PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F	A2	(11) International Publication Number: **WO 00/60433**
		(43) International Publication Date: 12 October 2000 (12.10.00)

(21) International Application Number: PCT/US00/09457

(22) International Filing Date: 6 April 2000 (06.04.00)

(30) Priority Data:
09/287,866 7 April 1999 (07.04.99) US

(63) Related by Continuation (CON) or Continuation-in-Part (CIP) to Earlier Application
US 09/287,866 (CON)
Filed on 7 April 1999 (07.04.99)

(71) Applicant *(for all designated States except US)*: ATTIVO SOFTWARE, INC. [US/US]; 1200 Villa Street, Mountain View, CA 94041 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: KHEDRO, Taha [SY/US]; 600 Sharon Park Drive #A207, Menlo Park, CA 94025 (US).

(74) Agent: RITTER, Michael, J.; Ritter, Van Pelt & Yi LLP, Suite 205, 4906 El Camino Real, Los Altos, CA 94022 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: TECHNIQUES FOR COMMUNICATING EVENTS AMONG USERS AND APPLICATIONS

(57) Abstract

Techniques for communicating events among users and applications are provided. Application events are monitored and detected so that action messages can be generated that encapsulate the application event and other information about the event. The action messages are sent to interested users so that the underlying application event can be replicated in an application of the interested user. Therefore, application events can be coordinated among multiple users simultaneously.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

TECHNIQUES FOR COMMUNICATING EVENTS AMONG USERS AND APPLICATIONS

BACKGROUND OF THE INVENTION

The present invention relates to communicating (or sharing) information in multiple user environments. More specifically, the invention relates to capturing application events and propagating the application events to interested users so that the users can coordinate working together.

Computers are amazing tools that allow us to perform many tasks efficiently in both the work place and the home. Perhaps the most common application that is run on computers is the word processing program that allows users to produce documents including letters, reports, articles, books, and the like. Computer applications can also be used to produce spreadsheets, slide presentations, databases, drawings, and many more.

Oftentimes, computer projects require that multiple users interact and work together. For example, the documentation for a large software application may be produced by multiple people. A simple way that access to the shared document can be managed is provided by most operating systems by allowing only one user to have access to the document (and have write capabilities).

Of course, allowing only one user to have access to the document means that the users must work serially on the project. Although working serially may be satisfactory for some projects, other projects require that the users are able to work in parallel on the project.

Database applications provide a good example of allowing users to work on projects in parallel. Many database applications allow multiple users to access the database simultaneously. When a user is accessing a record of the database, that record is typically locked so that the integrity of the information stored in the database is maintained.

By locking a record, a database application allows multiple users to access and enter data in the database simultaneously. However, there are many projects where it would be desirable for users to be able to see the changes to the data that other users have made. Moreover, it would be beneficial to have techniques that allow multiple users to work together on projects in a way that allowed them to see the changes that other users have made to the data and decide whether to accept these changes for themselves.

SUMMARY OF THE INVENTION

The present invention provides techniques for communicating application events among multiple users. More specifically, application events are detected and propagated to other interested users (via applications) encapsulated in action messages. The interested users whose privileges allow them to receive the application events can decide on a case by case basis whether to accept the application event. In addition, the interested users can decide to accept or reject all the application events that are received. The invention allows multiple users to work together more efficiently and with more knowledge than is provided prior art techniques. Several embodiments of the invention are described below.

In one embodiment, the invention provides a method of distributing an application event. An event from a user to an application is captured. Once captured, information

about the event is obtained and a message that includes the event and the information about the event is created. The message that is created is then sent to interested users.

In another embodiment, the invention provides a method of propagating an action message. An action message including a header and a self contained object is received. Information from the header and the self contained object is extracted from the action message. The appropriate call to apply data of the self contained object to an application is determined and the call is made to realize the action message in the application. In a preferred embodiment, the call is an application programming interface (API) call.

In another embodiment, the invention provides a method of processing transactions. A transaction for an application including at least one action message is received. An action message in the transaction is selected and it is determined if the action message includes new content. If the action message includes new content, the new content is added to a content list. Users that are interested in the action message are identified and the access privileges of those users that interested are checked. The action message is put in a local queue for the application for the users that are interested and have appropriate access privileges.

In another embodiment, the invention provides a method of sending action messages. A check is made to determine if an application having action messages in a local queue is online on the same activity. If the application is online, a transaction that includes the action messages is sent to a remote agent. Otherwise, if the application is not online, the action messages are saved until the application is online.

In another embodiment, the invention provides a system for communicating action messages among users. The system includes an application that executes on computer systems for multiple users. An agent executes on each computer system that detects application events and sends the application events to a coordinator. The coordinator determines the users that are interested in each application event and sends the application events to the agents on the computer systems of interested users.

Other features and advantages of the invention will become readily apparent upon review of the following description in association of the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates an example of a computer system that can be utilized to execute the software of an embodiment of the invention.

FIG. 2 illustrates a system block diagram of the computer system of FIG. 1.

FIG. 3 shows a network of multiple computer systems.

FIG. 4 shows the interaction of a typical user with an application, such as a word processor or database management system.

FIG. 5 shows a system of one embodiment of the invention that allows users to receive application events generated by other users.

FIG. 6 illustrates queues that may be maintained for each application by an agent of the invention.

FIG. 7 shows queues that may be maintained for each activity of an application by a coordinator of the invention.

FIG. 8 shows a flow chart of a process of distributing an application event to interested users.

FIGS. 9A and 9B show a flow chart of another process of distributing an application event to interested users.

FIG. 10 shows a flow chart of a process of propagating the action message so that the action message can be utilized in the application.

FIGS. 11A-11C show a flow chart of a process of processing transactions so that action messages in the transactions can be sent to interested users that have the appropriate access privileges.

FIG. 12 shows a flow chart of a process of sending action messages when the user is online with the same activity.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

In the description that follows, the present invention will be described in reference to embodiments that detect application events so that the application events can be propagated to interested users who have the appropriate access privileges. However, the invention is not limited to any particular language, computer architecture or specific implementation. Therefore, the description of the embodiments that follows is for purposes of illustration and not limitation.

