain operation 1004 receives an accept instruction, then the implementation process 1000 proceeds to an accept operation 1008. If the determination module 1006 determines the obtain operation 1004 receives a reject instruction, however, then the implementation process 1000 proceeds to a reject operation 1010. Details regarding example processes for implementing the accept and reject operations 1008, 1010 are discussed below with reference to FIGS. 13 and 14. The implementation process 1000 completes and ends at a stop module 1012.

[0096] FIG. 13 is a flowchart illustrating an operational flow for an example accept process 1100 by which an accepted editing operation may be instantiated into the merged version of the data file. The accept process 1100 initializes and begins at a start module 1102 and proceeds to a clean operation 1104. The clean operation 1104 removes any annotations or other indicia from the accepted content. For example, any underlining, strikethroughs, coloring, or other such annotations of the accepted content are removed from the merged version of the data file.

[0097] A determination module 1106 determines whether or not the accepted content resulted in content and/or formatting being added into the data file. For example, in one embodiment, the determination module 1106 determines whether or not the accepted content includes a character, word, table, column, graphic, or other data unit to the data file. In another embodiment, the determination module 1106 determines whether the accepted content added any formatting (e.g., bold, underlining, font color, highlighting, etc.) to the data file.

[0098] If the determination module 1106 determines the accepted editing operation resulted in content and/or formatting being added, then the accept process 1100 complete and ends at the stop module 1110. If the determination module 1106 determines the accepted editing operation removes content and/or formatting from the data file, however, then a remove operation 1108 deletes the content and/or formatting from the merged version of the data file. The accept process 1100 completes and ends at the stop module 1110 after completion of the remove operation 1108.

[0099] FIG. 14 is a flowchart illustrating an operational flow for an example reject process 1200 by which a rejected editing operation may be removed from or undone within the merged version of a data file. The reject process 1200 initializes and begins at a start module 1202 and proceeds to a clean operation 1204. The clean operation 1204 removes any annotations or other indicia from the content resulting from the rejected editing operation. For example, any underlining, strikethroughs, coloring, or other such annotations of the content are removed from the merged version of the data file.

[0100] A determination module 1206 determines whether or not the rejected editing operation would have resulted in content and/or formatting being added to the data file. For example, in one embodiment, the fourth determination module 1206 determines whether or not the rejected editing operation would have added a character, word, row, cell, or other data unit or content object to the data file. In another embodiment, the determination module 1206 determines whether the rejected editing operation would have added any formatting (e.g., bold, underlining, font color, highlighting, etc.) to the data file.

[0101] If the determination module 1206 determines the rejected editing operation would not have resulted in content and/or formatting being added to the data file, then the reject process 1200 complete and ends at the stop module 1210. If the determination module 1206 determines the rejected editing operation would have added content and/or formatting, however, then a remove operation 1208 deletes the content and/or formatting from the merged version of the data file. The reject process 1200 completes and ends at the stop module 1220 after completion of the remove operation 1208.

[0102] The principles of the present disclosure can be better understood by walking through example applications. In a first example application, FIGS. 15-22 illustrate changes to an example user interface displayed by an authoring application as a first user is editing a user copy of a data file online. In a second example application, FIGS. 23 - 29 illustrate changes to an example user interface displayed by an authoring application as a first user is editing a user copy of a data file offline.

[0103] In FIG. 15, a user interface 2000 of an authoring application includes a display window 2010 including an editing area 2012 displaying content of the user copy of the data file, a command toolbar 2016 providing command options, and a status bar 2018 indicating a status of the user copy of the data file. For example, the status bar 2018 may indicate when content updates are available for instantiation into the user copy of the data file. The status bar 2018 also may indicate when editing conflicts have been identified. In the example shown, the status bar 2018 indicates no content updates are available from the master copy of the data file.

[0104] The authoring application enables editing of the user copy of the data file, for example, using the example authoring process 300 of FIG. 3. In the example shown, the authoring application enables the first user to edit the user copy of the data file freely (see edit operation 304 of FIG. 3) within the editing area 2012 of the display window 2010. In FIG. 15, the first user is editing a first data unit of content displayed in the editing area 2012. The first user has arranged a cursor 2019 over the first data unit to enable editing of the first data unit. A second user has obtained a lock 2022 on a second data unit of content within the editing area 2012. The lock 2022, which is displayed within the editing area 2012, inhibits the first user from freely editing the second data unit.

[0105] In FIG. 16, the first user has edited the first data unit to change the word "dog" to "doe". In one embodiment, changing the word "dog" to "doe" includes providing an editing operation to delete the word "dog" and an editing operation to insert the word "doe". In another embodiment, changing the word "dog" to "doe" includes providing an editing operation to delete the character "g" and an editing operation to insert the character "e". In other embodiments, other editing operations may be utilized to achieve the same results.

[0106] In one embodiment, editing of the first data unit initiates a transmission to the master copy of the data file of a request by the first user to lock the first data unit. In another embodiment, the first user may provide express instructions to lock the first data unit. For example, the first user may select the first data unit and select a lock option on the user interface 2000. Accordingly, a lock 2024 owned by the first user has been placed around the first data unit in FIG. 16. In the example shown, the lock 2024 is distinguished from the lock 2022 owned by the second user. For example, the lock 2024 owned by the first user is displayed in dotted lines and the lock 2022 owned by the second user is displayed in solid lines. Distinguishing locks owned by the first user from locks owned by other users may enable the first user to know which data unit the first user has locked without deterring the first user from editing the data unit. In one embodiment, each user may have a distinct lock (e.g., each user's lock may have a distinct color, shading, formatting, etc.).

[0107] In FIG. 16A, the authoring application obtains updates indicating intervening changes made to the master copy of the data file (see receive operation 306 of FIG. 3). In one embodiment, the authoring application receives the update after a predetermined interval of time elapses. In another embodiment, the authoring application receives the update in response to a request to synchronize the user copy with the master copy. For the purposes of this example application, the authoring application is assumed to have received the update automatically after a predetermined time interval. The status bar 2018 of the display window 2010 has been updated in FIG. 16A to indicate an update has been received.

[0108] In one embodiment, the authoring application receives a metadata update from the master copy. In another embodiment, the authoring application receives a content update from the master copy. In the example application, the authoring application receives both a metadata update and a content update. The metadata update indicates the second user has released the lock 2022 on the second data unit and has obtained a lock (see lock 2028 of FIG. 17) on the first data unit. In this example, the second user synchronized a lock request for the first data unit with the master copy before the first user. Accordingly, the second user was awarded the lock on the first data unit. The metadata update also indicates a third user has obtained a third lock 2026 on the third data unit. The content update indicates the second user has edited the first data unit to change the word "lazy" to "lively".

[0109] When the authoring application receives the update, the authoring application determines whether any editing conflicts exist between the user copy and the received update. If no editing conflicts exist, then the authoring application instantiates the metadata updates automatically (see update operation 308 of FIG. 3). In the example application, the metadata update indicates the third user has established a lock on the third data unit. Accordingly, the third lock 2026 is shown around the third data unit in FIG. 16A.

[0110] In one embodiment, if a metadata update conflicts with a user change, however, then the metadata update is not instantiated until the user attempts to synchronize the user copy with the master copy. In the example application, the metadata update indicates the second user has a lock on the first data unit, which conflicts with the first user's request to lock the first data unit. Accordingly, the second user's lock on the first data unit is not instantiated into the user copy automatically. In another embodiment, a release of a content lock is not instantiated until any identified editing conflicts are resolved. Accordingly, the second user's release of the lock 2022 on the second data unit is not instantiated into the user copy automatically.

[0111] If no editing conflicts exist, then the authoring application also enables the user to determine when the content updates should be instantiated (e.g., merged) into the user copy of the data file (see update operation 308 of FIG. 3). For example, the authoring application may provide an instantiation interface by which the first user may provide instructions to instantiate the update. Non-limiting examples of instantiation interfaces include buttons, menu options, and other interface tools.

[0112] If the authoring application identifies an editing conflict, however, then the authoring application inhibits the user from instantiating the updates into the user copy. In one embodiment, the authoring application does not display an instantiation interface. In another embodiment, the authoring application does not display an indication of the availability of the update. In the example application, the metadata update conflicts with a user change. Accordingly, the authoring application does not provide the first user with an opportunity to instantiate the content update.

[0113] In FIG. 17, the first user instructs the authoring application to attempt to synchronize the user copy of the data file with the master copy (see synchronize operation 310 of FIG. 3). In one embodiment, the authoring application implements the synchronize process 500 shown in FIG. 5. The authoring application obtains an updates from the master copy if appropriate (see obtain operation 504 of FIG. 5) and determines whether any editing conflicts result from the update (see identify operation 506 of FIG. 5). In the example shown, the authoring application does not obtain any new updates (i.e., no intervening changes have been made to the master copy since the last update). The authoring application determines that intervening changes represented by the previously received update conflict with user changes.

[0114] The authoring application then merges the user copy of the data file with the master copy of the data file (see merge operations 508 of FIG. 5) and displays the merged copy of the data file to the first user. In the example shown, the authoring application displays the merged version of the data file in the editing area 2012 in FIG. 17. In general, the authoring application may integrate the user and master copies using any desired merge technique. For example, in one embodiment, the authoring application displays all content added to, deleted from, and/or revised in the user copy of the data file and the master copy of the data file.

[0115] In another embodiment, the authoring application may determine the version of the data file represented by the master copy and may perform any content or formatting additions, deletions, and/or revisions from the user copy on unlocked data units of the master copy version of the data file (i.e., instantiates mergeable conflicts). For unmergeable conflicts, such as user edits to data units locked on the master copy (e.g., user edits performed prior to receiving the metadata update indicating the lock), the authoring application may add a duplicate data unit adjacent the locked data unit and may revise the duplicate data unit based on the revisions made in the user copy (e.g., change content, add a content lock, etc.). Accordingly, the data unit remains locked and unchanged in accordance with the state of the master copy. However, changes to the data unit by the first user are retained in the merged version of the data file until the first user chooses to remove them.

[0116] In the example shown in FIG. 17, the first data unit is shown locked by the second user (see lock 202) and changed by editing operations performed by the second user. For example, the second occurrence of the word "lazy" has been changed to "lively". A duplicate of the first data unit has been generated and arranged adjacent the first data unit. The duplicate is locked to the first user (see lock 2024) and includes revisions made by the first user prior to the attempt to synchronize (e.g., the second occurrence of "dog" has been changed to "doe"). The second data unit is still locked to the second user in FIG. 17. In one embodiment, the authoring application only releases locks when the user copy is synchronized with the master copy. In another embodiment, the authoring application only releases locks when the locked data unit is fully synchronized. In other embodiments, the authoring application may indicate the lock has been released.

[0117] Because the authoring application has identified editing conflicts, the authoring application does not continue to save to the master copy (see first determination module 510 of FIG. 5). Rather, the authoring application displays one or more alert messages to notify the first user of the existence of the editing conflicts (see alert operation 512 of FIG. 5). For example, in FIG. 17, an alert 2017 is displayed in the display window

2010 to inform the first user of the presence of editing conflicts. In the example shown, the status bar 2018 also has been updated to indicate the presence of editing conflicts. In other embodiments, the status bar 2018 may be updated to indicate a number of updates that have not yet been instantiated, a number of editing conflicts identified, or other such information. In still other embodiments, other types of alerts or indicia may provide notice of the existence of editing conflicts to the first user.

[0118] The authoring application also presents a toggle interface 2015 to the first user in FIG. 17 to enable the user to choose between continuing to edit the data file without viewing conflicts and reviewing the editing conflicts (see second determination module 514). The first user may choose to continue editing the data file without reviewing the editing conflicts (see continue operation 516 of FIG. 5). The first user also may choose to select the toggle interface 2015 to reveal the editing conflicts (see review operation 518 of FIG. 5). In one embodiment, selection of the toggle interface 2015 also activates a summary window 2030. The first user may continue to edit the merged version of the data file regardless of whether the first user chooses to view the editing conflicts.

[0119] One example process by which the authoring application may implement the review operation 518 of FIG. 5 includes the review process 700 of FIG. 7. When the user selects the toggle interface 2015 (FIG. 17), the authoring application presents the editing conflicts to the user (see display operation 704 of FIG. 7). For example, in one embodiment, selecting the toggle interface 2015 splits the display window 2010 between the editing area 2012 and a summary window 2030 in which the editing conflicts are listed (see list 2034 of FIG. 18) and summary information about the conflicts may be displayed (see summary area 2032 of FIG. 18).

[0120] In general, the editing area 2012 displays the merged version of the data file and enables the user to freely edit the merged version. In some embodiments, the merged version of the data file displayed in the editing area 2012 is annotated to indicate which portions of content are in conflict. For example, in one embodiment, the merged version of the data file is annotated to indicate whether the conflicting content resulted from insertions (e.g., via underlining or other formatting, text color, highlighting, or other such indicia) or deletions (e.g., via strikethroughs or other formatting, text color, highlighting, or other such indicia).

[0121] In the example shown, revisions to content are shows as a series of deletions and insertions. In other embodiments, however, revisions to content may be distinctly annotated. In other embodiments, the metadata (e.g., locks) also may be annotated to indicate which portions of metadata conflict. In the example shown in FIG. 18, the first data unit does not contain any editing conflicts since the first data unit is locked on the server to the second user. Accordingly, the first data unit is not annotated. The duplicate data unit, which is shown in FIG. 18 as locked to the first user, is underlined to indicate the data unit has been inserted into the data file.

[0122] The conflicting editing operations also are displayed by the summary window 2030. The summary area 2032 of the resolution interface indicates the number of conflicting editing operations contained within the merged version of the data file. In the example shown, the summary area 2032 indicates one conflicting editing operation has been identified. The listing 2034 of the summary window 2030 displays the conflicting content. In one embodiment, the listing 2034 displays the conflicting content separate from the non-conflicting content. In another embodiment, the listing 2034 annotates the conflicting content to indicate whether the content as inserted and/or deleted (e.g., see indicia 2036 of FIG. 18). In another embodiment, the listing 2034 annotates the conflicting content to indicate the origin (e.g., user copy or master copy) of the conflicting content (e.g., see indicia 2038 of FIG. 18).

[0123] The first user of the authoring application may interact with either the editing area 2012 or the summary window 2030 (FIG. 18) to provide instructions (see obtain operation 706 of FIG. 7) either to return to editing the data file or to resolve any of the editing conflicts displayed (see first determination module 708 of FIG. 7). For example, in one embodiment, the user may select the toggle interface 2015 again to hide the editing conflicts. Choosing to hide the conflicts removes the annotations from the editing conflicts. In one embodiment, choosing to hide the editing conflicts will cause the user interface 2000 of FIG. 17 to be displayed. The synchronize process 700 would end (see stop module 714 of FIG. 7), the authoring application would determine that at least one editing conflict remained unresolved (see first determination module 510 of FIG. 5), and the first user would again be able to choose between freely editing the data file and resolving conflicts.

[0124] Alternatively, the first user may provide instructions to accept or reject conflicting content or portions thereof. In one embodiment, the first user may select content displayed within the editing area 2012 of the display window 2010 and may select an option on a resolution interface to provide instructions to resolve the editing operation. For example, the first user may select the content on which instructions are to be provided and may select a menu option from a resolution mention 2040 (see FIG. 19). In other embodiments, the first user may provide the resolution instruction using another type of resolution interface tool.

[0125] In the example shown in FIG. 19, the first user selects (see selection indicia 2013) a first portion of the conflicting content and selects a reject option on the resolution menu 2040 via the cursor 2019. Accordingly, the authoring application determines the user has not provided instructions to return to editing (see first determination module 708 of FIG. 7) and performs the resolution instruction (see implement operation 710 of FIG. 7). In other embodiments, the first user may have selected all or a different portion of the conflicting content.

[0126] One example process by which the authoring application may perform the rejection of the selected portion of the conflicting content is the reject process 1200 of FIG. 14. The authoring application removes the annotations from the selected conflicting content in the editing area 2012 of the display window 2010 (see clean operation 1204 of FIG. 14). The authoring application also determines (see determination module 1206 of FIG. 14) the rejected editing operation resulted in the selected conflicting content being added into the data file. Accordingly, the authoring application removes the selected conflicting content from the merged version of the data file (see remove operation 1208 of FIG. 14). If the editing operation had resulted in content being deleted, then the reject process 1200 would have ended without removing the selected conflicting content from the data file.

[0127] Returning to the review process 700 of FIG. 7, the authoring application determines that at least one of the identified editing conflicts remains unresolved (see second determination module 712 of FIG. 7), thereby restarting the review process 700. By restarting the review process 700, the first user may choose to continue resolving editing conflicts (e.g., using the review process 700 of FIG. 7) while the user interface is configured in the conflict resolution mode. Alternatively, the first user may choose to hide the editing conflicts at any time by selecting the toggle interface 2015.

[0128] FIG. 20 shows the user interface 2000 after the first user has rejected the selected conflicting content (see display operation 704 of FIG. 7). The rejected conflicting content has been removed from the editing area 2012. Furthermore, the authoring application updates the summary window 2030 to reflect the resolution of the conflicting content by removing the rejected content from the listing 2034 within the summary window 2030. Because the conflicting content was not completely resolved, the summary area 2032 remains unchanged.

[0129] Continuing with the review process 700, the authoring application obtains another set of instructions from the first user (see obtain operation 706 of FIG. 7). In this example application, the authoring application obtains instructions to resolve the remaining conflicting content. For example, FIG. 21 illustrates another resolution interface 2050 with which the first user may provide resolution instructions to the authoring application. The resolution interface 2050 is a menu listing resolution options (e.g., accept, reject, etc.). The menu is arranged adjacent the selected conflicting content. In one embodiment, the authoring application displays the resolution interface 2050 when the user selects conflicting edits from the listing 2034 within the summary window 2030. In another embodiment, the authoring application may display the resolution interface 2050 within the editing area 2012 when conflicting content is selected within the editing area 2012. In other embodiments, however, the first user may use any desired type of resolution interface to provide resolution instructions to the authoring application.

[0130] In FIG. 21, the user selects (e.g., via a right mouse click, via a left mouse click, hovers over, or otherwise selects via another input device) the remaining conflicting content within the listing 2034 of the summary window 2030 and selects the accept option on the resolution interface 2050 (e.g., using the cursor 2019). Accordingly, the authoring application determines the first user did not provide instructions to hide the editing conflicts and return to editing (see determination module 708 of FIG. 7) and implements the accept instruction (see implement operation 710 of FIG. 7).

[0131] One example process by which the authoring application may accept the selected conflicting content includes the accept process 1100 of FIG. 13. The authoring application removes any annotations from the selected conflicting content (see clean operation 1104 of FIG. 13). The authoring application determines the selected conflicting content was inserted into the data file (see determination module 1106 of FIG. 13). Accordingly, the accept process 1100 completes and ends (see stop module 1110 of FIG. 13). If the authoring application had determined the conflicting content was deleted from the data file, then the authoring application would have deleted the conflicting content from the data file (see remove operation 1108 of FIG. 13).