FIG. 1 illustrates an example of a computer system that can be used to execute the software of an embodiment of the invention. FIG. 1 shows a computer system 1 that includes a display 3, screen 5, cabinet 7, keyboard 9, and mouse 11. Mouse 11 can have one or more buttons for interacting with a graphical user interface. Cabinet 7 houses a CD-ROM drive 13, system memory and a hard drive (see FIG. 2) which can be utilized to store and retrieve software programs incorporating computer code that implements the invention, data for use with the invention, and the like. Although CD-ROM 15 is shown as an exemplary computer readable storage medium, other computer readable storage media including floppy disk, tape, flash memory, system memory, and hard drive can be utilized. Additionally, a data signal embodied in a carrier wave (e.g., in a network including the Internet) can be the computer readable storage medium.

FIG. 2 shows a system block diagram of computer system 1 used to execute the software of an embodiment of the invention. As in FIG. 1, computer system 1 includes monitor 3 and keyboard 9, and mouse 11. Computer system 1 further includes subsystems such as a central processor 51, system memory 53, fixed storage 55 (e.g., hard drive),

removable storage 57 (e.g., CD-ROM drive), display adapter 59, sound card 61, speakers 63, and network interface 65. Other computer systems suitable for use with the invention can include additional or fewer subsystems. For example, another computer system could include more than one processor 51 (i.e., a multi-processor system) or a cache memory.

The system bus architecture of computer system 1 is represented by arrows 67. However, these arrows are illustrative of any interconnection scheme serving to link the subsystems. For example, a local bus could be utilized to connect the central processor to the system memory and display adapter. Computer system 1 shown in FIG. 2 is but an example of a computer system suitable for use with the invention. Other computer architectures having different configurations of subsystems can also be utilized.

FIG. 3 shows a network that connects multiple computer systems, such as the ones shown in FIGS. 1 and 2. As shown, multiple computer systems 1 are interconnected through a network 101. Network 101 allows the computer systems to share data and resources. Typically, the computer system that controls access to data or a resource is called the server because it provides access to client computer systems. The network can be any kind of network including a local area network (LAN) and wide area network (WAN), such as the Internet.

The components of the invention can be implemented on various components of a network. Although the invention will be described in terms of preferred embodiments, it will be apparent to those of skill in the art that the components can be implemented in different ways without departing from the spirit and scope of the invention.

FIG. 4 illustrates a user interacting with a typical application, which in this case is a word processor or database application. A user 201 interacts with a graphical user interface (GUI) 203. GUI 203 allows the user to enter and receive data through an interface that is easy for the user to utilize. When a user interacts with GUI 203, GUI events are sent to application logic 205.

Application logic 205 receives GUI events and processes them to perform the actions requested by the user. The actions can include any number of forms including storing data in a database 207, files 209, retrieving stored data, making calculations on data, and displaying information through GUI 203, just to name a few.

An application programming interface (API) 211 allows for programmatic data entry/access 213. In other words, API 211 provides calls that direct the application to perform specific actions. Thus, the API calls can be thought of as a back door through which to direct the application. Also, APIs can provide access to application events generated by applications as a result of user interaction. Now that a typical application environment has been described, an embodiment of a system of the invention will be described.

FIG. 5 shows a system that allows application events to be detected and propagated to other users according to the invention. A user 301 can interact with applications 303 and 305. The interaction with the applications can be as described in reference to FIG. 4. Accordingly, user 301 may interact with applications 303 and 305 utilizing a graphical user interface for the applications (not shown).

A GUI 307 allows user 301 to, among other things, view application events that have been sent to the user, as well as generated by local applications, and specify what should be done with application events and configure the users preferences. An agent 309 monitors application events including GUI events so that they can be propagated or sent to interested users. Agent 309 can store the application events that are to be sent to the interested users.

In order to better describe the system of FIG. 5, the system will be described in terms of an example where agent 309 has captured an event that may be of interest to other users. Once agent 309 captures an application event, the agent sends the application event to a coordinator 311. When used herein, the term "action message" will be used to denote a message that includes an application event and additional information regarding the application event. In preferred embodiments, multiple application events can be sent together as a transaction in order to make the system more efficient.

Coordinator 311 receives the application event and determines if there are users interested in receiving the event. For example, coordinator 311 can maintain a list of application events in which users are interested. Coordinator 311 determines if there are any users that are interested in the received application event and then determines if the users have the appropriate access privileges to receive the event. For example, the action message can include the access privileges that specify the users that are allowed to receive the application event.

Assuming there are application events that should be sent to an interested user, coordinator 311 determines if the interested user is online on the same activity of the

application in which the application event was generated. If so, coordinator 311 sends an action message, preferably bundled in a transaction, to an agent 313.

Agent 313 monitors applications 315 and 317 for application events. Additionally, agent 313 receives application events from coordinator 311 and can propagate the application events to integrated applications 315 and/or 317. A GUI 319 allows a user 321 to view application events that have been specified should only be committed to applications upon user confirmation. In order to finish this example, GUI 319 could display an application event that was generated by user 301. User 321 could then specify through GUI 319 that the application event should be accepted and agent 313 would make the appropriate calls to commit the application event to the appropriate application, which could be application 315 as an example.

The above has described coordinator 311 as receiving application events and determining users that may be interested in the application events. These functions can also be performed by an agent. Additionally, a GUI can be used to communicate directly with coordinator 311, such as in an embodiment where a web based GUI communicates with the coordinator.

FIG. 5 has been described in a high level to introduce several aspects of the invention. For simplicity, only two users and two applications were shown. However, a system can include many more users and many more applications. Also, there can be multiple coordinators (see dashed lines in FIG. 5) and the agents and coordinators can be on the same or different computer systems. Furthermore, the system was described as propagating an application event from user 301 to user 321, but this process can be bi-directional and the application events can be propagated to multiple interested users.

Although the network configuration of the system shown in FIG. 5 can be varied, a typical configuration is that a client computer system 323 interacts with user 301 and a client computer system 325 interacts with user 321. A server computer system 327 includes coordinator 311 and therefore coordinates communication among multiple users. Other network configurations can also be utilized in different environments.

Each agent can maintain queues of action messages. FIG. 6 shows queues that may be utilized to store action messages for each application. An application 401 is monitored by an agent. The agent maintains an in queue 403 that stores action messages that are to be sent to application 401. Similarly, an out queue 405 stores action messages that have been sent form application 401. The queues are utilized to more efficiently organize the action messages that are being routed through the agent.

Action messages are also routed through the coordinator so it is also beneficial to maintain queues for the coordinator. FIG. 7 shows queues that can be utilized to store action messages for the coordinator. The coordinator can organize the action messages in terms of activity (or session), application, and then whether the action messages are being sent to or from the application of interest. Application 501 has had two (or more) activities 503 and 505 online. For each activity, the coordinator maintains in queues 507 and out queues 509. In queues 507 store action messages that have been received from the agents and out queues 509 store action messages that are to be sent to the agents.

Now that the overall system and some data structures have been described, it may be beneficial to describe processes that occur within a system. FIG. 8 shows a flow chart of a process of distributing an application event. At a step 601, an event from the user to an application is captured. The event is typically an application event and can include

GUI, application logic, and database events. As mentioned previously, the event is typically captured by an agent that monitors events generated by an application as a result of user interaction.

Information about the event is obtained at a step 603. The information can include the identity of the user who is generating the event, the date and time of the event, the object or component on which the event is performed, creator/user, the owner of the object or component, the last updated time, the application in which the event was generated, and access privileges of who can receive the action message including this event.

At a step 605, a message that includes an event and the information about the event is created. The message is then sent to interested users at a step 607. FIG. 8 shows a high level process of distributing an application event and FIGS. 9A and 9B show more details in another embodiment.

FIGS. 9A and 9B show a flow chart of a process of distributing an application event. At a step 651, a user interacts with an application. The user may be entering data, accessing data or performing any other functions within the application. During user interaction, the system generates an application event at a step 653.

At a step 655, the application event is detected and the event type of the application event is determined. In preferred embodiments, the application event is detected and initially processed by an agent running on the system in which the application event was generated. The event type can be that data was added, deleted, modified, and the like.

The object that is associated with the event is found at step 657. Once the object has been identified, information about the object including the schema of the object is extracted at a step 659. At a step 660, it is determined if a change specified by the event is allowed. If not, the application event is disallowed at a step 661, thereby avoiding a conflict. Otherwise, the information and schema will be utilized to generate a self contained object that will be included in an action message as follows.

At a step 662, information for a header of the action message is determined. The information may include the user who caused the application event to be generated, the date of the application event, the owner of the object that is acted on by the application event, the last update date and time of the object, and any other information that may be desired.

A self contained object and schema is created at a step 663. The self contained object and schema (also called an "action object") can include the event type (such as added, deleted, modified, etc.), context of the event, status, and access privileges. For example, the access privileges can be whether the action message, and underlying application event, is private, can be distributed to the public at large, or can only be distributed to certain users.

The schema is the structure of the object and can include the super class, class, and list of properties, where each property includes a name, value, type, and unit. For example, in a computer aided design (CAD) application, the super class of an object could be "entity" and the class could be "circle." The list of properties could include definitions for the radius, center, color and the like of the circle. As an example, the name of the

property could be radius, the value could be "r," the type could be "float," and the unit "inches."

At a step 665, the action message is created and includes the self contained object and the header. Accordingly, the action message is an encapsulation of the application event and includes other information that may be beneficial in replicating the application event in another user's activity of the application. The action message is sent to interested users at 667. In a preferred embodiment, the action message is sent by an agent to a coordinator.

FIG. 10 shows a flow chart of a process of propagating the action message in a activity of an application of another user. At a step 701, the action message is received. In preferred embodiments, the action message is received by an agent that is monitoring the application events of another user.

At a step 703, the information from the message header is extracted. The self contained object is extracted from the action message at a step 705. At a step 706, it is determined if a change specified by the self contained object is allowed. If not, the application of the self contained object is disallowed at a step 707, thereby detecting a conflict.

The self contained object is examined to determine the appropriate call to apply the self contained object data to the application at a step 708. Once the call (e.g., API call) is determined, the call is made to realize the action message, and underlying application event, in the application at a step 709. In preferred embodiments, the flow described in FIG. 10 is performed by an agent. Therefore, FIGS. 9A–9B and 10 have described

processes that may be performed by the agents. FIGS. 11A-11C show a process of processing transactions that will include a process that a coordinator may perform to coordinate between the agents at either end of the action message.

In FIG. 11A, starting with a step 801, a user gets online with an application on a computer system. At a step 803, the user is online with the activity of the application. At some point in time, the user performs an operation that generates an application event. The application event is detected by the agent, which precedes to generate an action message as described in FIGS. 9A and 9B. Once a certain number of action messages have been created, the agent will send the transaction to the coordinator. This is denoted by the user committing a transaction at a step 805.

At a step 807, the coordinator receives the transaction and acknowledges receipt back to the agent. The coordinator starts processing the transaction at a step 809.

As the transaction can include multiple action messages, the coordinator checks if the transaction has more action message to process at a step 811. If it is determined that there are more action messages to process at a step 813, the coordinator gets the next action message to process at a step 815.

At a step 817, the coordinator examines the content of the action message to determine if the action message includes new content. The content can be utilized by users to specify application events in which they are interested. If it is determined that the action message includes new content at a step 819, the coordinator adds the new content to a content list at a step 821.

At a step 823, the coordinator identifies users that are interested in the action message and underlying application event. Once the coordinator has identified interested users, the coordinator checks the access privileges of those interested at a step 825 to determine if they can receive the action message. The access privileges of the action event can be specified in a self contained object in the action message as described previously.

If there are action messages to be sent to interested users (or applications), and their agents, the coordinator puts the appropriate messages in each application's out queue at a step 827. Referring back to FIG. 7, the coordinator may place the action messages in out queues 509 of the activity of the application.

Once the coordinator has placed application messages into local queues that store the action messages to be sent out, the coordinator can send the action messages to the interested users and their agents. FIG. 12 shows a flow chart of a process of sending action messages to interested users. At a step 901, for each application having action messages in the out queue, the coordinator checks if the application is online on the same activity. If it is determined that the application is online on the same activity at a step 903, the coordinator sends a transaction including the action messages to the agent representing that application at a step 905. Once the agent receives the transaction, the agent will respond with an acknowledgement that is received by the coordinator at a step 907.

If it is determined that the application is not online on the same activity, the coordinator saves the action messages in the out queue until the user is online with the activity at a step 909. One the agent receives the transaction including action messages, the agent can propagate the action messages as described in FIG. 10.

At each agent, a user can specify whether action messages will be automatically accepted and committed to an application or whether the user would like to manually review the action messages and accept only those action messages that they are interested. For example, if a user specifies that all action messages should be automatically accepted, the agent will automatically extract the underlying application events and commit them to the appropriate application. However, if the user specifies that the action messages should be manually accepted, the agent will display the underlying application events to the user so that the user can select those that should be accepted and committed.