[0132] Returning to the review process 700 of FIG. 7, the authoring application determines no more editing conflicts remain unresolved (see second determination module 712). Accordingly, the review process 700 completes and ends (see stop module 714 of FIG. 7), thereby completing the review operation 514 of the synchronize process 500 of FIG. 5. The synchronize process 500 proceeds back to the first determination module 510 at which the authoring application determines whether any identified editing conflicts remain unresolved.

[0133] Since the first user resolved all identified editing conflicts in this example application, the authoring application determines that no editing conflicts remain unresolved. Accordingly, the authoring application checks with the master copy to determine whether additional updates are available (see the second determination module 520 of FIG. 5). If additional updates have become available, then the synchronization process 500 cycles back to obtain and merge the new updates. The new update includes any intervening changes made since the previous update was obtained. Any conflicts stemming from the new update are resolved as discussed above. This process of obtaining updates and resolving any identified conflicts repeats until the first user has resolved all editing conflicts between the user copy of the data file and the current state of the master copy of the data file.

[0134] When the authoring application determines no editing conflicts exist (see first determination module 510 of FIG. 5) and no new updates are available (see second determination module 520 of FIG. 5), the authoring application sends an update from the user copy to the master copy (see store operation 522 of FIG. 5) to complete synchronization of the user and master copies. For example, in one embodiment, the authoring application may send the user copy in its entirety to a storage device with instructions to overwrite the master copy. In another embodiment, the authoring application may send an incremental update indicating how the user copy differs from the master copy.

[0135] The authoring application alerts the first user when the authoring application successfully uploads the update to the master copy (see indicate operation 524 of FIG. 5). For example, in FIG. 22, the status bar 2018 has been updated to indicate the update was successfully transmitted. In some embodiments, the authoring application also may display an alert window 2060 indicating the success of the upload. In one embodiment, the alert window 2060 may provide further explanation, e.g., indicating consequences of the upload. In the example shown in FIG. 22, the summary window 2030 is removed from the user interface since no editing conflicts are identified. In other embodiments, however, the summary window 2030 may remain until toggled by the first user.

[0136] The second example application, in which the first user is editing a data file offline, is provided in FIGS. 23-29. In FIG. 23, a user interface 2200 of an authoring application includes a display window 2210 including an editing area 2212 displaying content of the user copy of the data file, a command toolbar 2216 providing command options, and a status bar 2218 indicating a status of the user copy of the data file. For example, the user interface 2200 may be the same as user interface 2000 of FIG. 15.

[0137] In the example shown, the status bar 2218 indicates the first user is editing a user copy of a data file offline (i.e., is not communicatively connected to a storage device storing a master copy of the data file). Another user has a lock 2222 on the first data unit within the editing area 2212. Accordingly, the authoring application inhibits the first user from editing the first data unit.

[0138] The authoring application enables editing of the user copy of the data file, for example, using the example authoring process 300 of FIG. 3. In the example shown, the authoring application enables the first user to edit the user copy of the data file freely (see edit operation 304 of FIG. 3) within the editing area 2012 of the display window 2010. FIG. 24 illustrates editing changes made by the first user to the user copy of the data file. In particular, the first user has added two sentences to the third data unit. In other embodiments, however, the first user may have added, revised, and/or deleted any content or metadata within the user copy of the data file.

[0139] Because the first user is editing the user copy offline, the authoring application cannot synchronize a lock request for the third data unit. In one embodiment, the authoring application stores the lock request until the user logs online. In such an embodiment, the authoring application may display a lock around the third data unit. In another embodiment, the authoring application does not attempt to lock the third data unit. In one embodiment, the authoring application alerts the first user that a lock request cannot be synchronized and, accordingly, that editing the data unit may result in editing conflicts. For the purposes of this example application, the authoring application stores the lock request for later synchronization with the master copy.

[0140] In FIG. 25, the authoring application obtains updates indicating intervening changes made to the master copy of the data file (see receive operation 306 of FIG. 3). In one embodiment, the authoring application receives the update after a predetermined interval of time elapses. In another embodiment, the authoring application receives the update in response to a request to synchronize the user copy with the master copy. For the purposes of this example application, the authoring application is assumed to have received the update automatically after a predetermined time interval. The status bar 2218 of the display window 2210 has been updated in FIG. 25 to indicate an update has been received.

[0141] When the authoring application receives the update, the authoring application determines whether any editing conflicts exist between the user copy and the received update. If no editing conflicts exist, then the authoring application instantiates the metadata updates automatically (see update operation 308 of FIG. 3). For the purposes of this example application, the authoring application is assumed to have received a content update indicating changes made to the third data unit, which conflicts with the first user's lock request for the third data unit. However, the update does not indicate a lock around the third data unit (e.g., the other user may have locked the third data unit when making the edit and subsequently released the lock). Accordingly, the authoring application does not provide the first user with an instantiation interface or otherwise inhibits instantiation of the update.

[0142] In FIG. 26, the first user connects to the storage device (i.e., logs online) and instructs the authoring application to attempt to synchronize the user copy of the data file with the master copy (see synchronize operation 310 of FIG. 3). In one embodiment, the authoring application implements the synchronize process 500 shown in FIG. 5. The authoring application obtains an updates from the master copy if appropriate (see obtain operation 504) and determines whether any editing conflicts result from the update (see identify operation 506). In the example shown, the authoring application does not obtain any new updates (e.g., no intervening changes have been made to the master copy since the last update). The authoring application determines that intervening changes represented by the previously received update (e.g., the added content to the third data unit) conflict with user changes (e.g., the lock request for the third data unit).

[0143] The authoring application then merges the user copy of the data file with the master copy of the data file (see merge operations 508 of FIG. 5) and displays the merged copy of the data file to the first user. In the example shown, the authoring application displays the merged version of the data file in the editing area 2212 in FIG. 26. As noted above, the authoring application may integrate the user and master copies using any desired merge technique. In the example shown, the authoring application has added new content from the master copy to the first and third data units of the user copy.

[0144] The first set of new content, "TBD," which was added to the first data unit, does not result in an editing conflict. The first user did not edit the first data unit. The second set of new content, "Jump, fox, jump!", which was added to the third data unit, results in an editing conflict since the first user attempted to lock the third data unit. The other user was able to edit the third data unit since the first user's lock request has not yet been synchronized with the master copy prior to the edit. However, since the update does not indicate the third data unit is locked by another user, the first user's lock is maintained around the third data unit.

[0145] Because the authoring application has identified an editing conflict, the authoring application does not continue to save to the master copy (see first determination module 510 of FIG. 5). Rather, the authoring application displays one or more alert messages to notify the first user of the existence of the editing conflicts (see alert operation 512 of FIG. 5). For example, in FIG. 26, an alert 2217 is displayed in the display window 2210 to inform the first user of the presence of editing conflicts. In the example shown, the status bar 2218 also has been updated to indicate the presence of editing conflicts. In other embodiments, the status bar 2218 may be updated to indicate a number of updates that have not yet been instantiated, a number of editing conflicts identified, or other such information. In still other embodiments, other types of alerts or indicia may provide notice of the existence of editing conflicts to the first user.

[0146] The authoring application also presents a toggle interface 2215 to the first user in FIG. 26 to enable the user to choose between continuing to edit the data file without viewing the editing conflicts and reviewing the editing conflicts (see second determination module 514 of FIG. 5). The first user may choose to continue editing the data file without reviewing the editing conflicts (see continue operation 516 of FIG. 5). The first user also may choose to select the toggle interface 2215 (FIG. 26) to reveal the editing conflicts (see review operation 518 of FIG. 5). In one embodiment, selection of the toggle interface 2215 also activates a summary window 2230 (see FIG. 27). The first user may continue to edit the merged version of the data file regardless of whether or not the first user chooses to view the editing conflicts.

[0147] One example process by which the authoring application may implement the review operation 518 of FIG. 5 includes the review process 700 of FIG. 7. When the user selects the toggle interface 2215, the authoring application presents the editing conflicts to the user (see display operation 704 of FIG. 7). For example, in one embodiment, selecting the toggle interface 2215 splits the display window 2210 between the editing area 2212 and a summary window 2230 in which the editing conflicts are listed (see list 2234) and summary information about the conflicts may be displayed (see summary area 2232).

[0148] In general, the editing area 2212 displays the merged version of the data file and enables the user to freely edit the merged version. In some embodiments, the merged version of the data file displayed in the editing area 2212 is annotated to indicate which portions of content are in conflict. For example, in one embodiment, the merged version of the data file is annotated to indicate whether the conflicting content resulted from insertions (e.g., via underlining or other formatting, font color, highlighting, opacity, or other such indicia) or deletions (e.g., via strikethroughs or other formatting, font color, highlighting, opacity, or other such indicia).

[0149] In the example shown, the first data unit does not contain any editing conflicts since the first data unit is locked on the server to the second user. Accordingly, the first data unit is not annotated. The second data unit also does not contain any editing conflicts so the second data unit is not annotated. Some content within the third data unit, which is shown as locked to the first user, is underlined to indicate the content has been inserted into the data unit.

[0150] In one embodiment, only conflicting content originating from the user copy is annotated. Advantageously, only annotating user created conflicts allows the user to understand the state of the master copy of the data file and differences between the master copy and the user copy. In other embodiments, however, conflicting content originating from the master copy may be annotated as well as or in place of the conflicting content from the user copy.

[0151] The conflicting editing operations also are displayed by the summary window 2230. The summary area 2232 of the resolution interface indicates the number of conflicting editing operations contained within the merged version of the data file. In the example shown, the summary area 2232 indicates one conflicting editing operation has been identified. The listing 2234 of the summary window 2230 displays the conflicting content. In one embodiment, the listing 2234 displays the conflicting content separate from the non-conflicting content. In another embodiment, the listing 2234 annotates the conflicting content to indicate whether the content as inserted and/or deleted (e.g., see indicia 2236 of FIG. 27). In another embodiment, the listing 2234 annotates the conflicting content to indicate the origin (e.g., user copy or master copy) of the conflicting content (e.g., see indicia 2238 of FIG. 27).

[0152] The first user of the authoring application may interact with a resolution interface (e.g., see resolution interface 2240 of FIG. 27) to provide instructions (see obtain operation 706 of FIG. 7) either to return to editing the data file or to resolve any of the editing conflicts displayed (see first determination module 708 of FIG. 7). For example, in one embodiment, the user may select the toggle interface 2215 again to hide the editing conflicts as discussed above.

[0153] Alternatively, the first user may provide instructions to accept or reject conflicting content or portions thereof. In one embodiment, the first user may select content displayed within the editing area 2012 of the display window 2010 and may select an option on a resolution interface 2240 to provide instructions to resolve the editing operation. For example, the first user may select the content on which instructions are to be provided and may select an accept button 2242 or a reject button 2244 from an example resolution interface 2040 (see FIG. 27). The example resolution interface 2040 also may include a next button 2246 and a previous button 2248 for sequencing among conflicts. In other embodiments, the first user may provide the resolution instruction using another type of resolution interface tool.

[0154] In the example shown in FIG. 28, the first user selects a first portion of the conflicting content (see selection indicia 2213 in the summary window 2230) and selects an accept button 2242 on the resolution interface 2040 via a cursor 2219. In other embodiments, however, the first user may have provided the instruction via any desired resolution interface. Accordingly, the authoring application determines the user has not provided instructions to return to editing (see first determination module 708 of FIG. 7) and performs the resolution instruction (see implement operation 710 of FIG. 7).

[0155] One example process by which the authoring application may accept the selected conflicting content includes the accept process 1100 of FIG. 13. The authoring application removes any annotations from the selected conflicting content (see clean operation 1104 of FIG. 13). The authoring application determines the selected conflicting content was inserted into the data file (see determination module 1106 of FIG. 13). Accordingly, the accept process 1100 completes and ends (see stop module 1110 of FIG. 13). If the authoring application had determined the conflicting content was deleted from the data file, then the authoring application would have deleted the conflicting content from the data file (see remove operation 1108 of FIG. 13).

[0156] Returning to the review process 700 of FIG. 7, the authoring application determines no more editing conflicts remain unresolved (see second determination module 712 of FIG. 7). Accordingly, the review process 700 completes and ends (see stop module 714 of FIG. 7), thereby completing the review operation 514 of the synchronize process 500 of FIG. 5. The synchronize process 500 proceeds back to the first determination module 510 (FIG. 5) at which the authoring application determines no editing conflicts remain unresolved. The authoring application also determines no additional updates are available from the master copy (see the second determination module 520 of FIG. 5) and, accordingly, sends an update from the user copy to the master copy (see store operation 522 of FIG. 5) to complete synchronization of the user and master copies.

[0157] The authoring application alerts the first user when the authoring application successfully uploads the update to the master copy (see indicate operation 524 of FIG. 5). For example, in FIG. 29, the status bar 2218 has been updated to indicate the update was successfully transmitted. In some embodiments, the authoring application also may display an alert window 2260 indicating the success of the upload. In one embodiment, the alert window 2260 may provide further explanation, e.g., indicating consequences of the successful upload. In the example shown in FIG. 29, the summary window 2230 is removed from the user interface 2200 since no identified editing conflicts remain unresolved. In other embodiments, however, the summary window 2230 may remain until deactivated by the first user (e.g., via toggle interface 2215 of FIGS. 27 and 28).

[0158] Embodiments of the disclosure may be implemented as a computer process (method), a computing system, or as an article of manufacture, such as a computer program product or computer readable media. The processes (programs) can be implemented in any number of ways, including the structures described in this document. One such way is by machine operations, of devices of the type described in this document. Another optional way is for one or more of the individual operations of the methods to be performed on a computing device in conjunction with one or more human operators performing some of the operations. These human operators need not be collocated with each other, but each can be only with a machine that performs a portion of the program.

[0159] The computer program product may be a computer storage media readable by a computer system and encoding a computer program of instructions for executing a computer process. The computer program product may also be a propagated signal on a carrier readable by a computing system and encoding a computer program of instructions for executing a computer process. The term computer readable media as used herein includes both storage media and communication media.

[0160] Those skilled in the art will appreciate that the disclosure may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The disclosure may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices. Generally, program modules include routines, programs, components, data structures, and other types of structures that perform particular tasks or implement particular abstract data types.

WE CLAIM:

1. A method of collaboratively authoring a data file comprising:

identifying at a first authoring application (130) implemented on a first computing device (110) an editing conflict existing between a user copy (155) of the data file and a master copy (150) of the data file;

merging the user copy of the data file and the master copy of the data file to generate a merged copy of the data file;

displaying the merged copy of the data file to the user of the first authoring application;

presenting to a user a toggle interface (815, 915, 2015, 2215) enabling the user to toggle between displaying the identified editing conflict and hiding the identified editing conflict, the toggle interface being presented when the editing conflict is identified and remaining until the identified editing conflict has been resolved, wherein the merged copy of the data file is freely editable by the user both when the identified editing conflict is displayed and when the identified editing conflict is hidden; and

providing a resolution interface (930, 2040, 2050, 2240) enabling the user to provide resolution instructions for the identified editing conflict when the user toggles to displaying the identified editing conflict.

2. The method of claim 1, wherein enabling the user to provide resolution instructions comprises enabling the user to select whether to accept or reject the editing conflict.

3. The method of claim 1, further comprising:

identifying at the first authoring application a plurality of editing conflicts existing between the user copy of the data file and the master copy of the data file, wherein presenting the toggle interface to the user enables the user to toggle between displaying the editing conflicts and hiding the editing conflicts.

4. The method of claim 3, further comprising:

displaying the identified editing conflicts when the user toggles to displaying the identified editing conflicts; wherein the presented resolution interface enables the user to accept or reject any of the editing conflicts.

5. The method of claim 1, wherein displaying the identified editing conflict comprises displaying the identified editing conflict within context of non-conflicting content.

6. The method of claim 5, wherein displaying the identified editing conflict comprises annotating the identified editing conflict to distinguish the identified editing conflict from the non-conflicting content.

7. The method of claim 6, wherein annotating the identified editing conflict comprises displaying an origin of the identified editing conflict.

8. The method of claim 6, wherein hiding the identified editing conflict comprises removing any annotation from the identified editing conflict.

9. The method of claim 1, further comprising receiving at the first authoring application an update indicating a current state of the master copy of the data file to synchronize the user copy of the data file with the master copy.

10. The method of claim 1, further comprising alerting the user when the editing conflict is identified.

11. The method of claim 1, further comprising:

receiving instructions to synchronize the user copy of the data file with the master copy;

generating a new copy of the data file based on the user copy, the new copy including the identified editing conflict, wherein editing the new copy of the data file does not trigger presentation of the toggle interface.

12. A computer readable storage medium storing executable instructions, which perform a method of resolving editing conflicts when executed by a computing device, the editing conflicts existing between a user copy (155) of a data file and a master copy (150) of the data file, the method comprising:

presenting a display window (810, 910, 2010, 2210) to a user of the computing device, the display window including an editing area (812, 912, 2012, 2212), a summary area (820, 920, 2030, 2230), and a resolution interface (930, 2040, 2050, 2240), wherein the resolution interface is configured to enable the user to provide resolution instructions for identified conflicting content selected by the user;

displaying any content locks (154, 2022, 2024) and content (152) of the user copy including any identified conflicting content within the editing area of the display window, wherein the user may edit the content of the user copy including the identified conflicting content freely in the editing area, the identified conflicting content being annotated to distinguish the identified conflicting content from non-conflicting content;

displaying the identified conflicting content and any identified conflicting content locks within the summary window; and

presenting a toggle interface (815, 915, 2015, 2215) to the user, wherein selection of the toggle interface by the user removes any annotations to the identified conflicting content within the editing area of the display window.

13. The computer readable storage medium of claim 12, wherein election of the toggle interface by the user also removes the resolution interface from the display window.

14. The computer readable storage medium of claim 12, further comprising displaying an alert reporting to the user that the identified conflicting content exists.

15. The computer readable storage medium of claim 12, wherein the identified conflicting content includes only content that is inserted or deleted by the user to the user copy of the data file and that interferes with intervening changes to the master copy of the data file.

16. A system for collaboratively editing a data file comprising:

a storage device (120) on which a master copy (150) of the data file is stored, the master copy having master content (152) and master locks (154);

a user device (110) on which a user copy (155) of the data file is stored, the user copy having user content (152') and user locks (154'), the user content being generated based on the master content and the user locks being generated based on the master locks;

an authoring application (130) being implemented on the user device, the authoring application being configured to receive from the storage device master content updates indicating any changes to the master content and master lock updates indicating any changes to the master locks,

the authoring application also being configured to identify any editing conflicts between the user copy of the data file and the master content updates and between the user copy and the master lock updates, and

wherein the authoring application automatically instantiates any changes to the master locks when the changes to the master locks do not conflict with any changes to the user locks; and

wherein the authoring application automatically instantiates any changes to the master content only when no editing conflicts are identified by the authoring application.

17. The system of claim 16, wherein the authoring application also is configured to send to the storage device user content updates indicating any changes to the user content only when no editing conflicts are identified by the authoring application.

18. The system of claim 16, wherein the authoring application also is configured to send to the storage device user lock updates indicating any changes to the user locks whether or not editing conflicts are identified by the authoring application.

19. The system of claim 16, further comprising:

a resolution interface configured to be displayed by the authoring application, the resolution interface being configured to present the user copy of the data file including displaying the user locks and displaying the user content annotated to indicate any identified editing conflicts.