While the above is a complete description of preferred embodiments of the invention, various alternatives, modifications and equivalents can be used. It should be evident that the invention is equally applicable by making appropriate modifications to the embodiments described above. Therefore, the above description should not be taken as limiting the scope of the invention that is defined by the metes and bounds of the appended claims along with their full scope of equivalents.

CLAIMS

What is claimed is:

1. A method of distributing an application event, comprising:

capturing an event from a user to an application;

obtaining information about the event;

creating a message that includes the event and the information about the event; and

sending the message to interested users.

2. The method of claim 1, wherein the capturing an event includes detecting the event.

3. The method of claim 1, wherein the capturing an event includes determining a type of the event.

4. The method of claim 1, wherein the capturing an event includes finding an object associated with the event.

5. The method of claim 4, wherein the obtaining information includes extracting information about the object.

6. The method of claim 5, wherein the information about the object includes a schema of the object.

7. The method of claim 1, wherein the obtaining information includes determining information for a header of the message.

8. The method of claim 1, wherein the creating a message includes creating a self contained object.

9. The method of claim 8, wherein the creating a message includes creating a schema for the self contained object.

10. The method of claim 9, wherein the message includes the self contained object and schema.

11. A method of propagating an action message, comprising:

receiving an action message including a header and a self contained object;

extracting information from the header from the action message;

extracting the self contained object from the action message;

determining an appropriate call to apply data of the self contained object to an application; and

making the appropriate call to realize the action message in the application.

12. The method of claim 11, wherein the call is an application programming interface (API) call.

13. A method of processing transactions, comprising:

receiving a transaction for an application including at least one action message;

selecting an action message in the transaction;

determining if the action message includes a new content;

if the action message includes a new content, adding the new content to an content list;

identifying for users that are interested in the action message;

checking access privileges of the users that are interested; and

putting the action message in a local queue for the application for the users that are interested and have appropriate access privileges.

14. The method of claim 13, wherein the local queue for the application is for a specific activity of the application.

15. The method of claim 13, wherein the users specify action messages of interest according to the contents of the action messages of interest.

16. The method of claim 13, wherein the action message specifies the access privileges.

17. The method of claim 16, wherein the access privileges are public, private or a list of users.

18. A method of sending action messages, comprising:

checking if an application having action messages in a local queue is online on a same activity;

if the application is online, sending a transaction that includes the action messages to a remote agent; and

if the application is not online, saving the action messages until the application is online.

19. The method of claim 18, further comprising receiving an acknowledgement that the transaction was received by the remote agent.

20. The method of claim 18, wherein the agent commits the action messages if a user has specified automatic acceptance of the action messages.

21. The method of claim 18, wherein the agent displays the action messages to a user for committing if the user has specified user acceptance of the action messages.

22. A system for communicating action messages among users, comprising:

an application that executes on computer systems for a plurality of users;

an agent that executes on each computer system that detects application events and sends the application events to a coordinator; and

the coordinator that determines the users that are interested in each application event and sends the application events to the agents on the computer systems of interested users.

23. The system of claim 22, wherein the coordinator checks if each interested user has the privilege of receiving the application event before the application event is sent.

24. The system of claim 22, wherein each user can specify whether the application events are automatically accepted or will be accepted by the user.

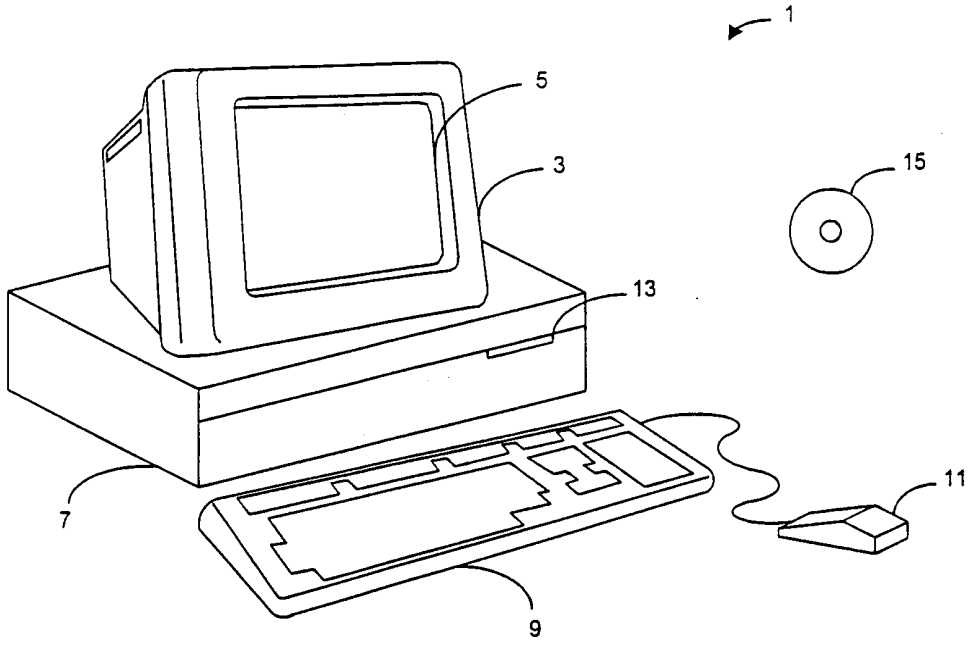

1
5
3
15
13
11
7
9

FIG. 1

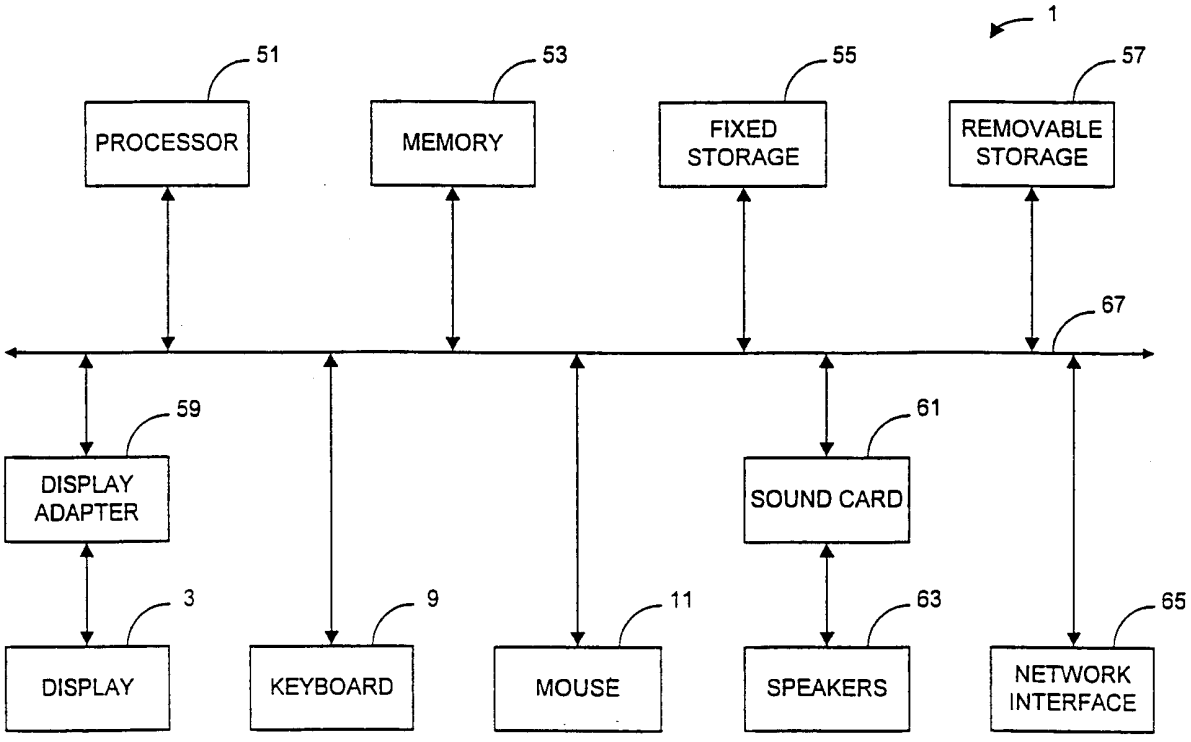

1
51
53
55
57
PROCESSOR
MEMORY
FIXED STORAGE
REMOVABLE STORAGE
67
59
61
DISPLAY ADAPTER
SOUND CARD
3
9
11
63
65
DISPLAY
KEYBOARD
MOUSE
SPEAKERS
NETWORK INTERFACE