20. The system of claim 16, further comprising a plurality of user devices communicatively coupled to the storage device, wherein each user device is configured to obtain a user copy of the data file, to receive updates from the storage device at periodic intervals, and to send updates to the storage device when the identified editing conflicts are resolved.

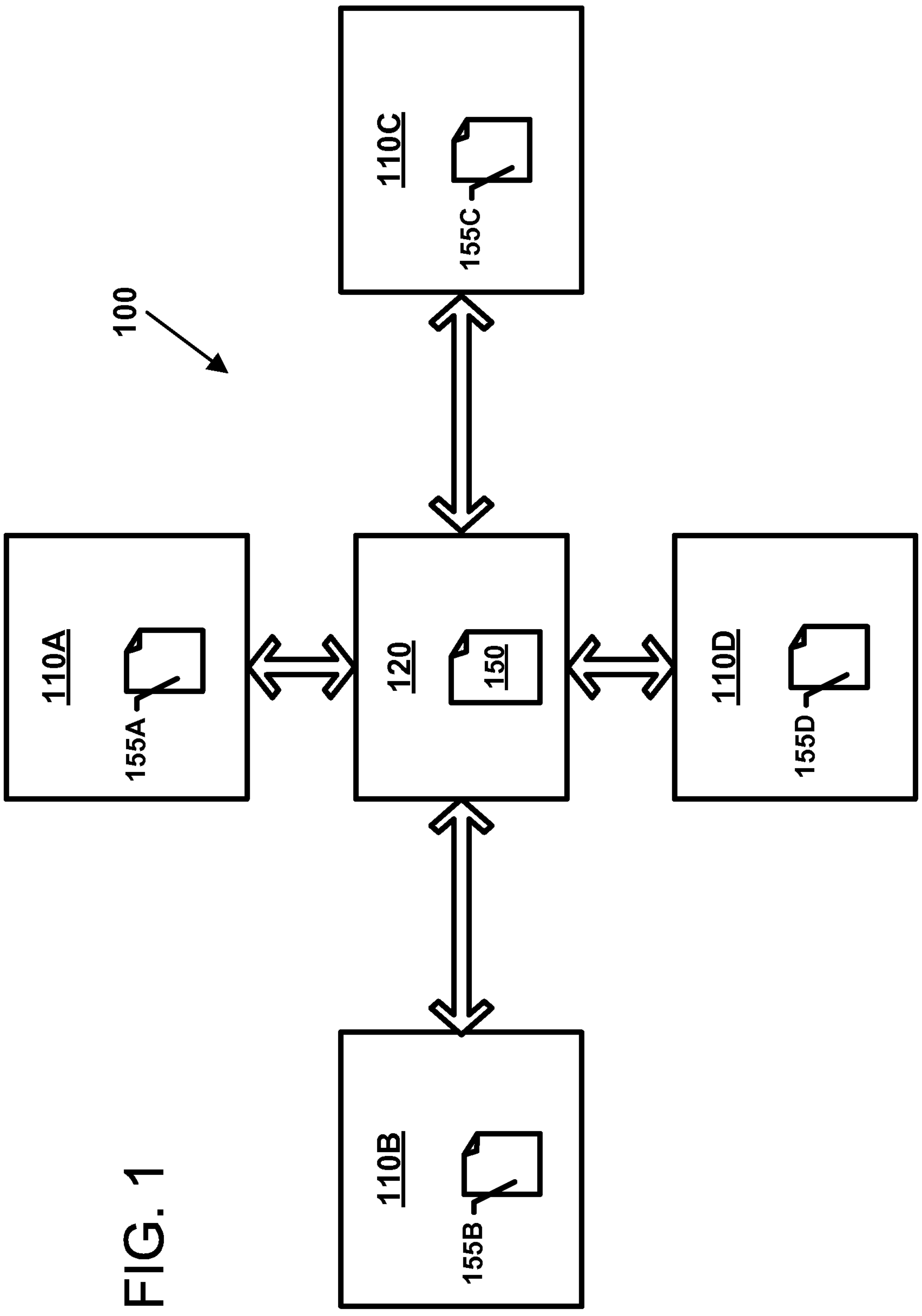


FIG. 1

FIG. 2

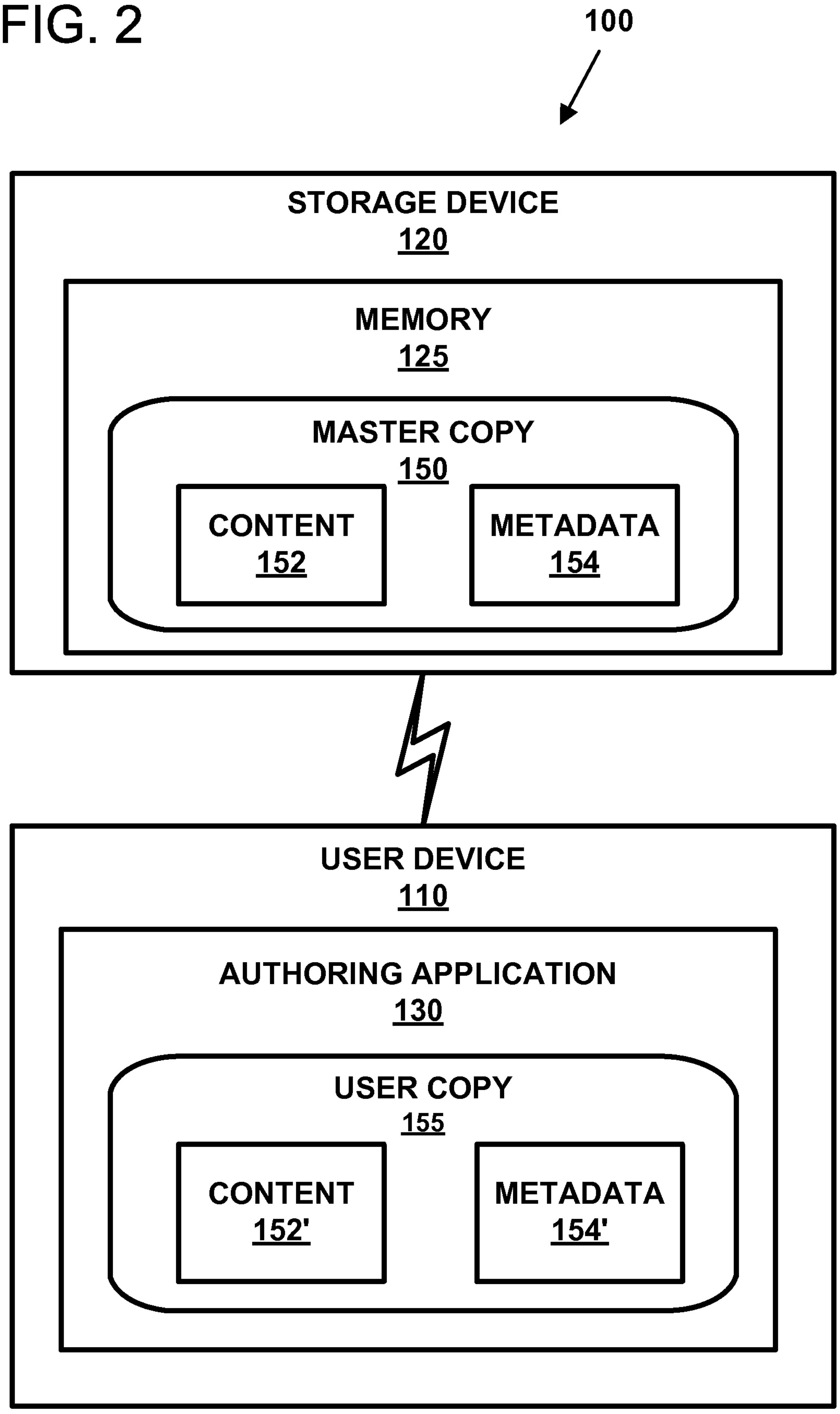
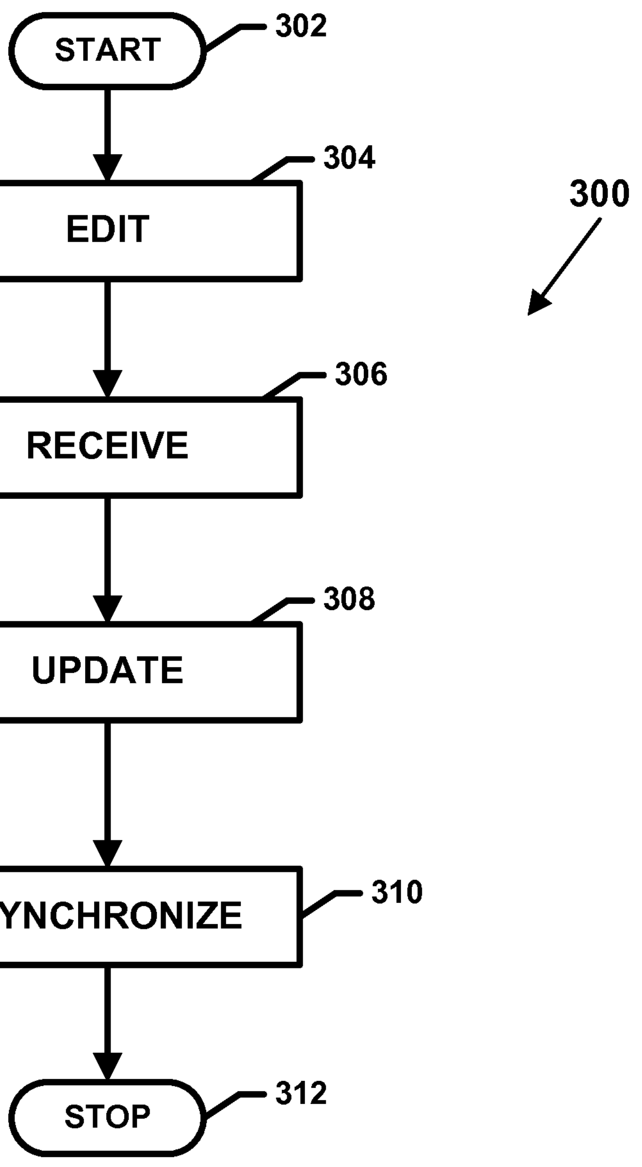

100
STORAGE DEVICE
120
MEMORY
125
MASTER COPY
150
CONTENT
152
METADATA
154
USER DEVICE
110
AUTHORING APPLICATION
130
USER COPY
155
CONTENT
152'
METADATA
154'

FIG. 3


START
302
EDIT
304
300
RECEIVE
306
UPDATE
308
SYNCHRONIZE
310
STOP
312

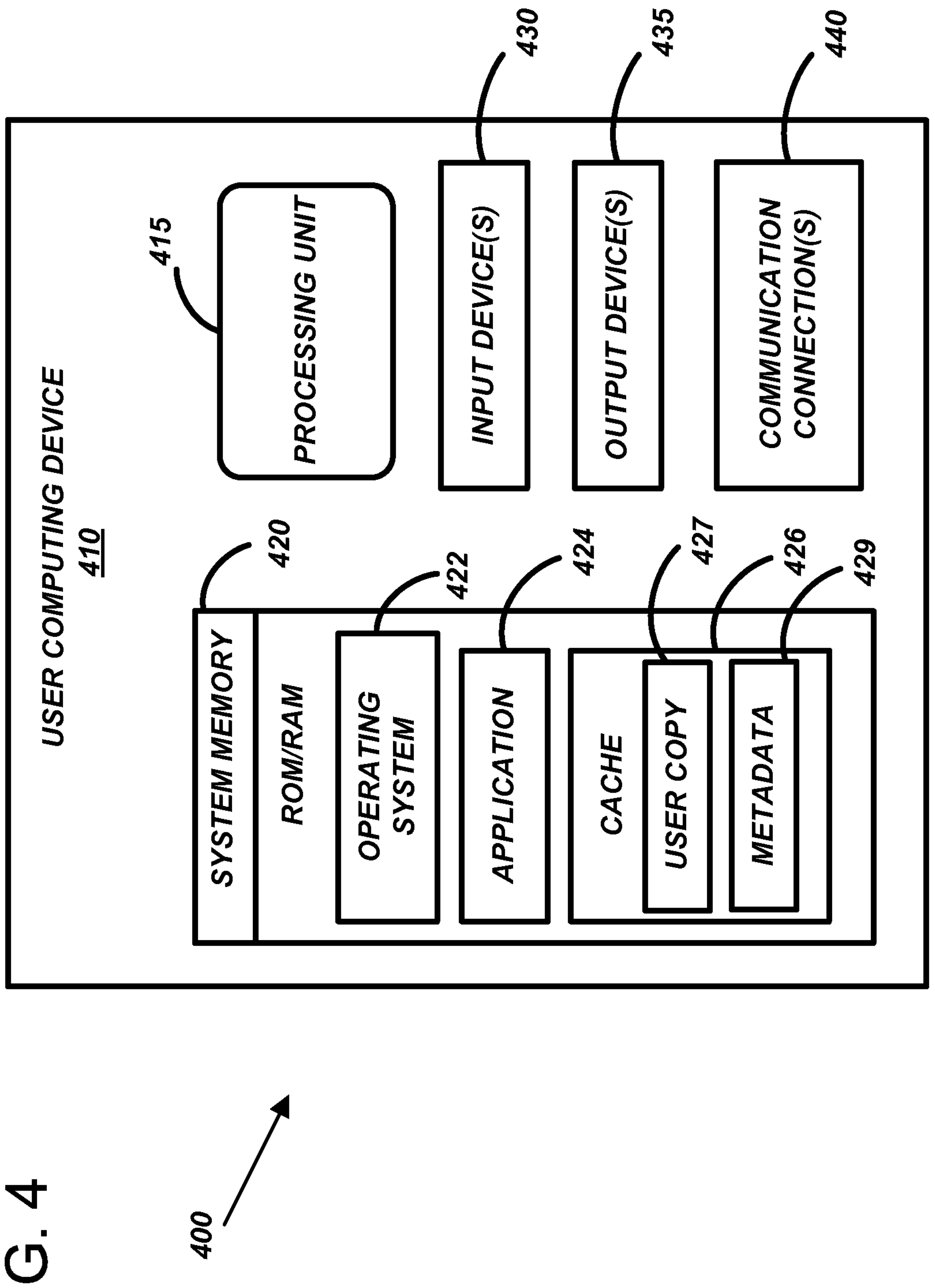

USER COMPUTING DEVICE
410
415
PROCESSING UNIT
430
INPUT DEVICE(S)
435
OUTPUT DEVICE(S)
440
COMMUNICATION CONNECTION(S)
420
SYSTEM MEMORY
ROM/RAM
422
OPERATING SYSTEM
424
APPLICATION
427
426
429
CACHE
USER COPY
METADATA
400

FIG. 4

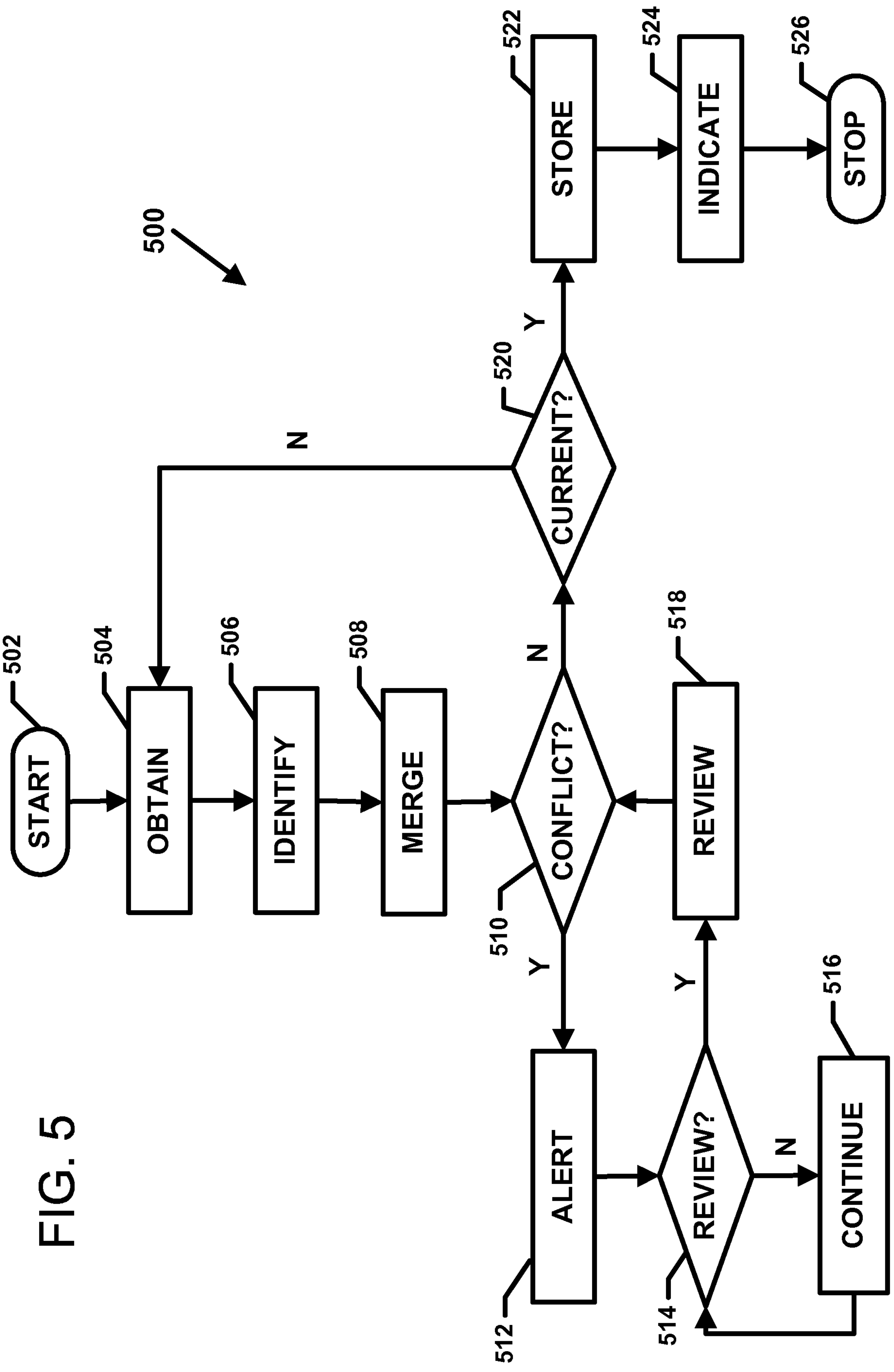

500
START
502
OBTAIN
504
IDENTIFY
506
MERGE
508
CONFLICT?
510
Y
N
ALERT
512
REVIEW?
514
Y
N
CONTINUE
516
REVIEW
518
CURRENT?
520
N
Y
STORE
522
INDICATE
524
STOP
526