FIG. 2

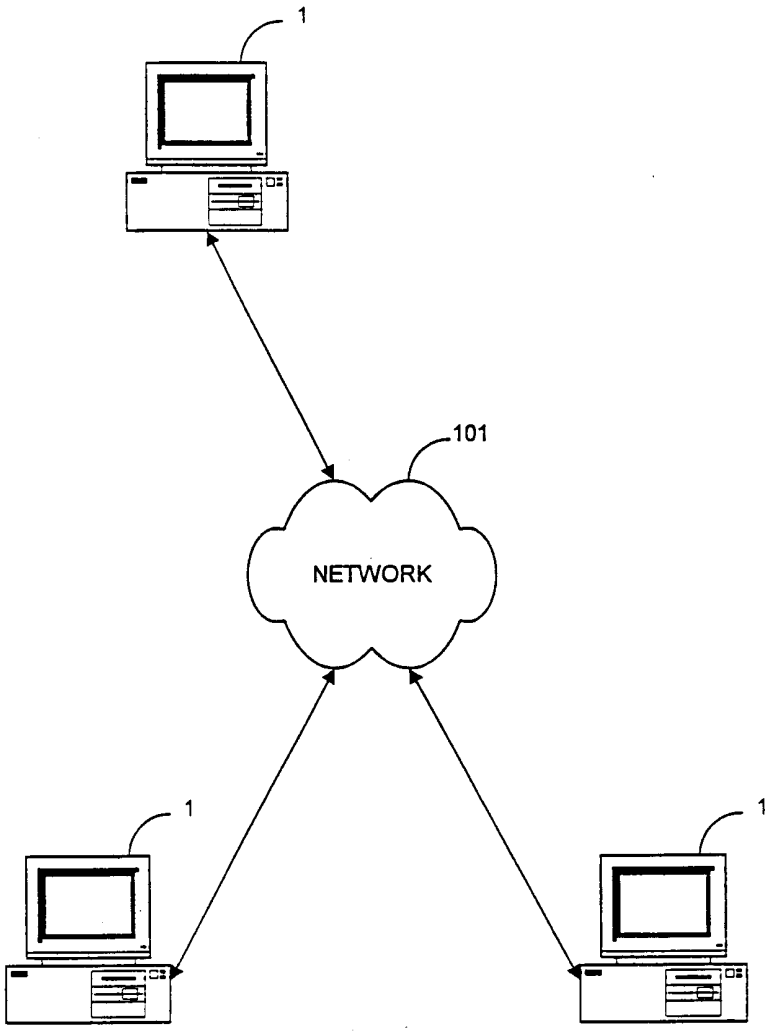


FIG. 3

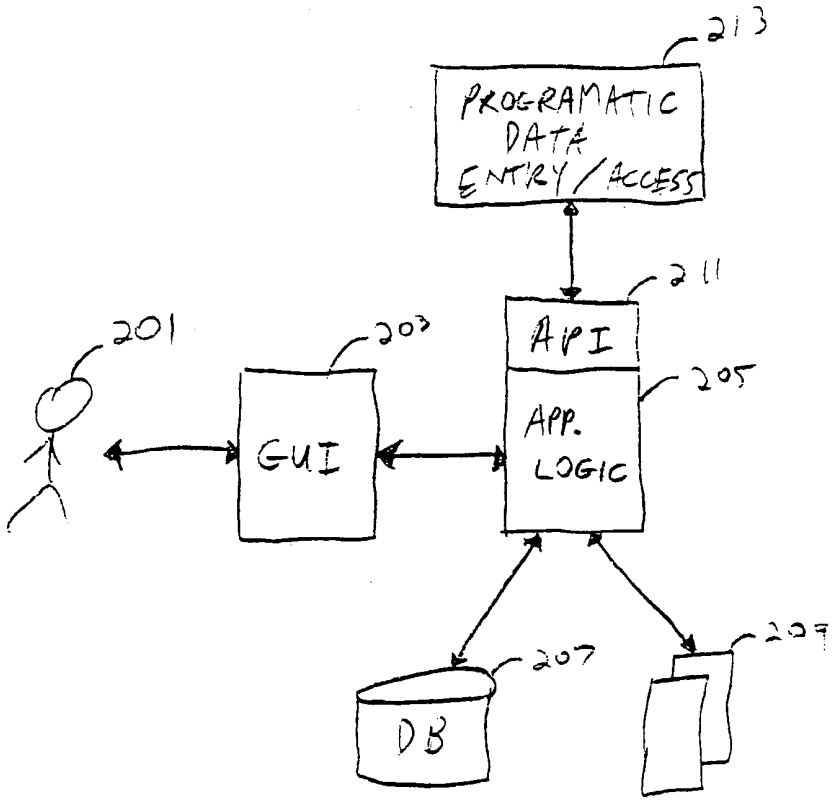

213
PROGRAMATIC DATA ENTRY/ACCESS
211
API
201
203
GUI
205
APP. LOGIC
207
DB

FIG. 4

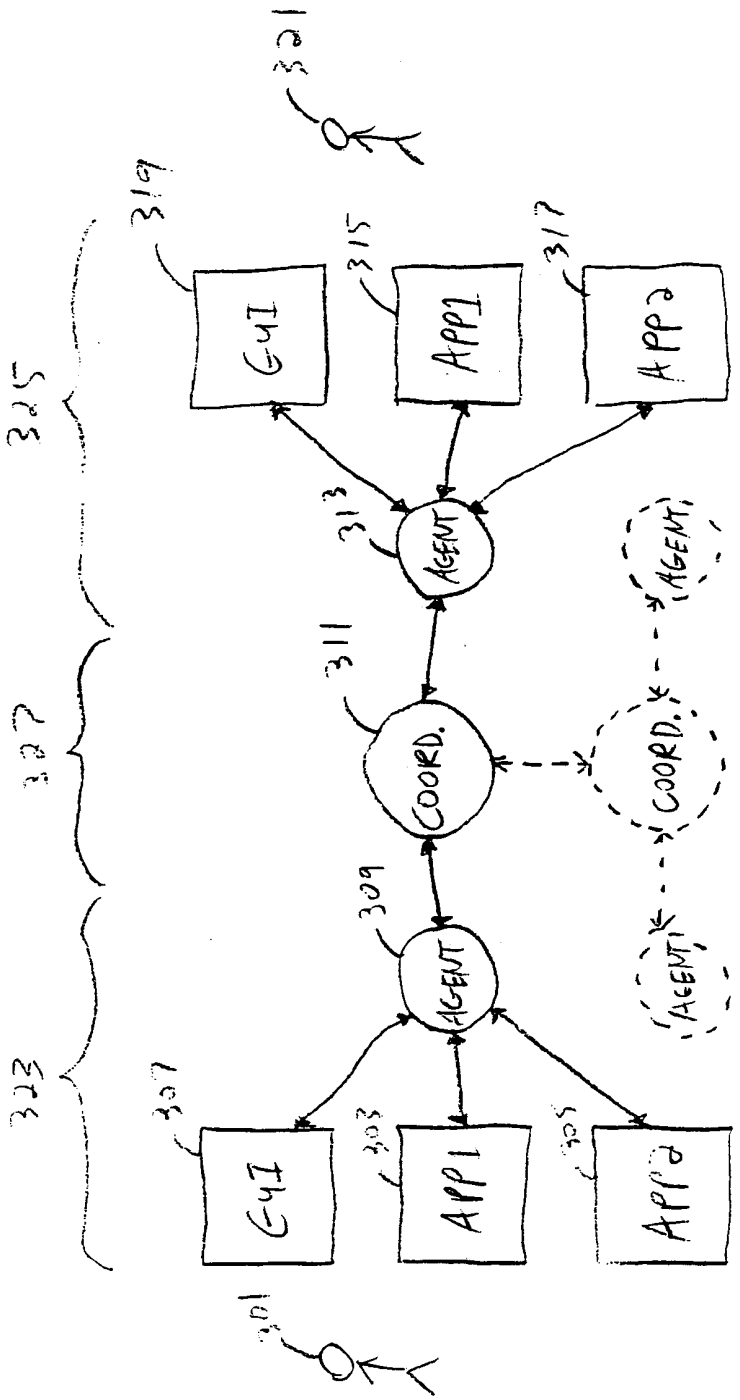

321
319
315
317
GUI
APP1
APP2
325
313
AGENT
311
COORD.
327
309
AGENT
COORD.
AGENT
AGENT
323
307
GUI
303
APP1
305
APP2
301

FIG. 5

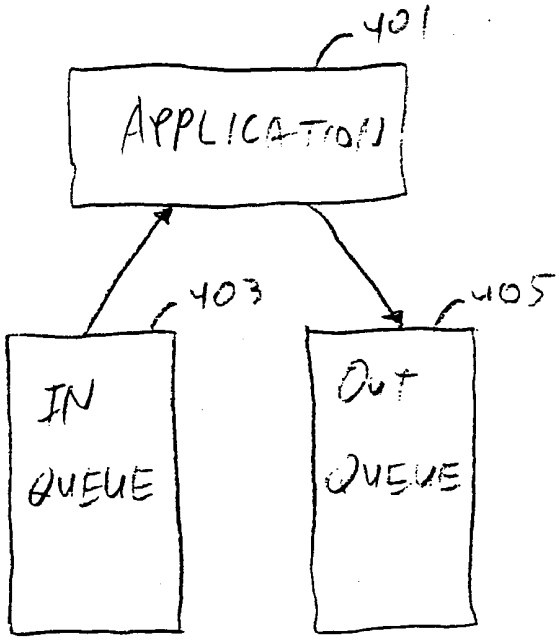


FIG. 6

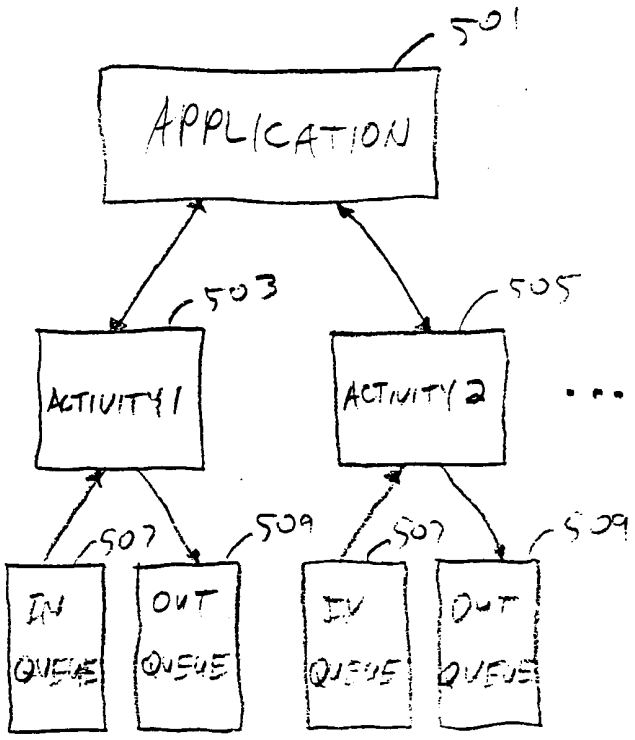

501
APPLICATION
503
505
ACTIVITY 1
ACTIVITY 2
507
509
507
509
IN QUEUE
OUT QUEUE
IN QUEUE
OUT QUEUE

FIG. 7

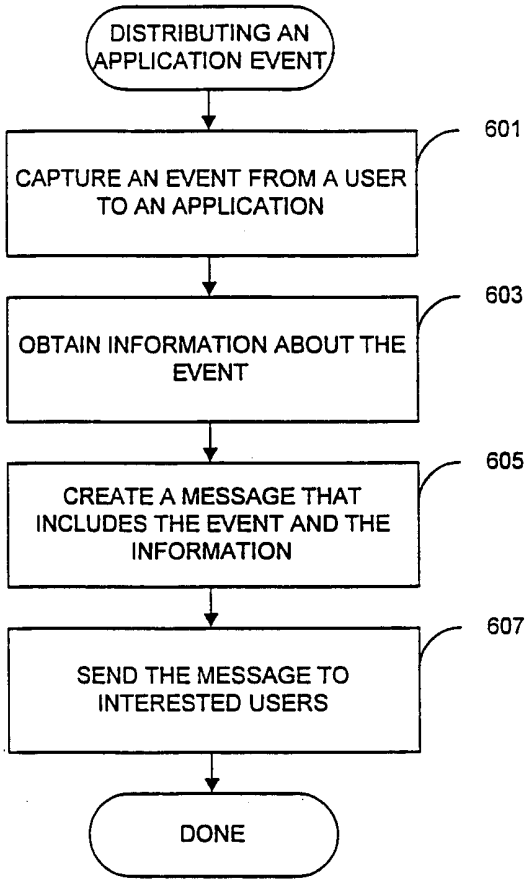

DISTRIBUTING AN APPLICATION EVENT
601
CAPTURE AN EVENT FROM A USER TO AN APPLICATION
603
OBTAIN INFORMATION ABOUT THE EVENT
605
CREATE A MESSAGE THAT INCLUDES THE EVENT AND THE INFORMATION
607
SEND THE MESSAGE TO INTERESTED USERS
DONE

FIG. 8

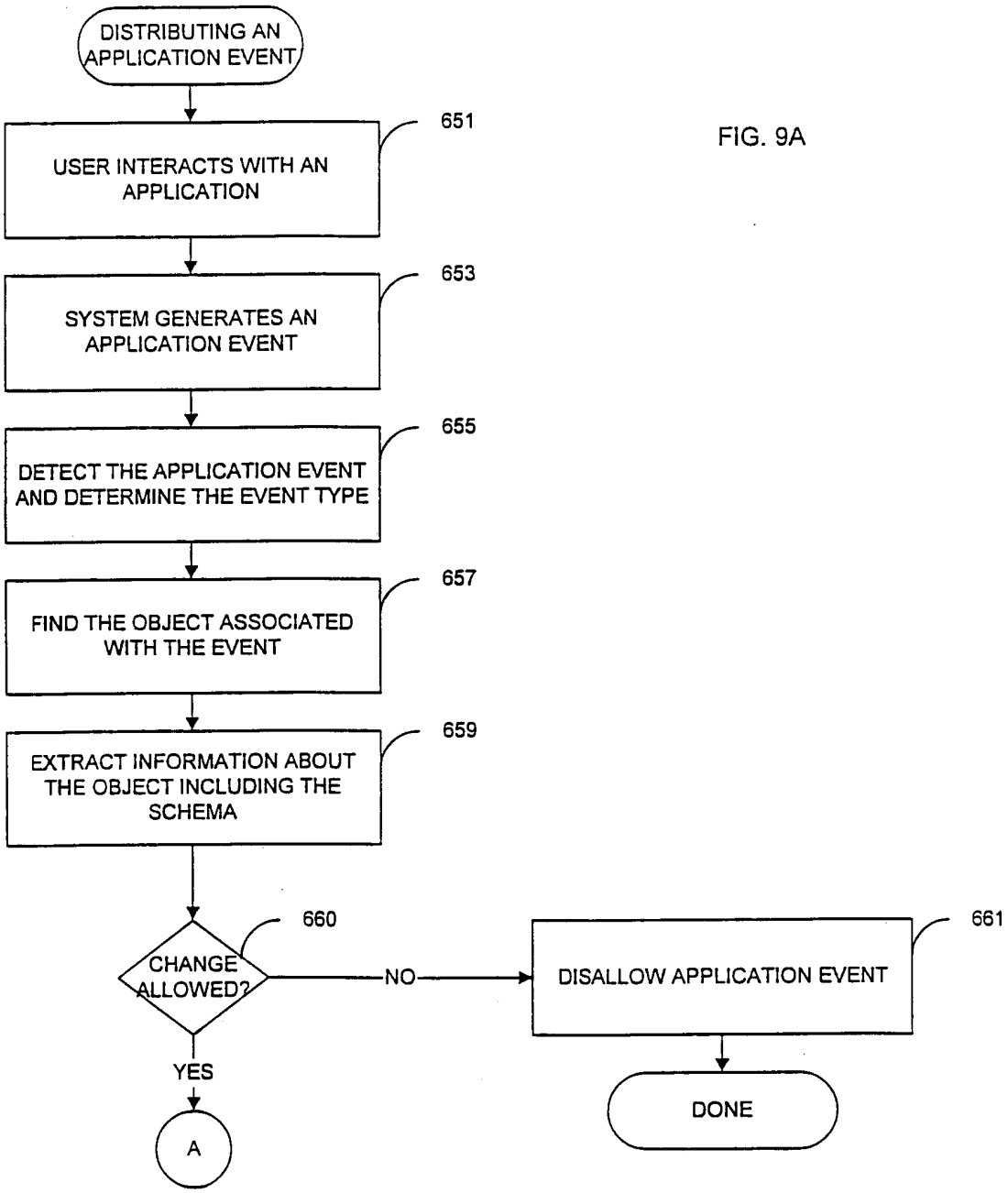

DISTRIBUTING AN APPLICATION EVENT
USER INTERACTS WITH AN APPLICATION
651
SYSTEM GENERATES AN APPLICATION EVENT
653
DETECT THE APPLICATION EVENT AND DETERMINE THE EVENT TYPE
655
FIND THE OBJECT ASSOCIATED WITH THE EVENT
657
EXTRACT INFORMATION ABOUT THE OBJECT INCLUDING THE SCHEMA
659
CHANGE ALLOWED?
660
NO
DISALLOW APPLICATION EVENT
661
YES
A
DONE