FIG. 5

FIG. 6

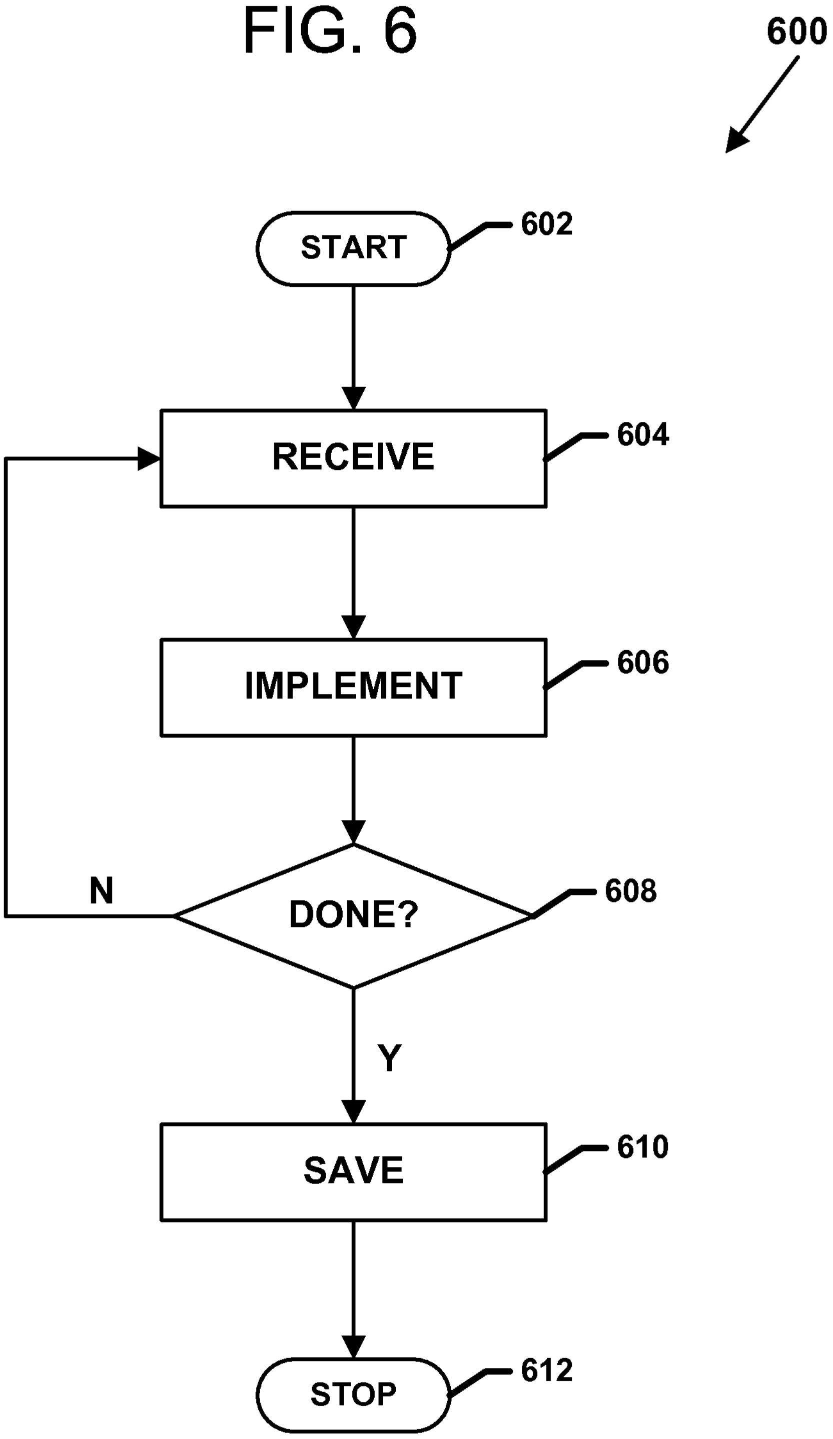
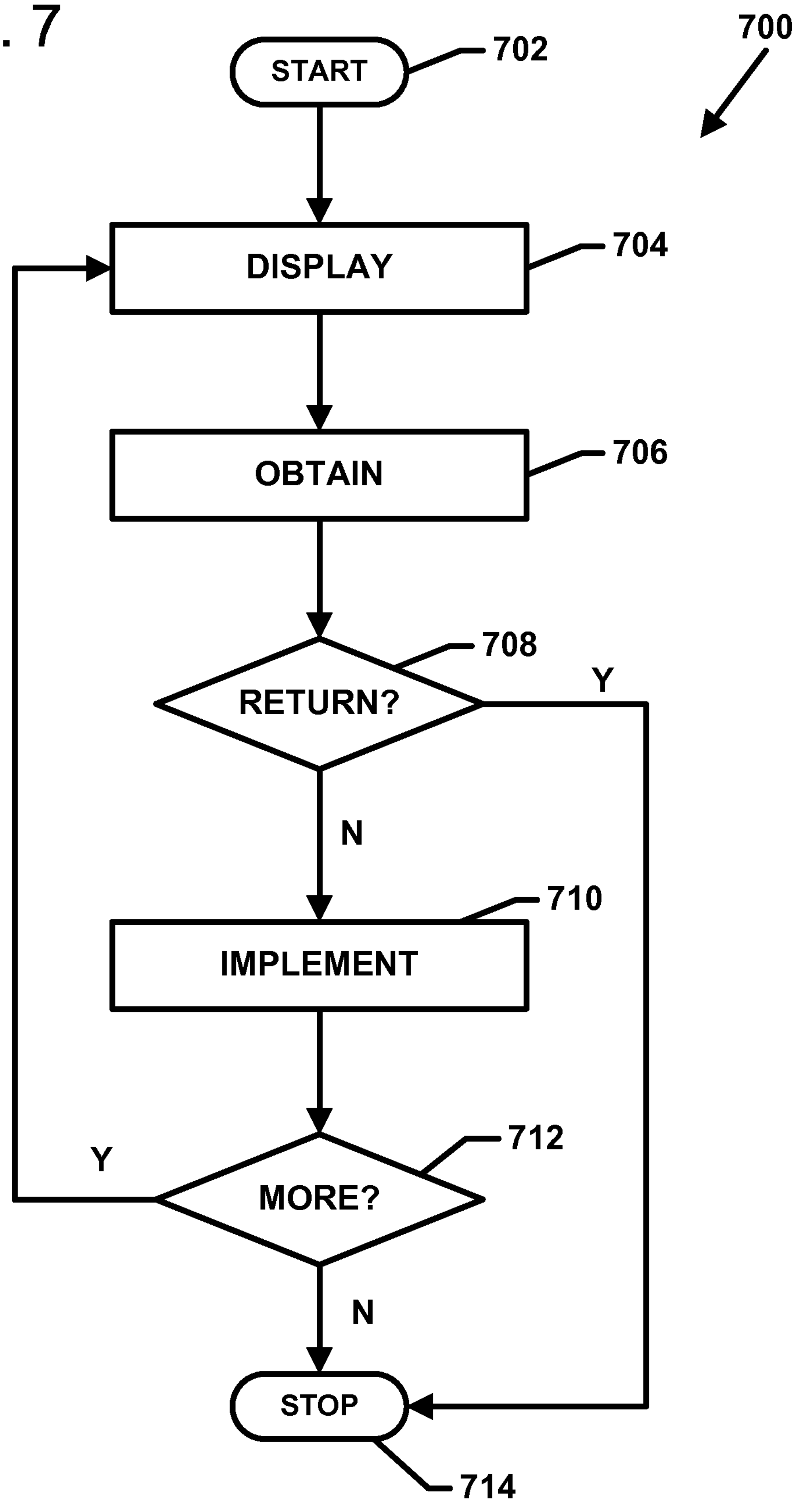

600
START
602
RECEIVE
604
IMPLEMENT
606
DONE?
608
N
Y
SAVE
610
STOP
612

FIG. 7


700
START
702
DISPLAY
704
OBTAIN
706
RETURN?
708
Y
N
IMPLEMENT
710
MORE?
712
Y
N
STOP
714

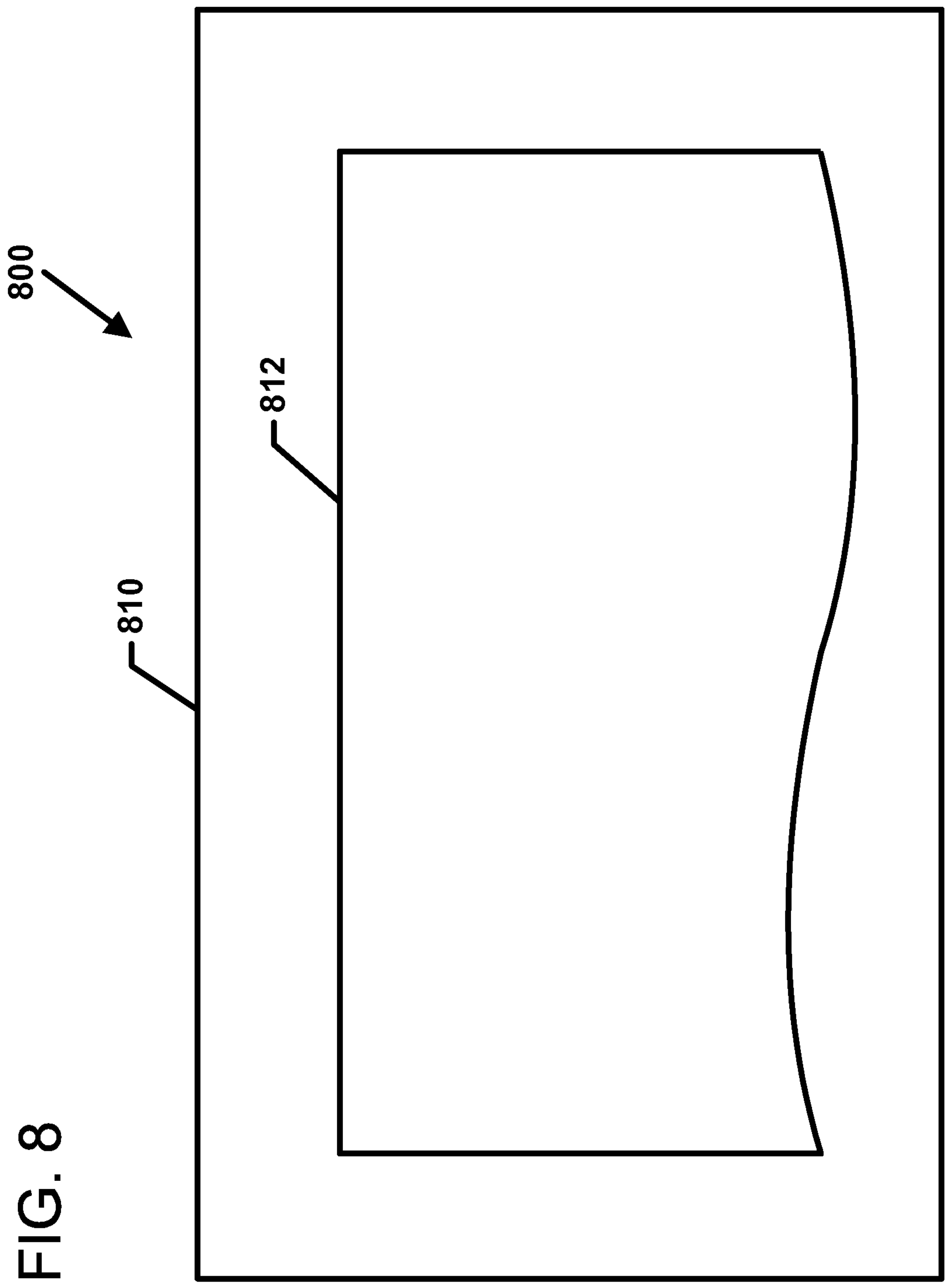


FIG. 8

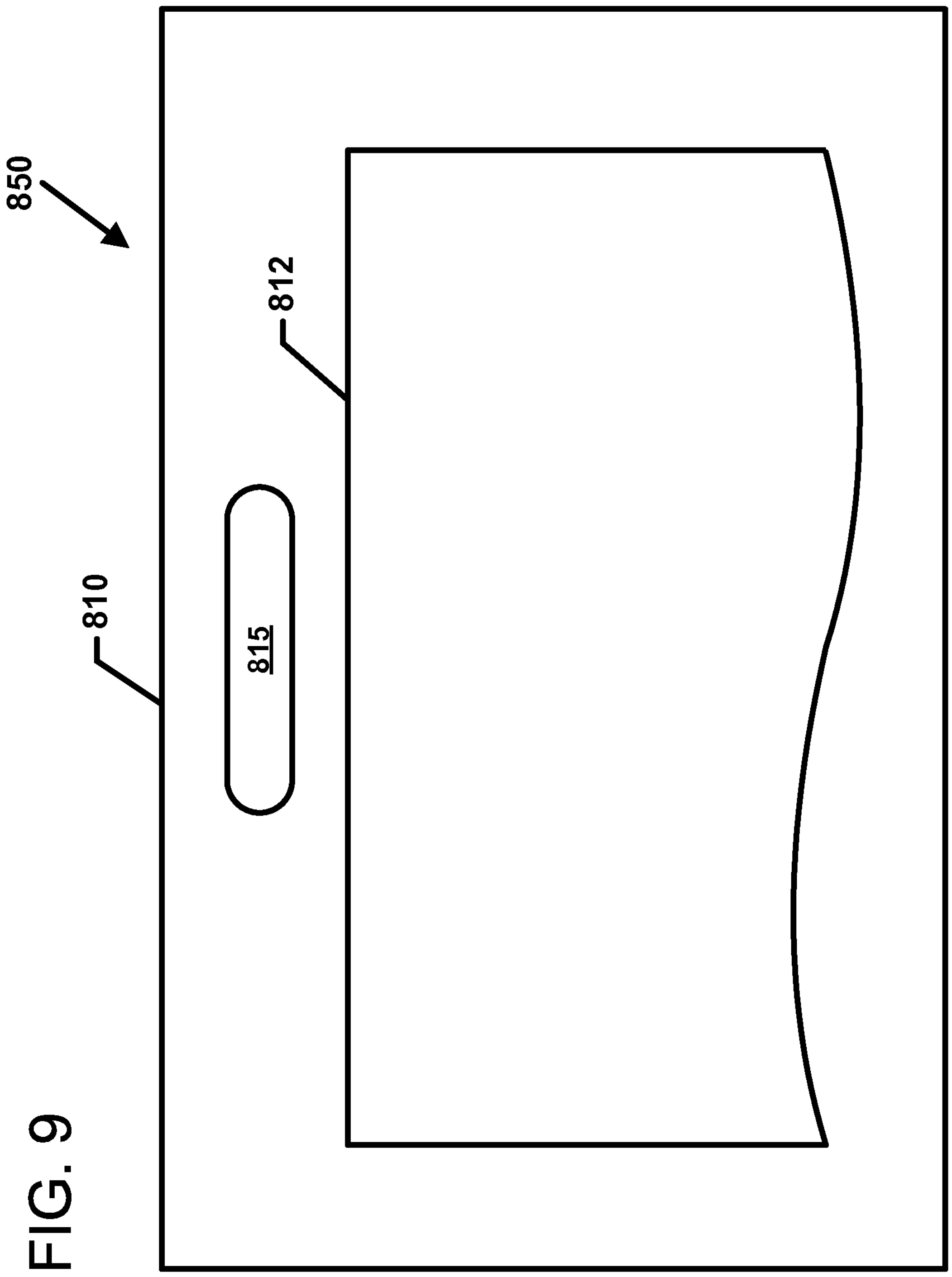


FIG. 9

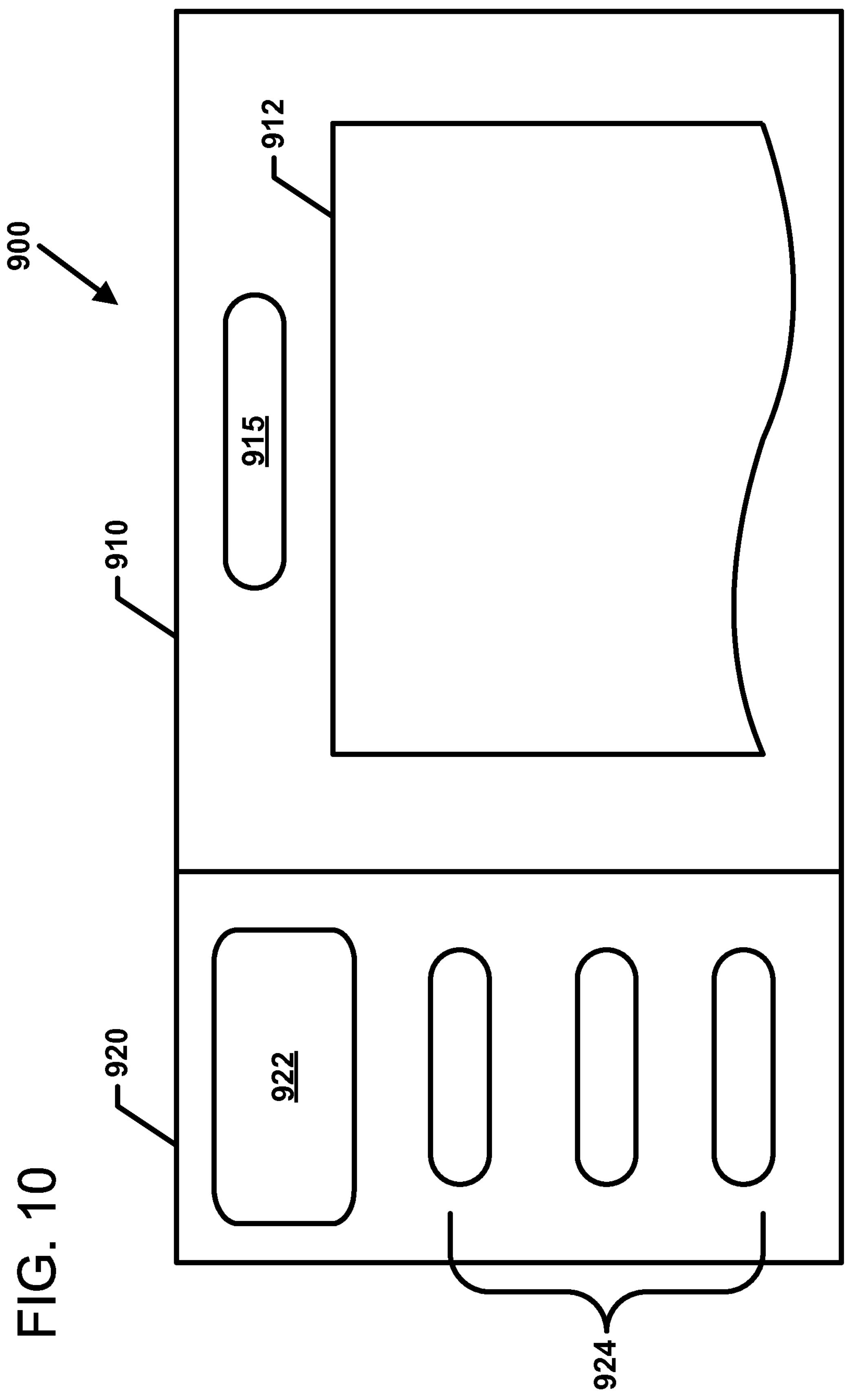


FIG. 10

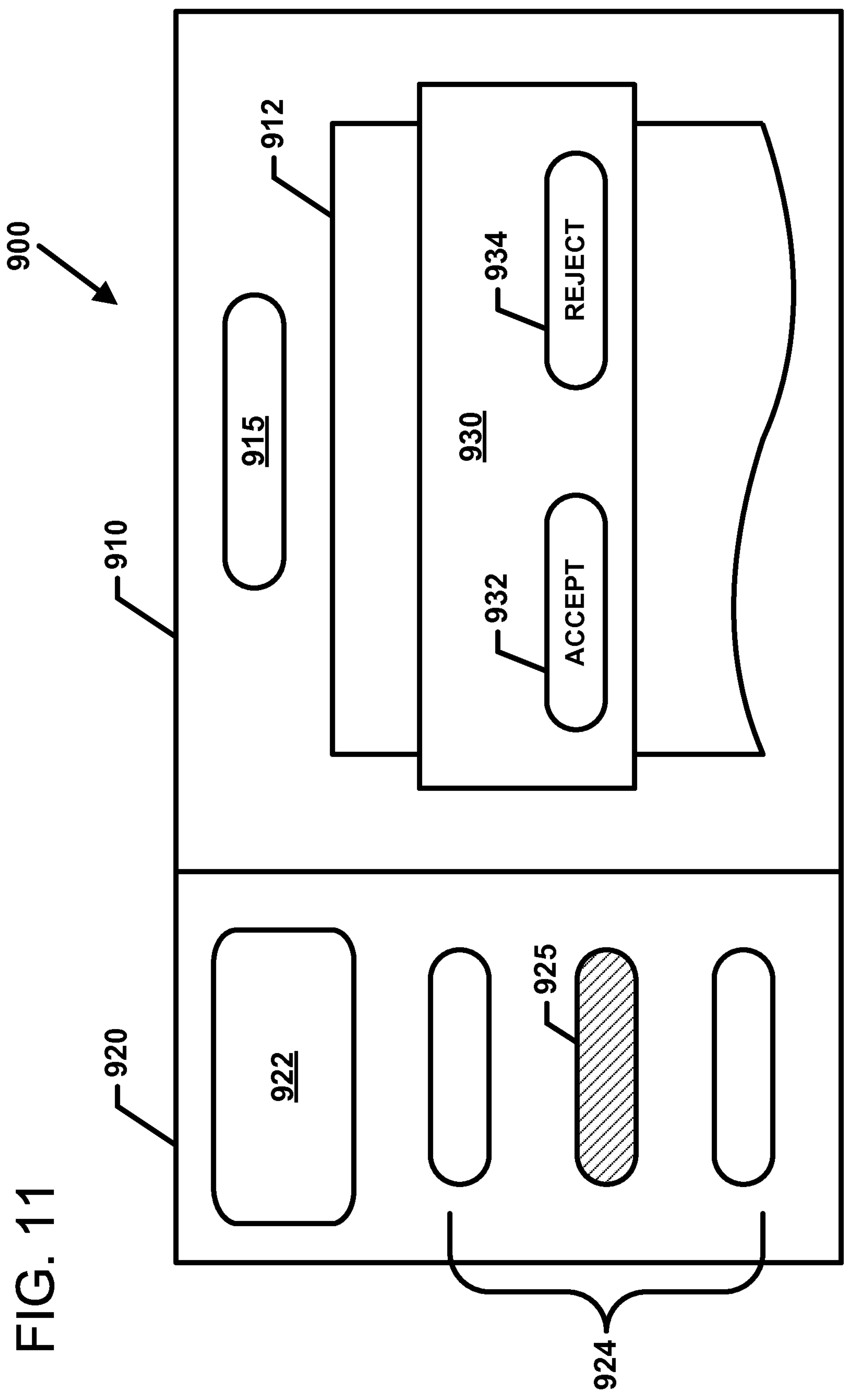


FIG. 11

FIG. 12

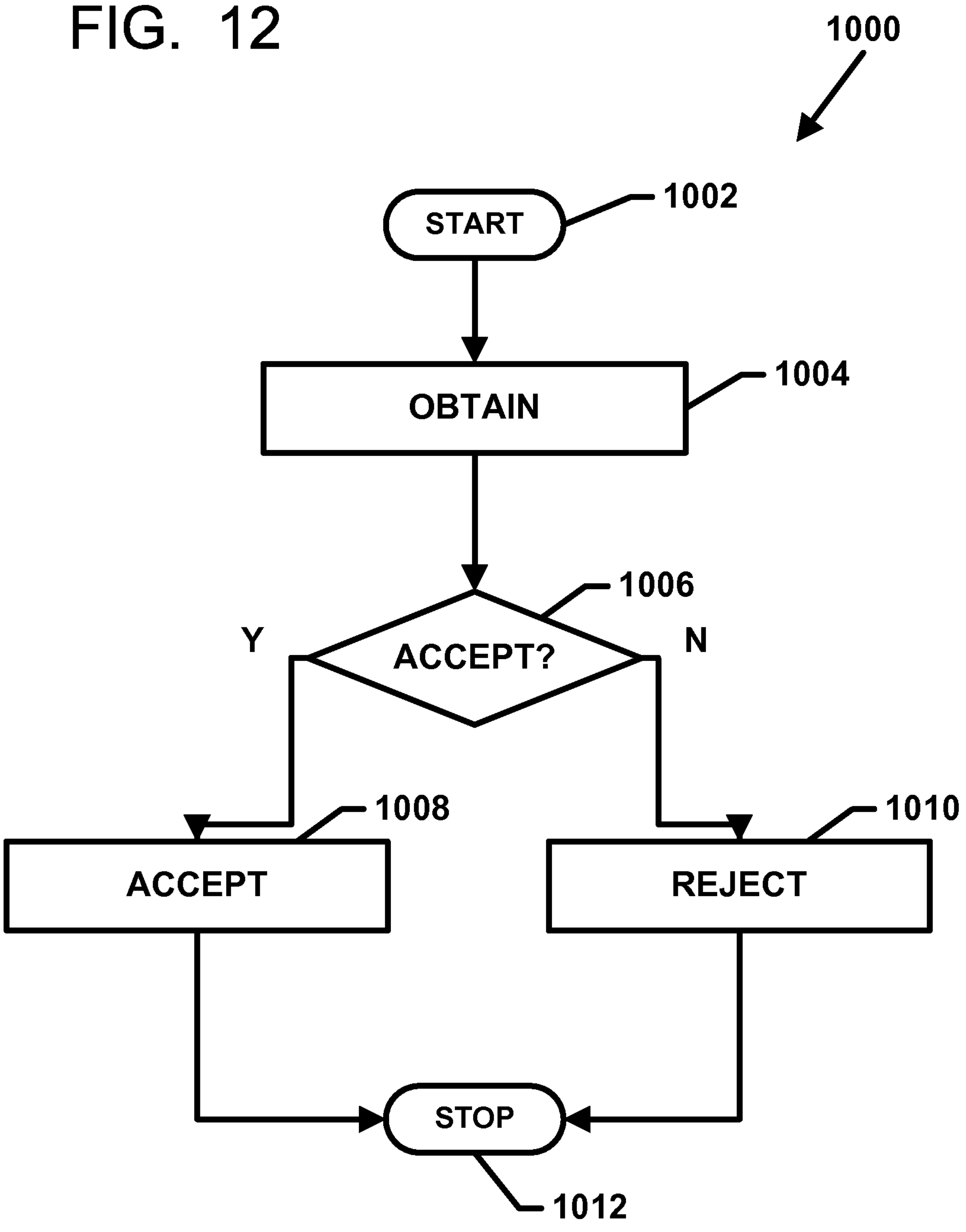

1000
START
1002
OBTAIN
1004
ACCEPT?
1006
Y
N
ACCEPT
1008
REJECT
1010
STOP
1012

FIG. 13

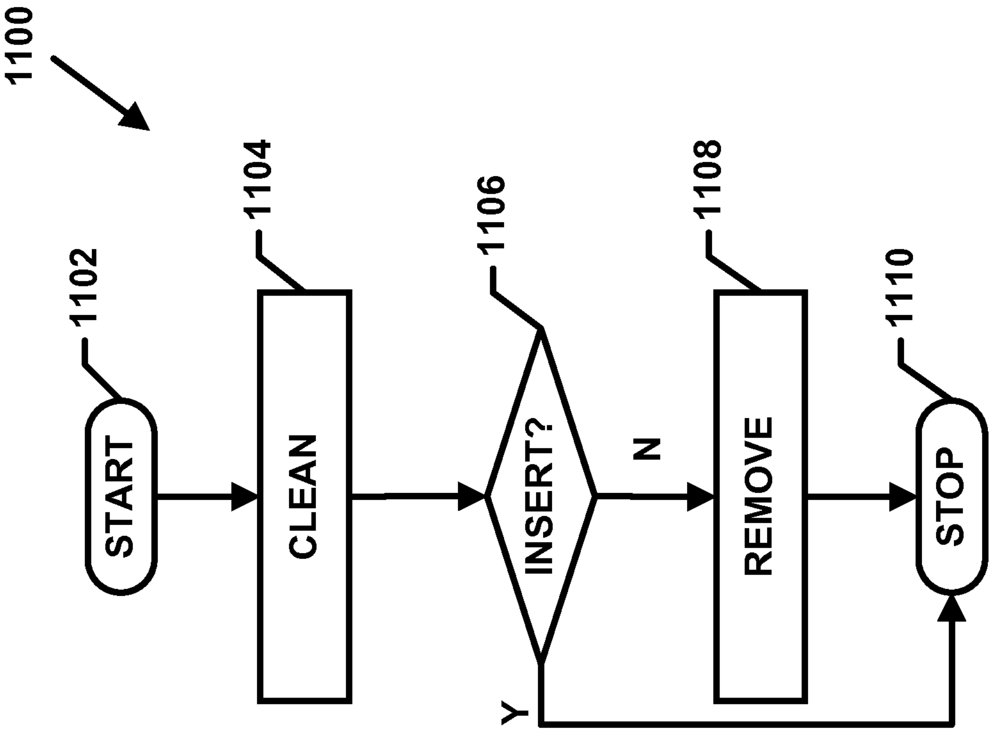

1100
1102
START
1104
CLEAN
1106
INSERT?
N
Y
1108
REMOVE
1110
STOP

FIG. 14

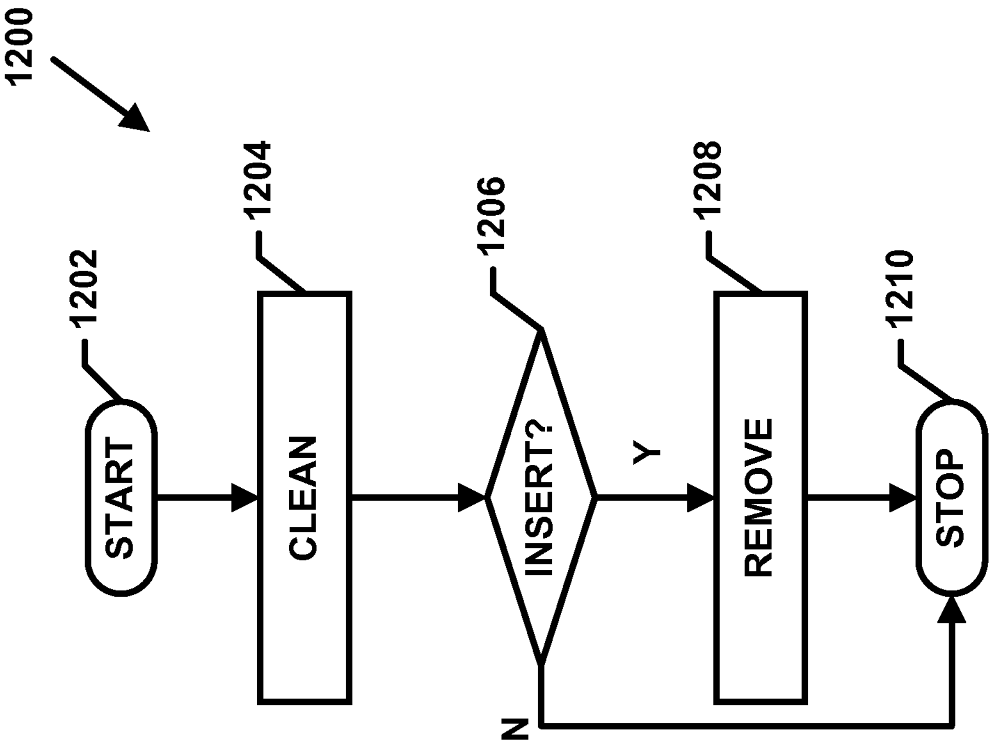

1200
1202
START
1204
CLEAN
1206
INSERT?
Y
N
1208
REMOVE
1210
STOP

FIG. 15

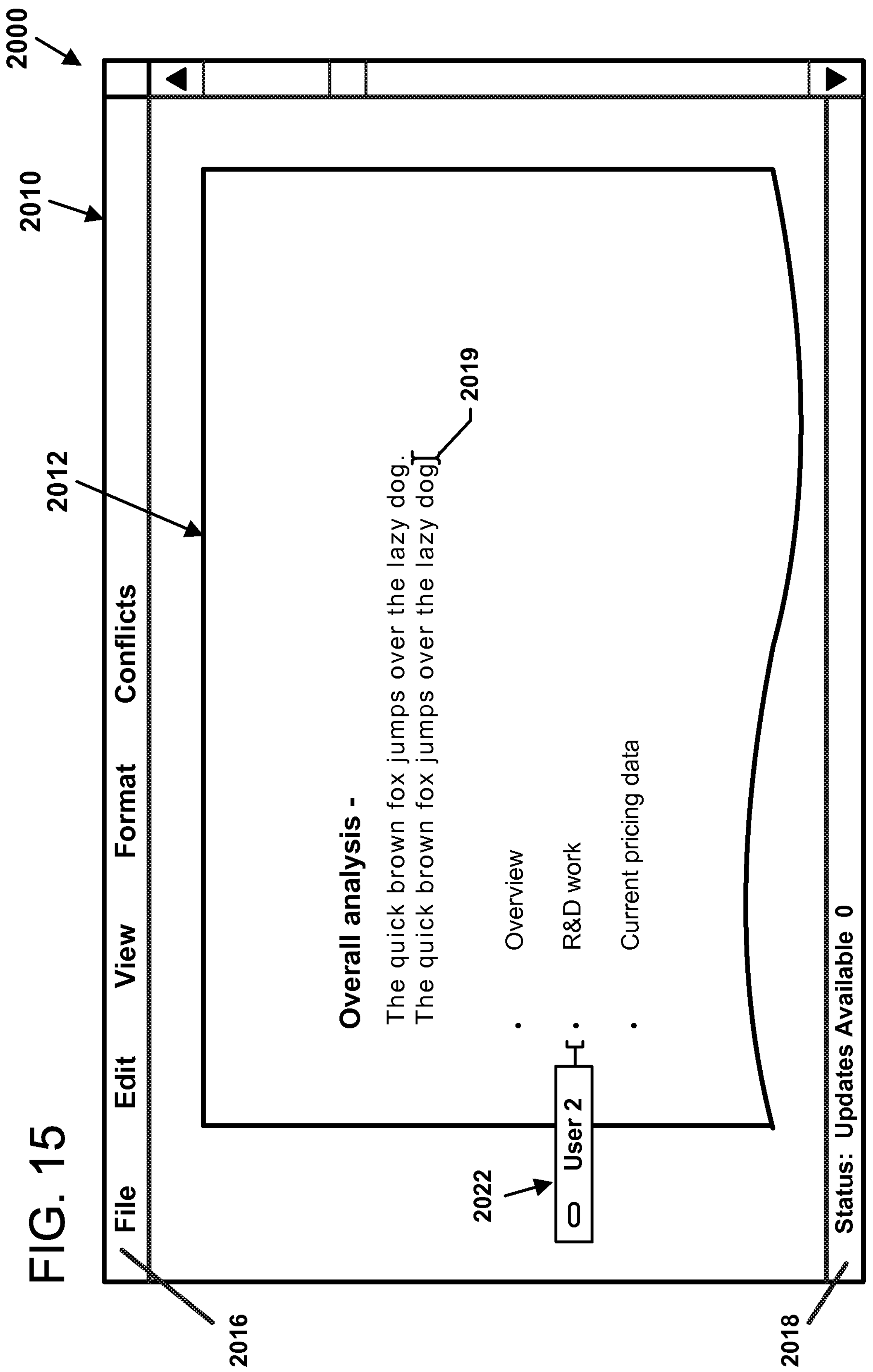
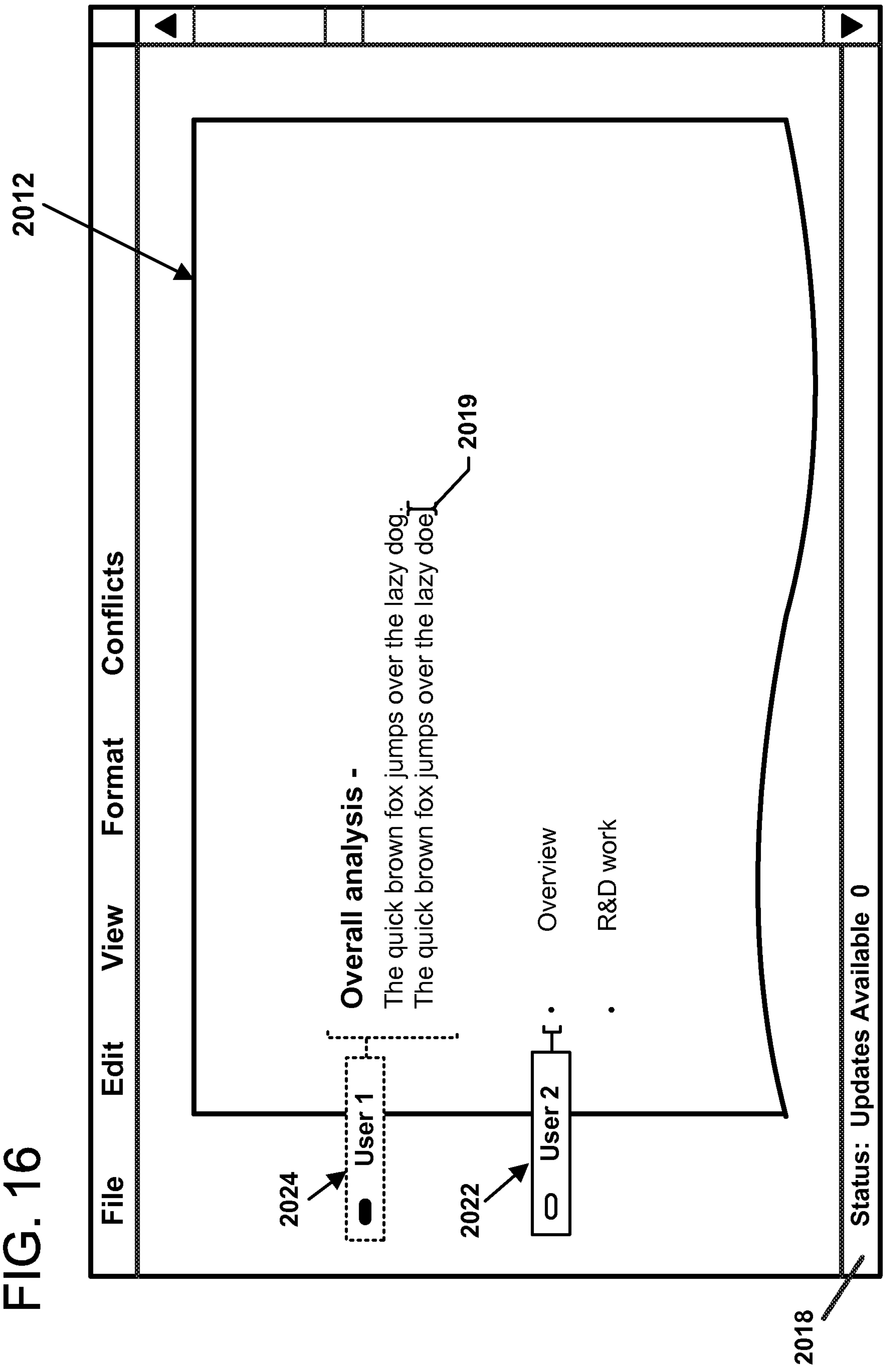

2000
2010
2012
2019
2016
2018
2022
File
Edit
View
Format
Conflicts
Overall analysis -
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog
Overview
R&D work
Current pricing data
User 2
0
Status: Updates Available 0

FIG. 16



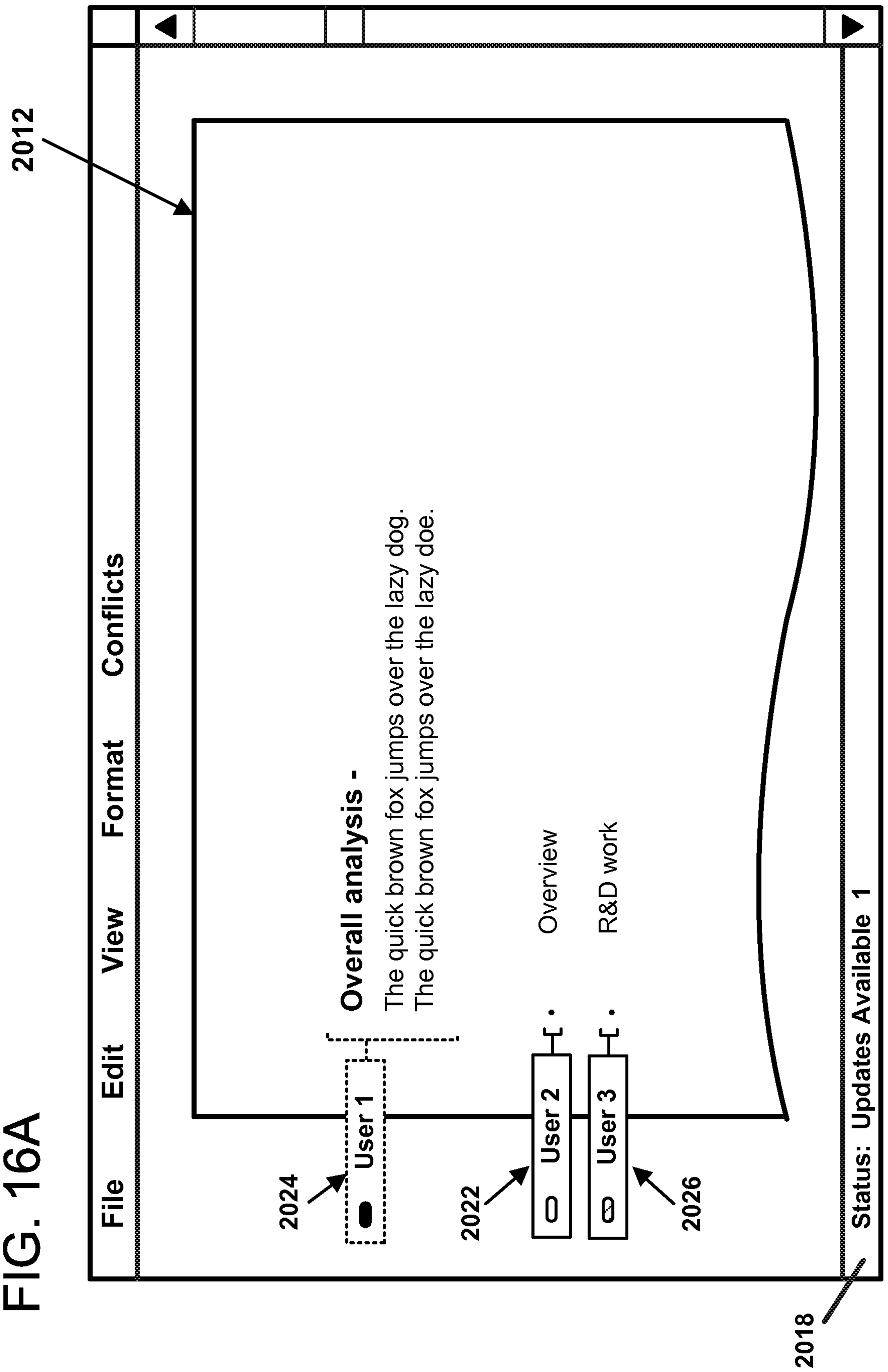
Overall analysis -

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy doe

- Overview
- R&D work

Status: Updates Available 0

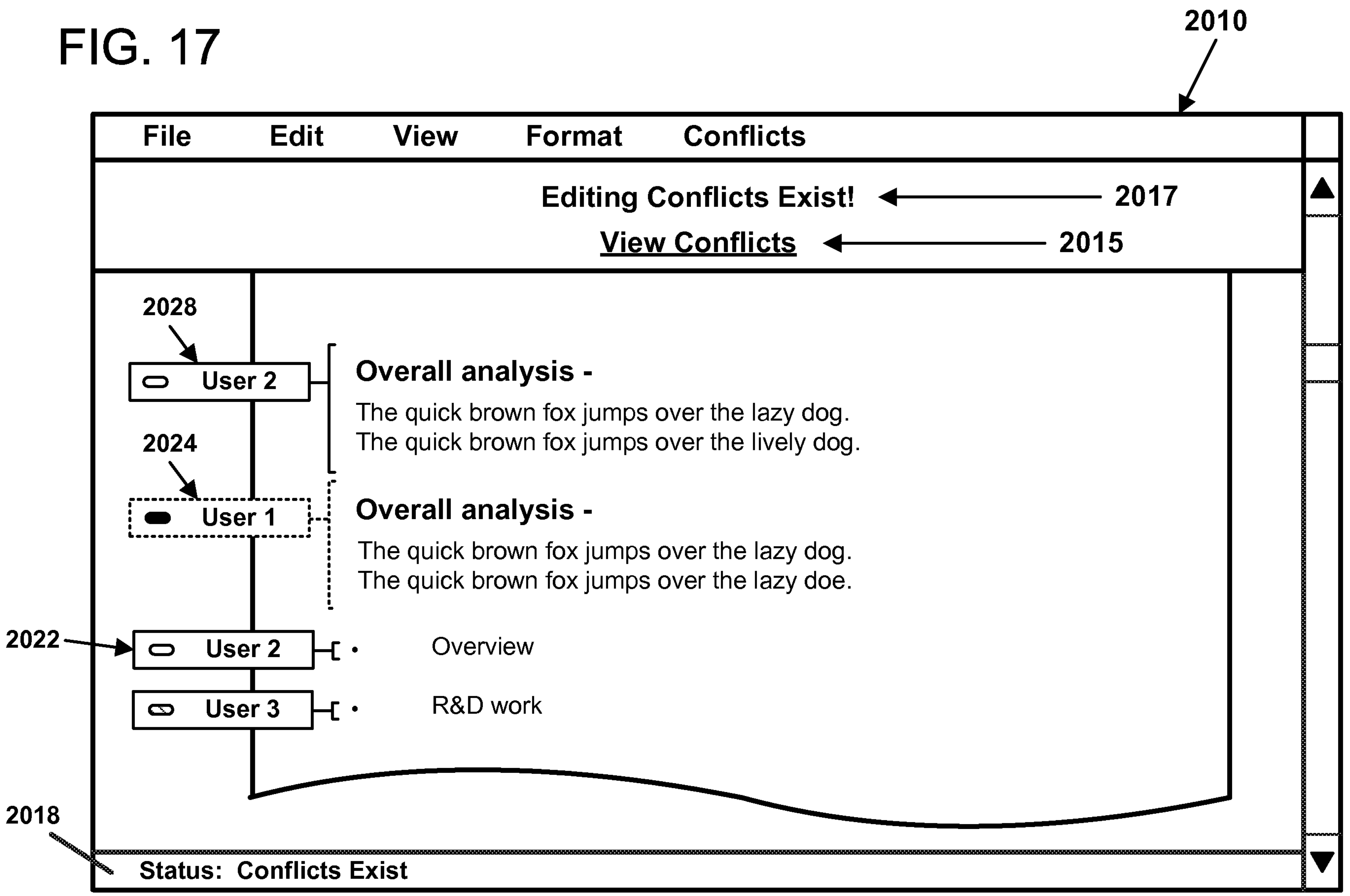
FIG. 16A



Overall analysis -
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy doe.

- Overview
- R&D work

FIG. 17



Editing Conflicts Exist!

View Conflicts

Overall analysis -

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lively dog.

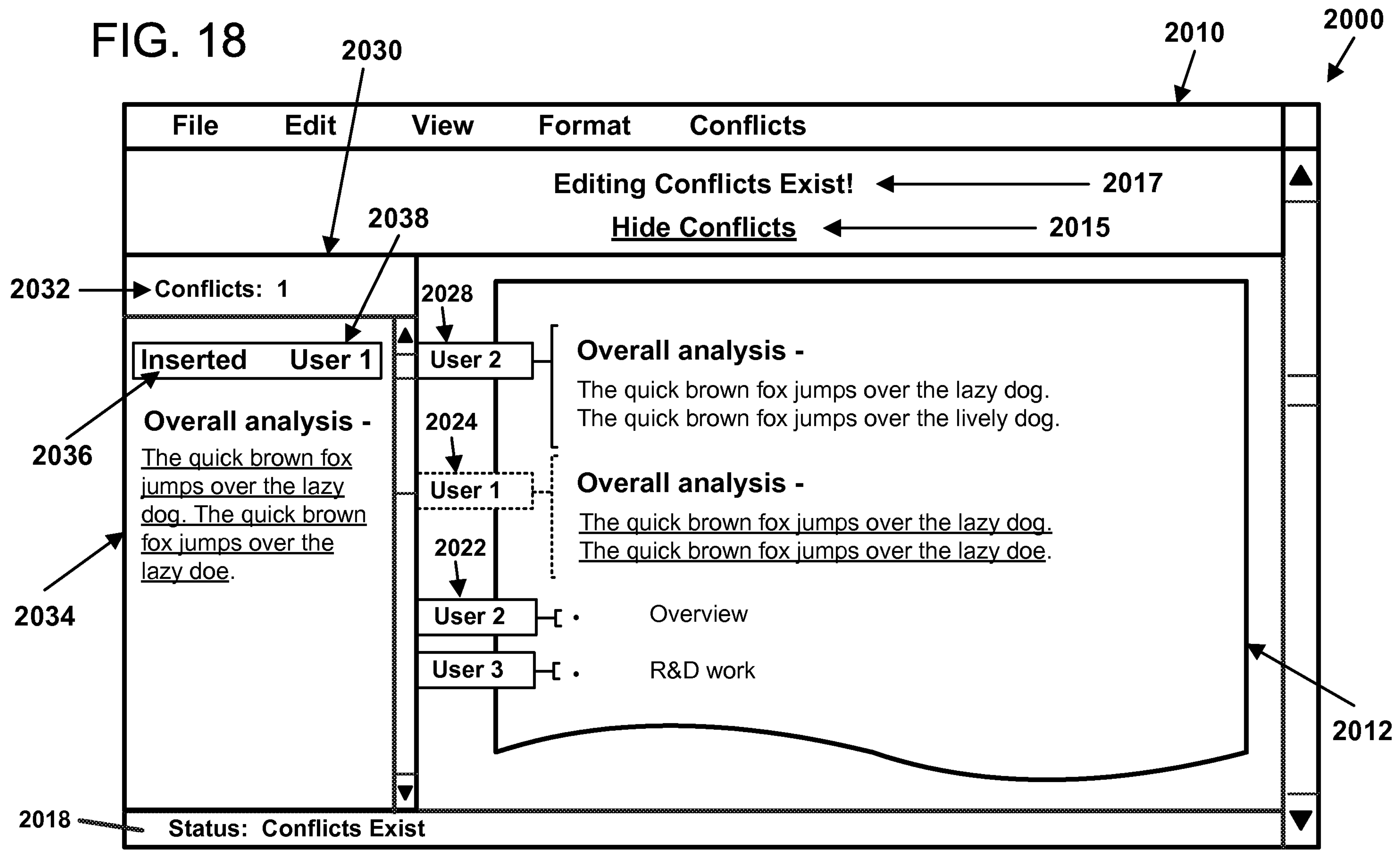
Overall analysis -

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy doe.

- Overview
- R&D work

Status: Conflicts Exist

FIG. 18



Overall analysis -

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy doe.

Overall analysis -

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lively dog.

Overall analysis -

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy doe.

- Overview
- R&D work

FIG. 19

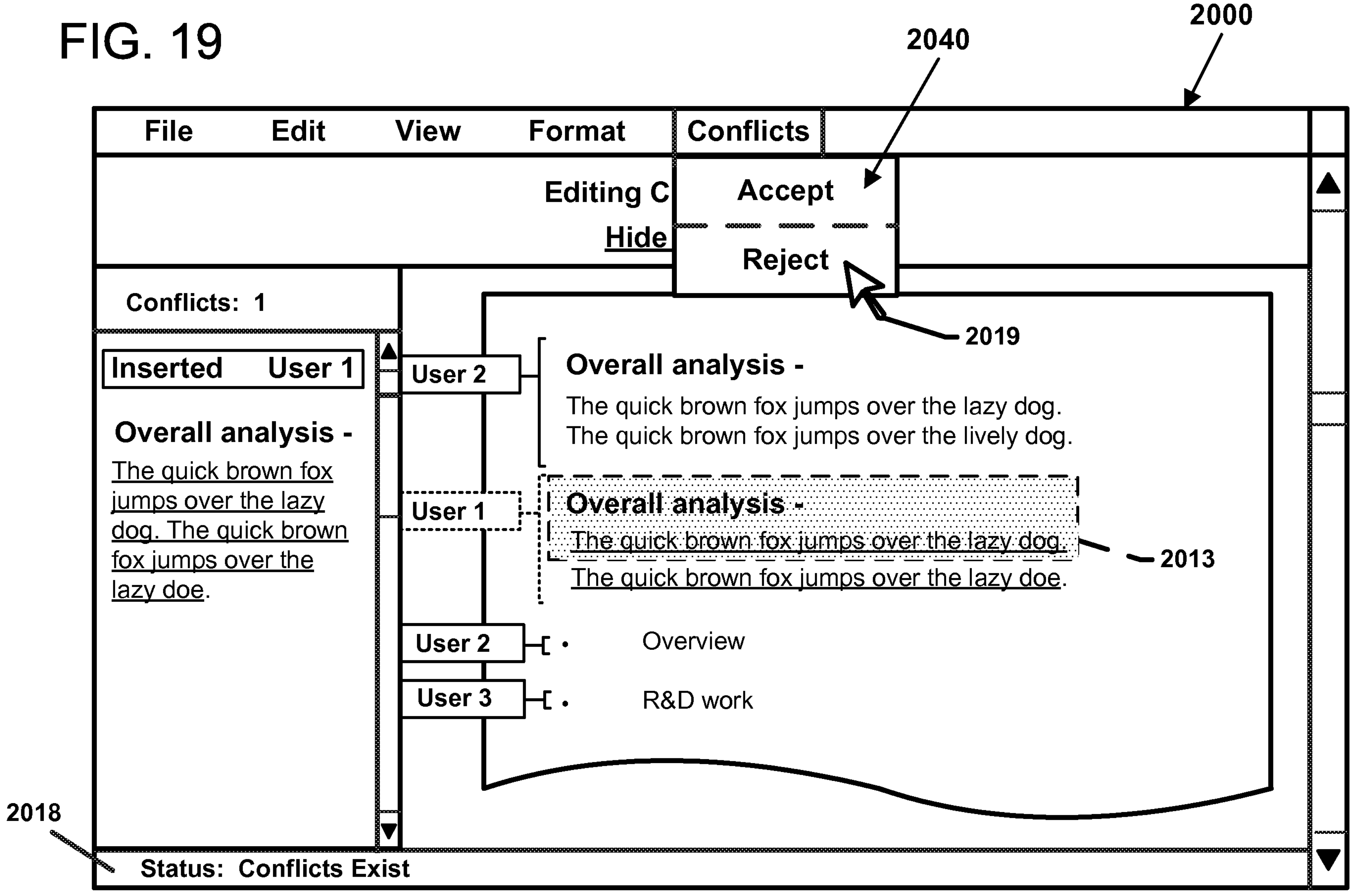
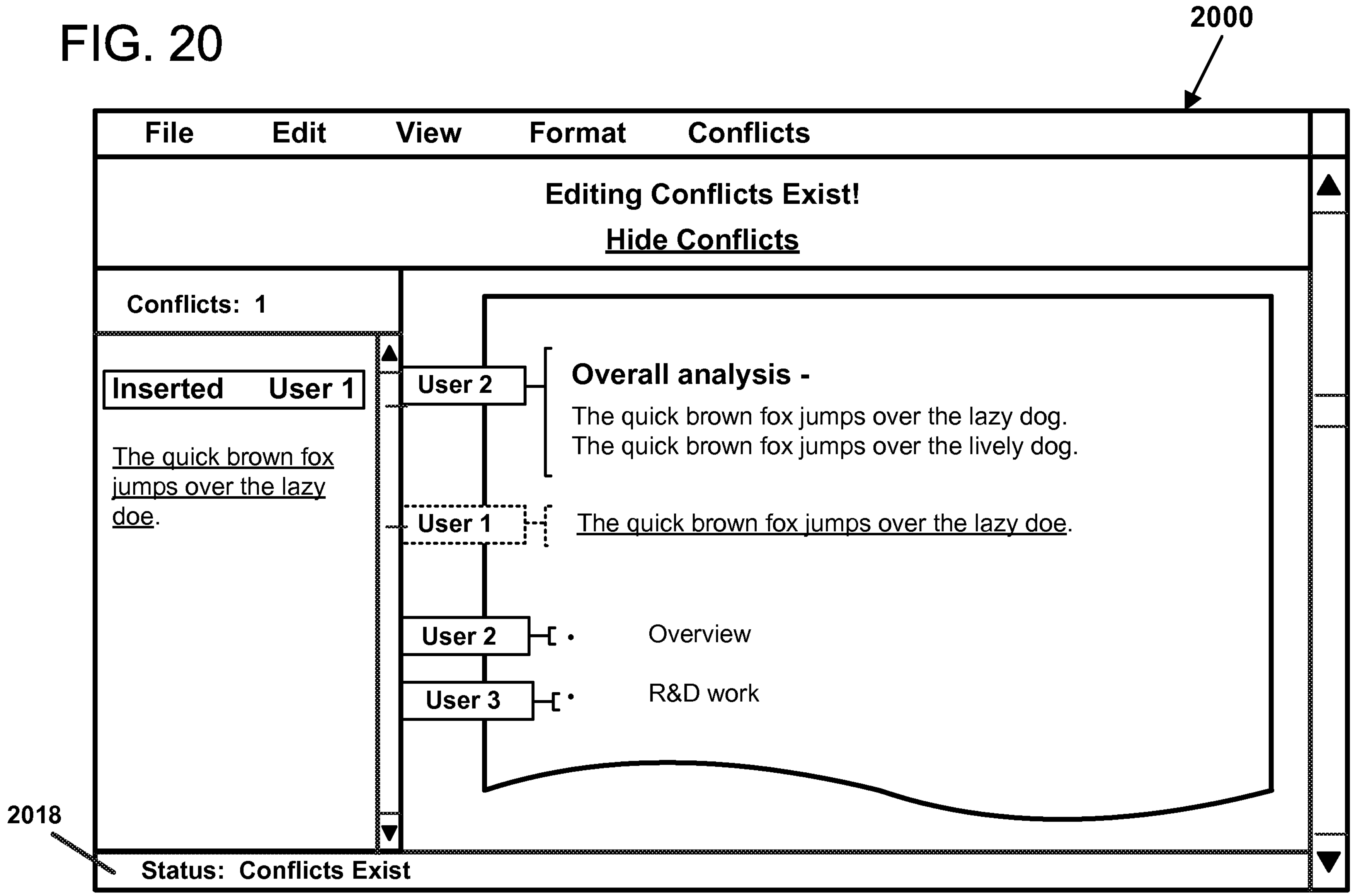

2000
2040
File
Edit
View
Format
Conflicts
Editing C
Hide
Accept
Reject
2019
Conflicts: 1
Inserted User 1
Overall analysis -
The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy doe.
User 2
Overall analysis -
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lively dog.
User 1
Overall analysis -
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy doe.
2013
User 2
Overview
User 3
R&D work
2018
Status: Conflicts Exist

FIG. 20



Editing Conflicts Exist!

Hide Conflicts

Conflicts: 1

Inserted User 1

The quick brown fox jumps over the lazy doe.

User 2

Overall analysis -

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lively dog.

User 1

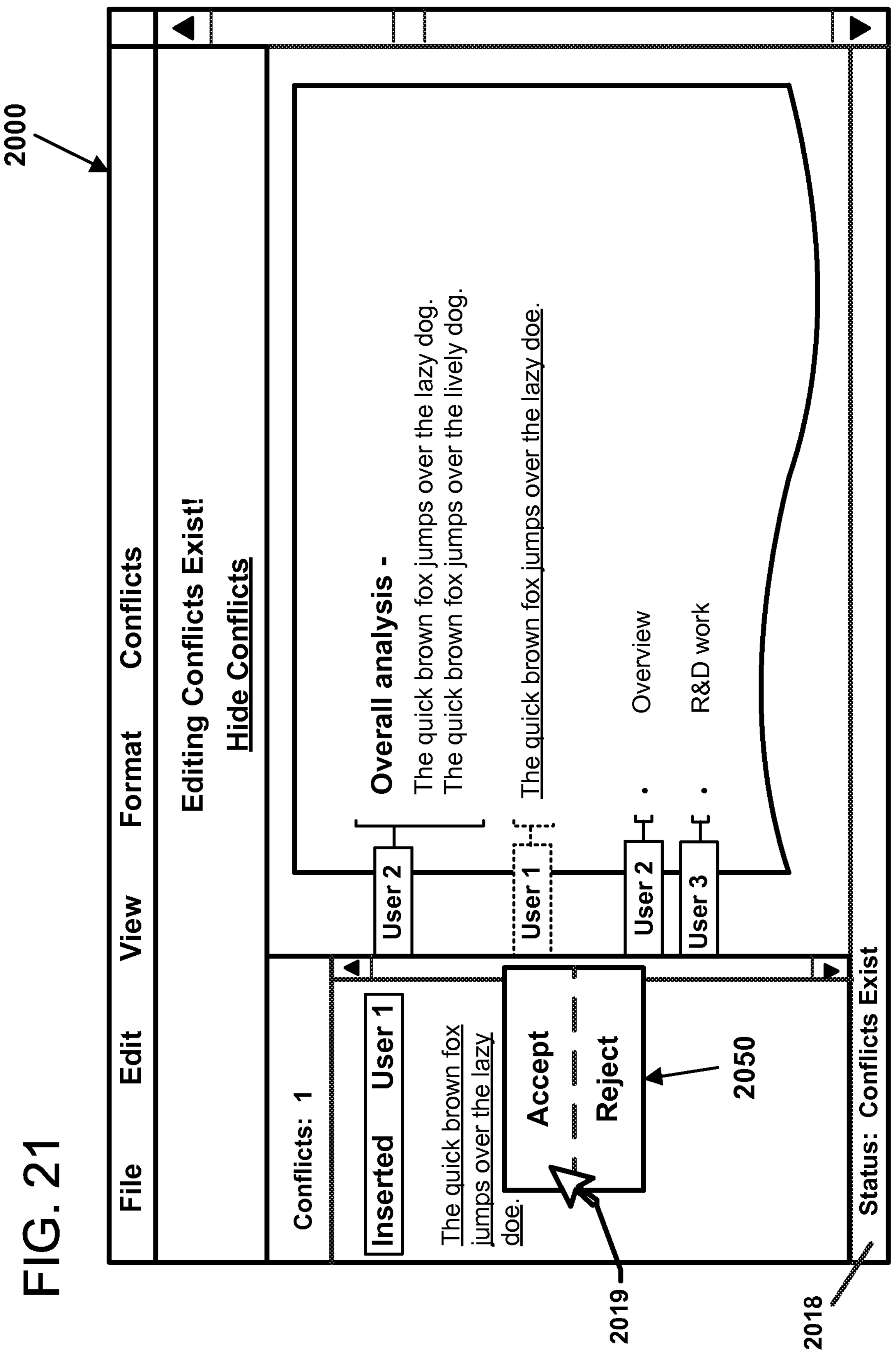
The quick brown fox jumps over the lazy doe.

User 2 • Overview

User 3 • R&D work

Status: Conflicts Exist

FIG. 21



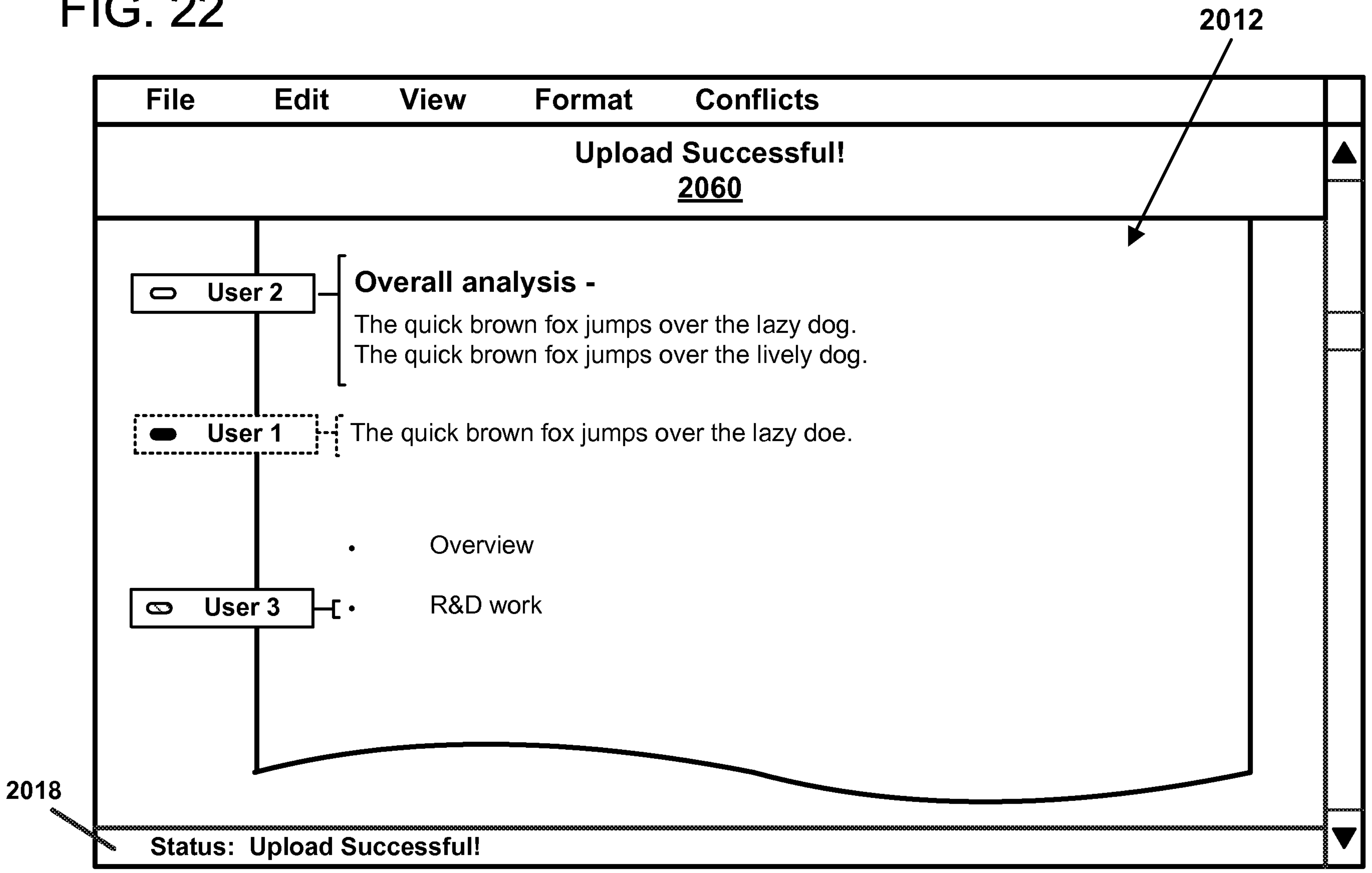
Overall analysis -
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lively dog.

The quick brown fox jumps over the lazy doe.

- Overview
- R&D work

The quick brown fox jumps over the lazy doe.

FIG. 22



Upload Successful!
2060

Overall analysis -
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lively dog.

The quick brown fox jumps over the lazy doe.

- Overview
- R&D work

Status: Upload Successful!

FIG. 23

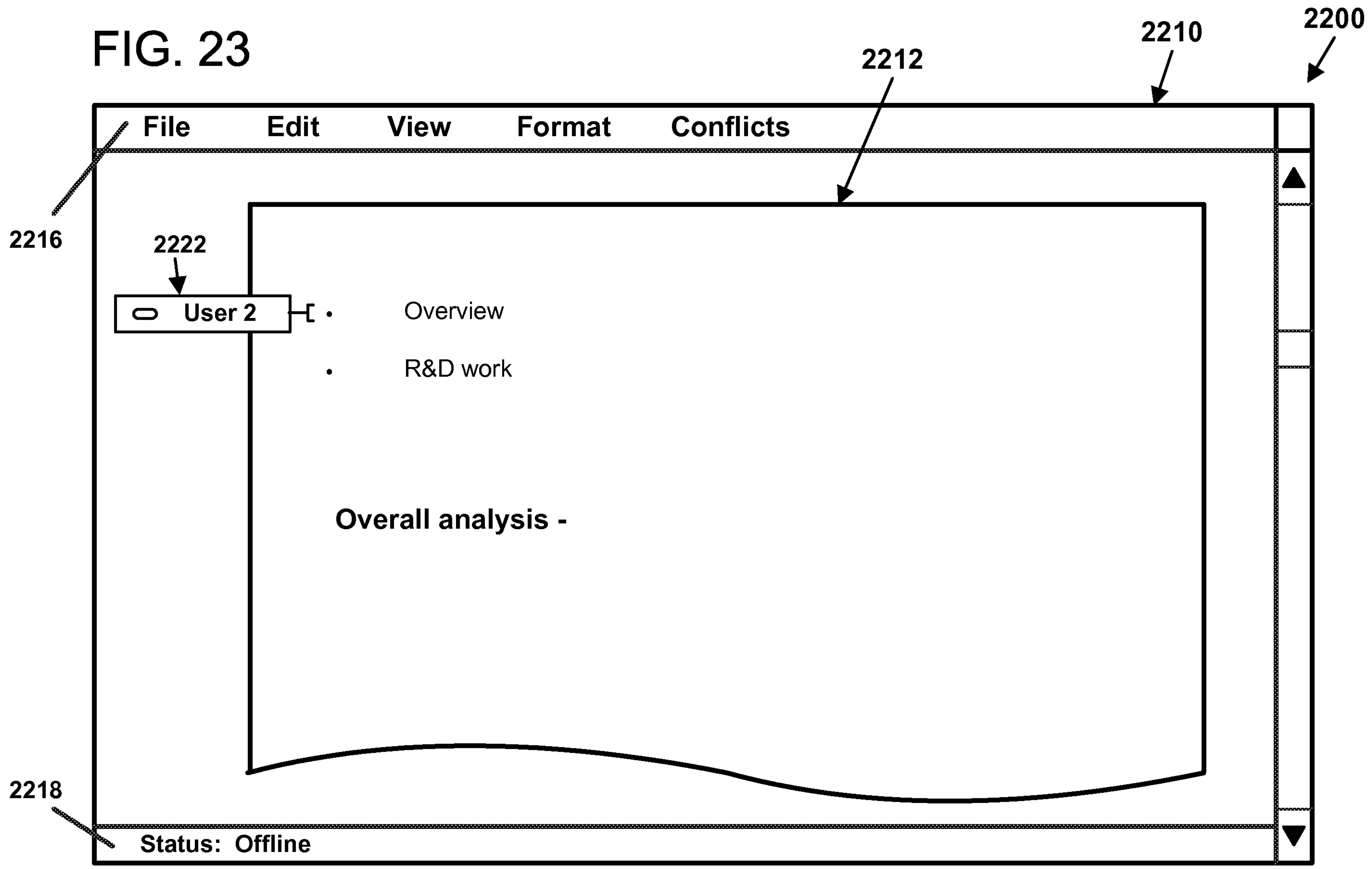
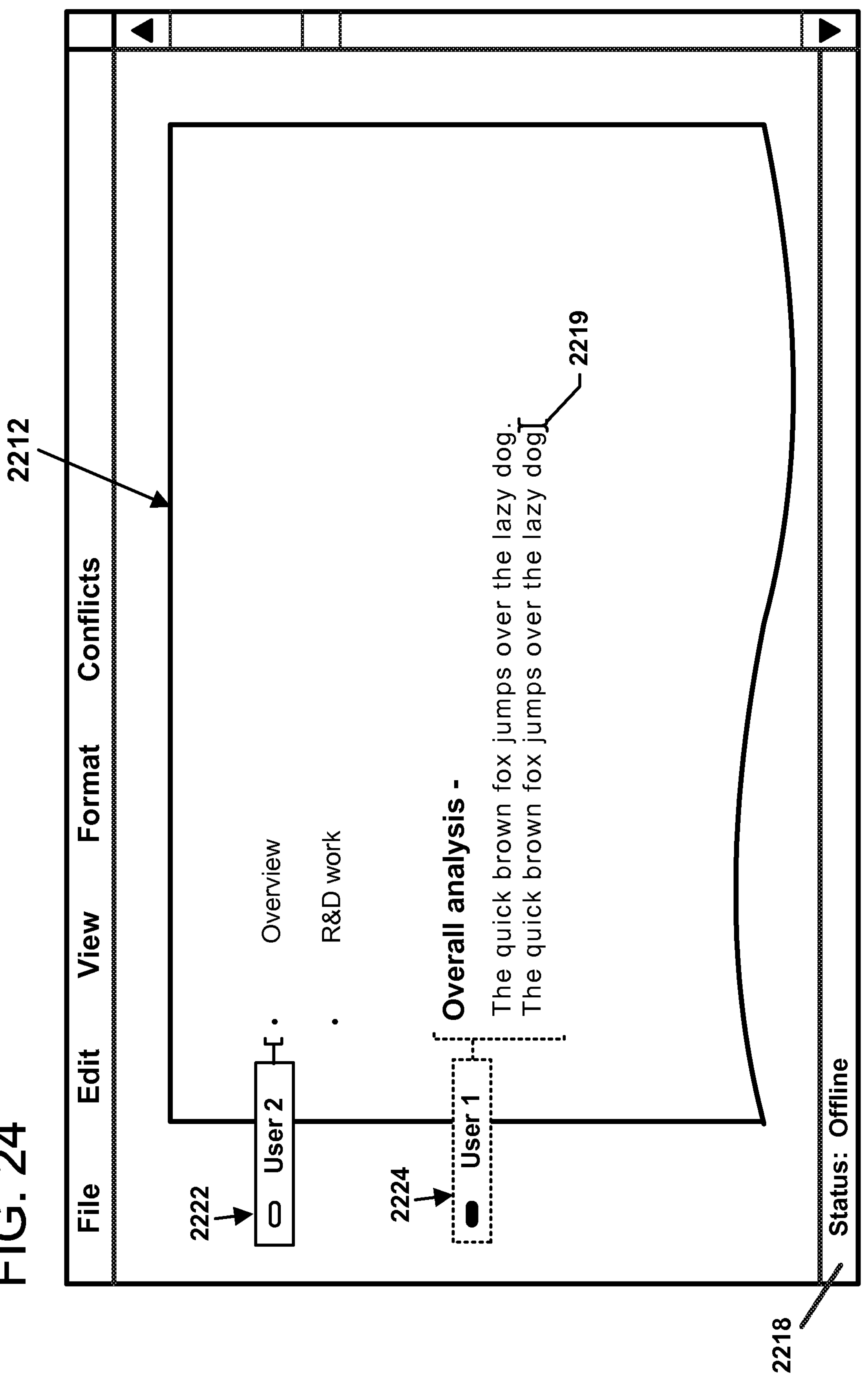

2200
2210
2212
File
Edit
View
Format
Conflicts
2216
2222
User 2
Overview
R&D work
Overall analysis -
2218
Status: Offline

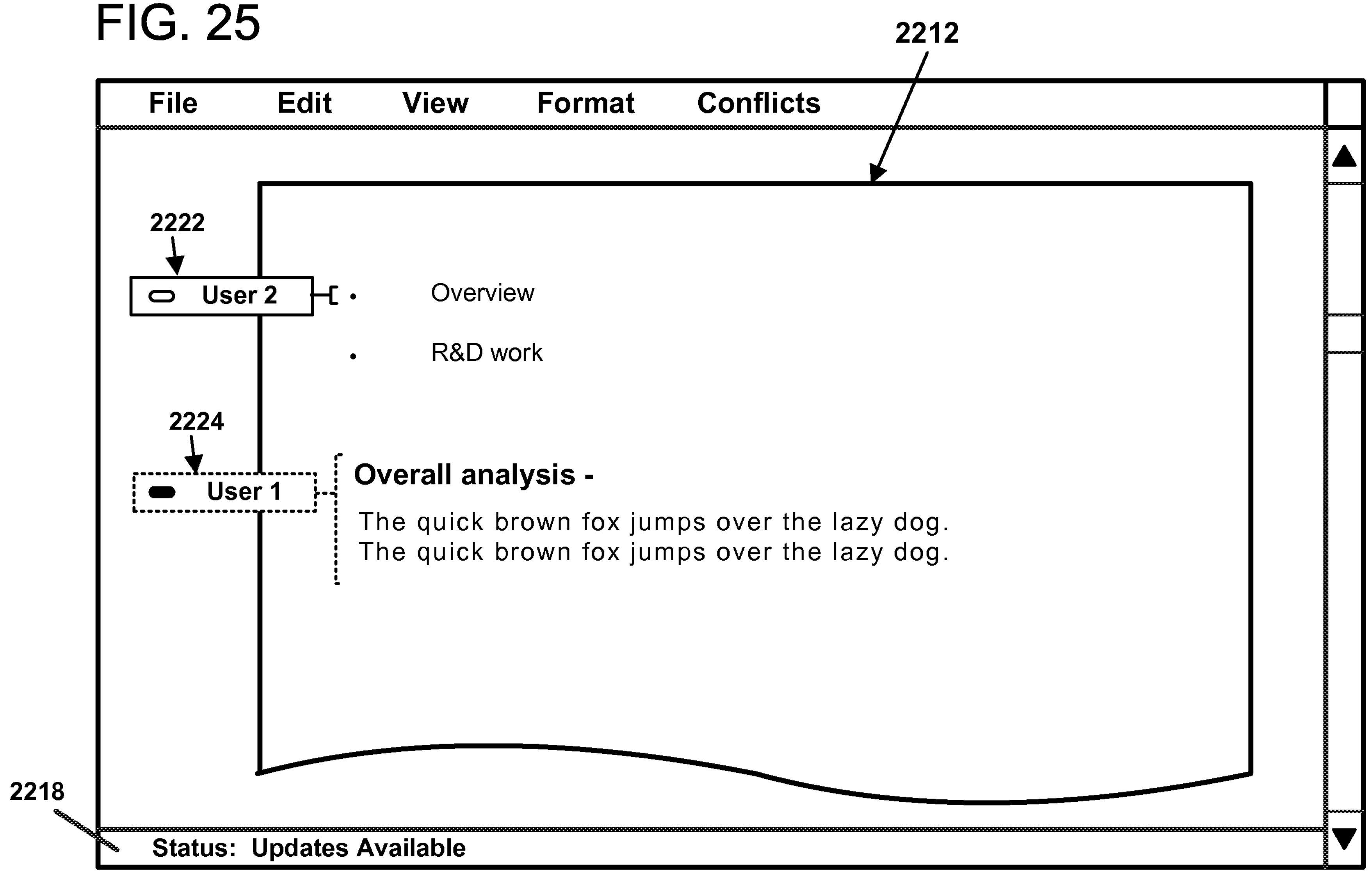
FIG. 24



Overall analysis -

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog

FIG. 25

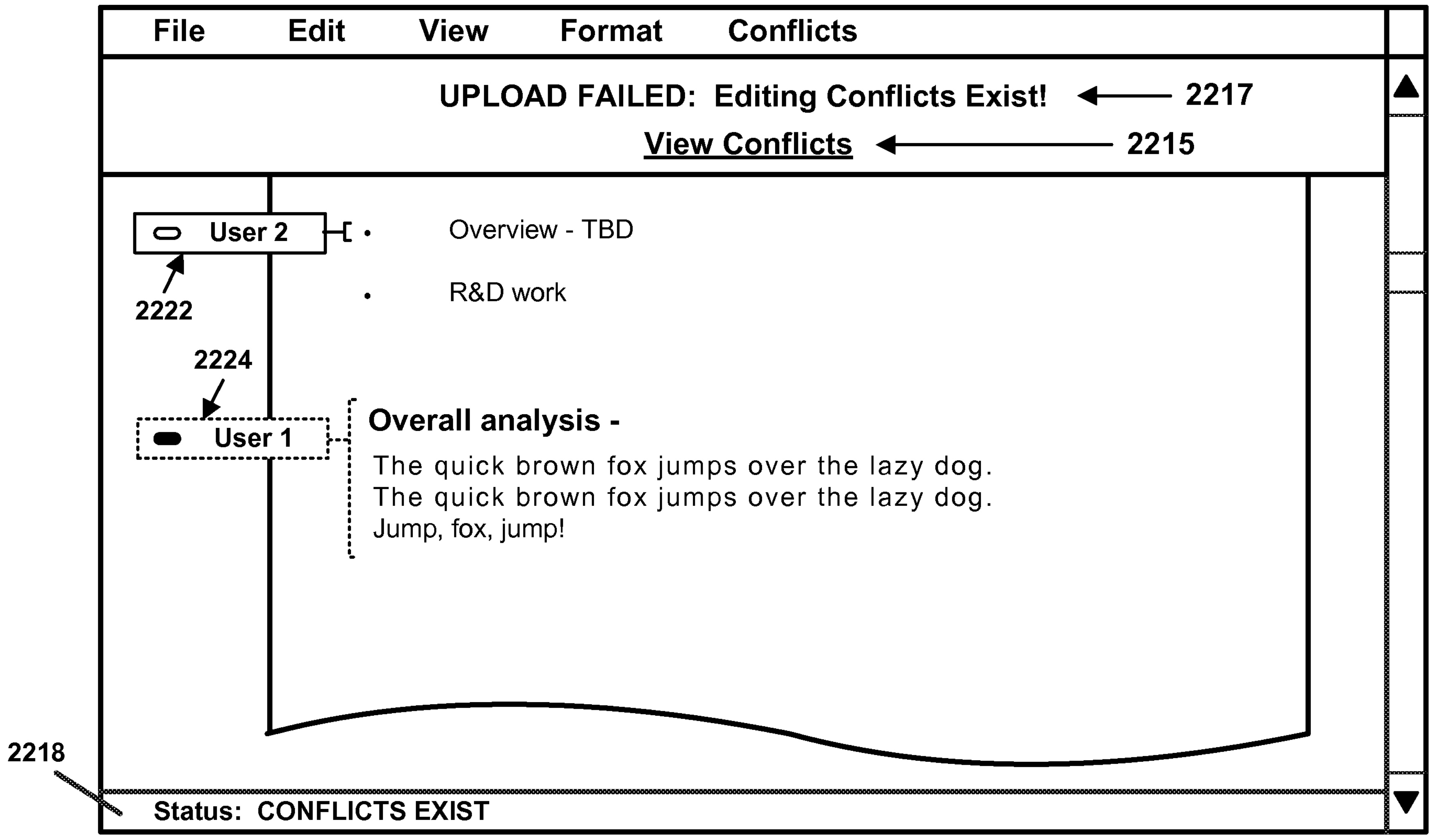


- Overview
- R&D work

Overall analysis -

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

FIG. 26



UPLOAD FAILED: Editing Conflicts Exist!

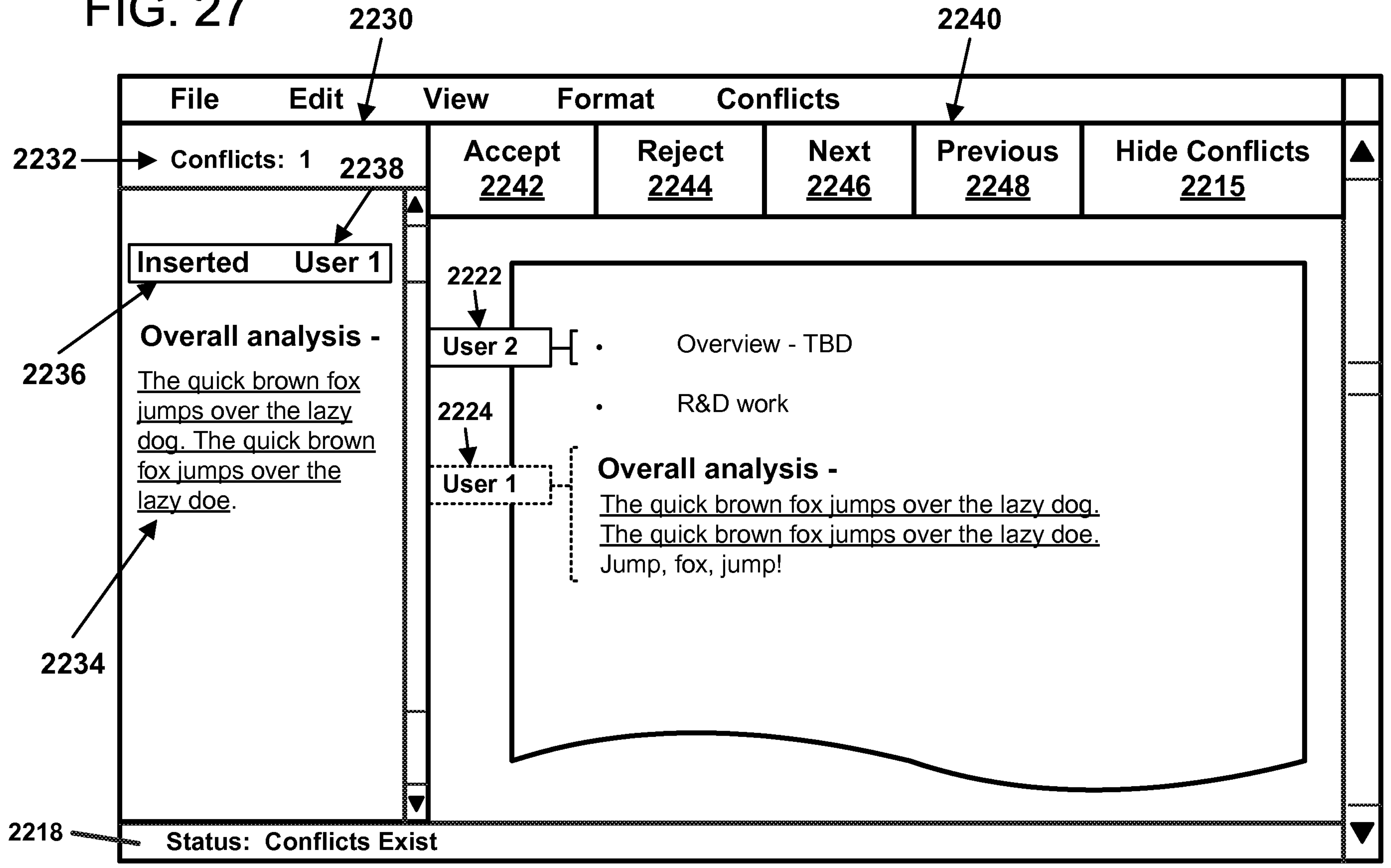
- Overview - TBD
- R&D work

Overall analysis -

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
Jump, fox, jump!

Status: CONFLICTS EXIST

FIG. 27

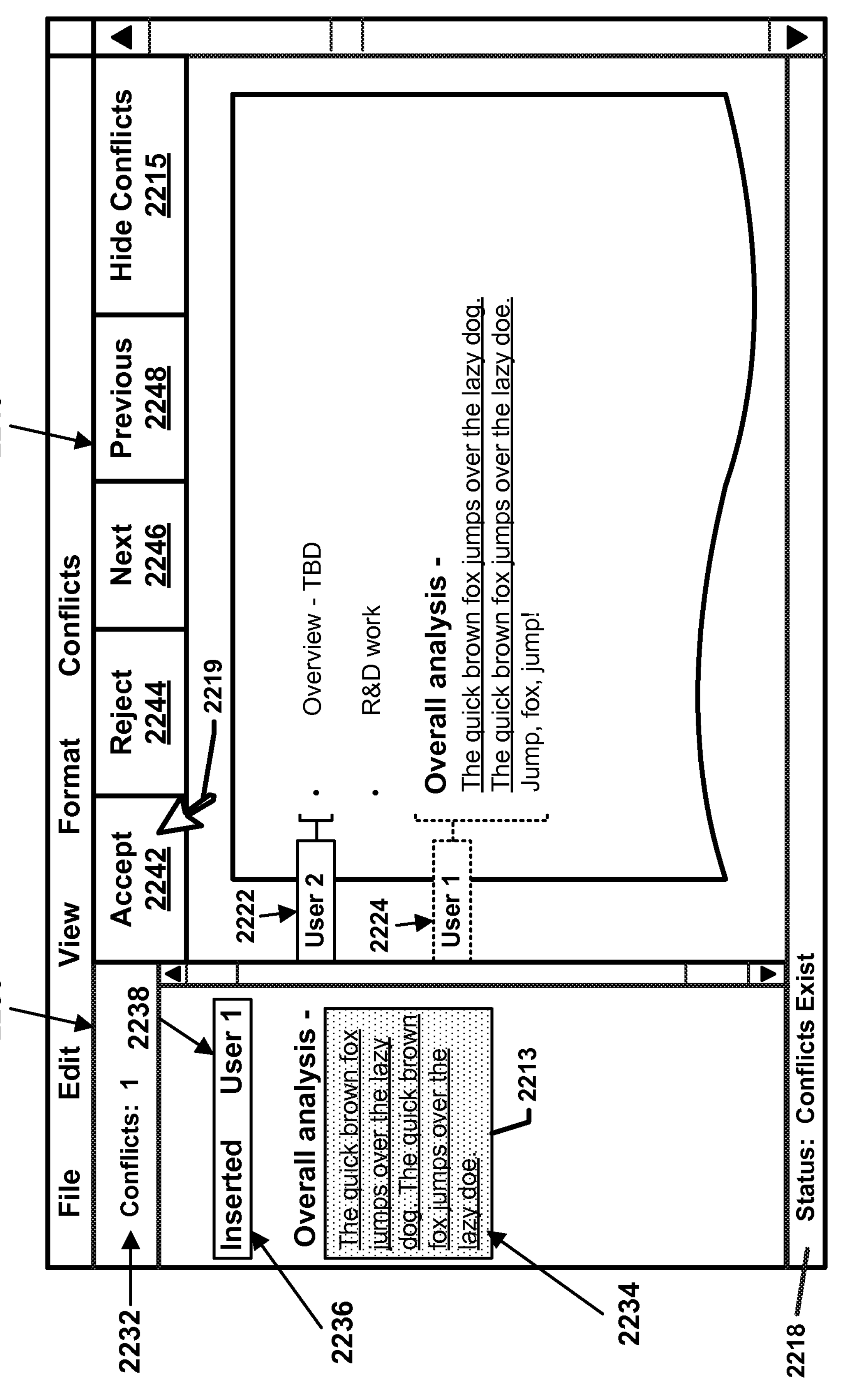


Overall analysis -
The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy doe.

- Overview - TBD
- R&D work

Overall analysis -
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy doe.
Jump, fox, jump!

FIG. 28

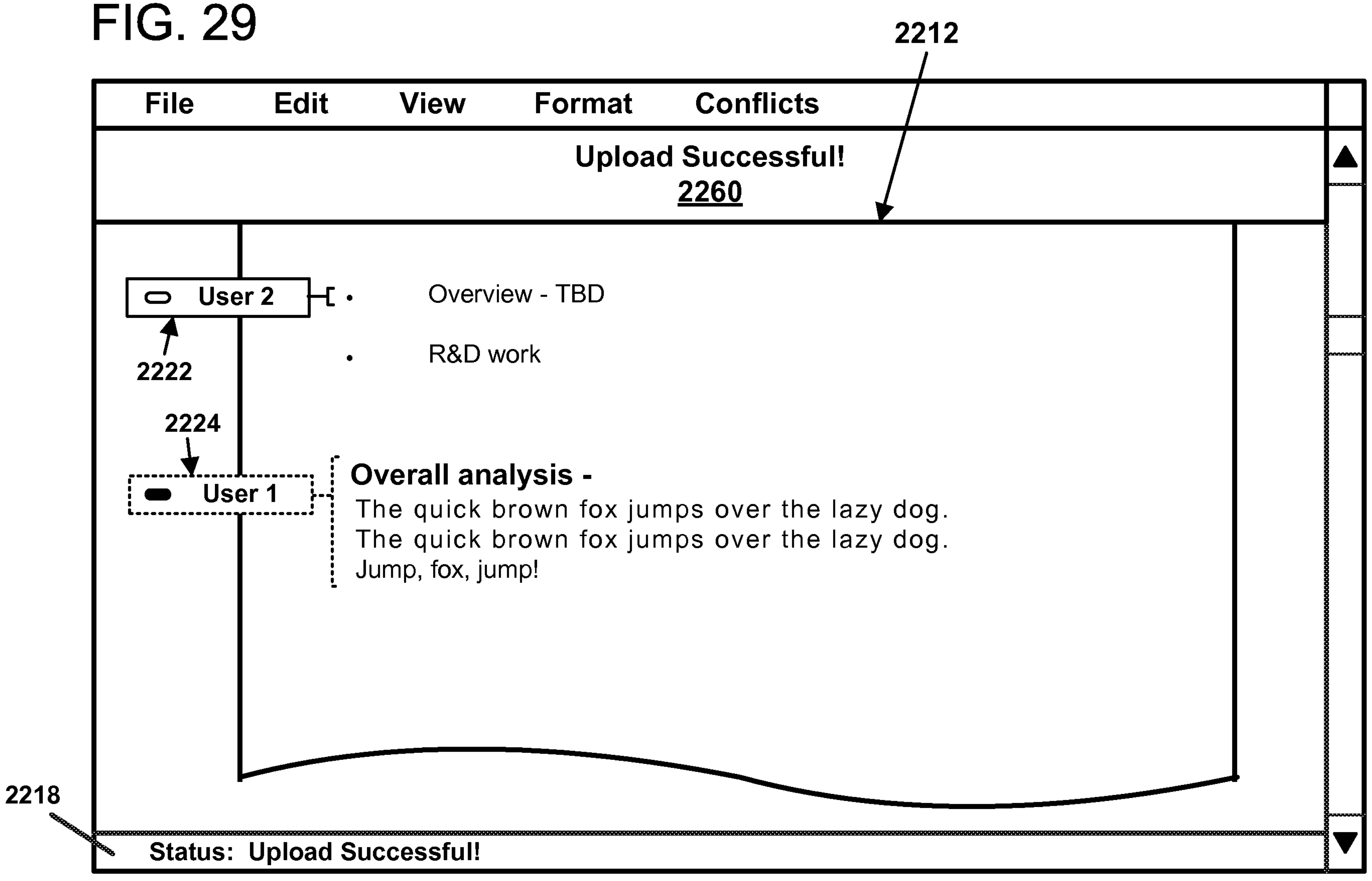


Overall analysis -
The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy doe.

• Overview - TBD
• R&D work

Overall analysis -
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy doe.
Jump, fox, jump!

FIG. 29



Upload Successful!

- Overview - TBD
- R&D work

Overall analysis -
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
Jump, fox, jump!

Status: Upload Successful!