FIG. 9A

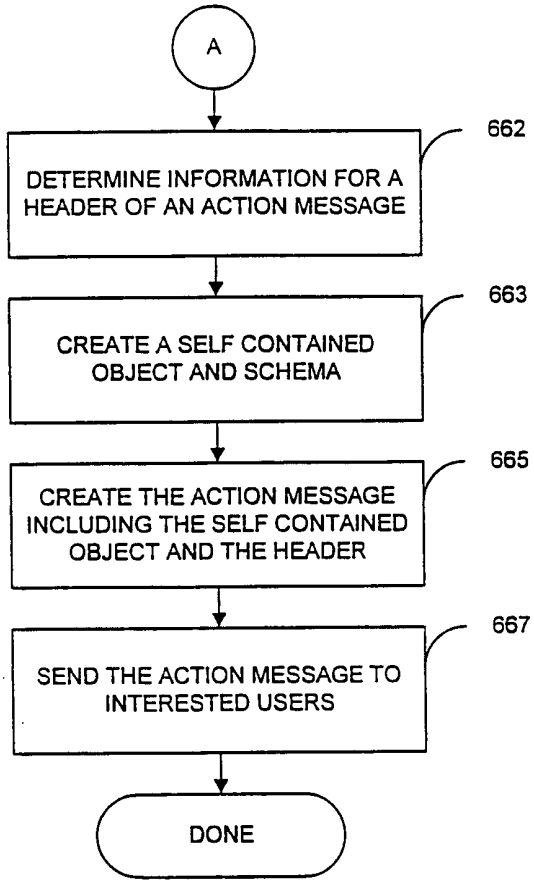

A
662
DETERMINE INFORMATION FOR A HEADER OF AN ACTION MESSAGE
663
CREATE A SELF CONTAINED OBJECT AND SCHEMA
665
CREATE THE ACTION MESSAGE INCLUDING THE SELF CONTAINED OBJECT AND THE HEADER
667
SEND THE ACTION MESSAGE TO INTERESTED USERS
DONE

FIG. 9B

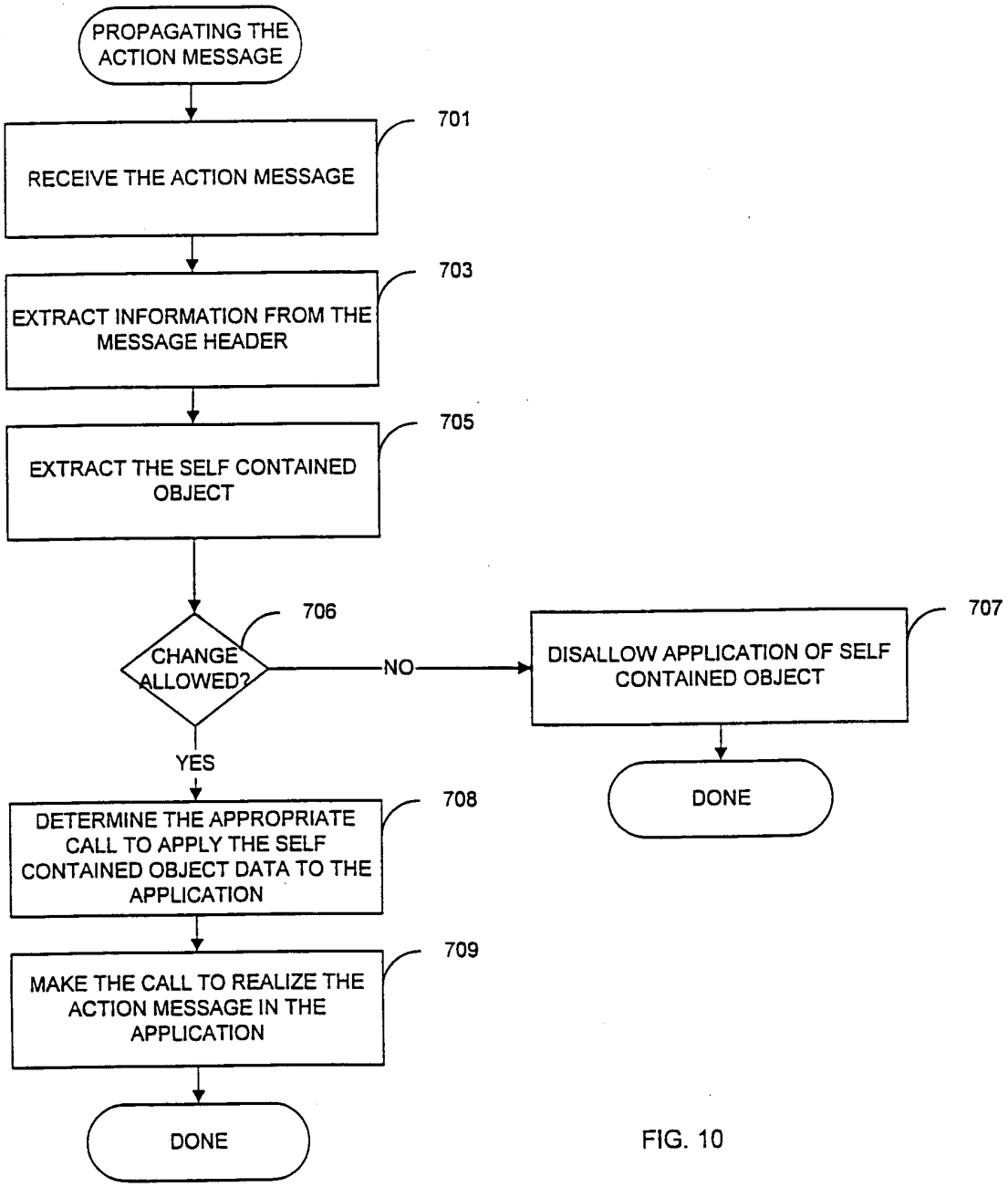

PROPAGATING THE ACTION MESSAGE
701
RECEIVE THE ACTION MESSAGE
703
EXTRACT INFORMATION FROM THE MESSAGE HEADER
705
EXTRACT THE SELF CONTAINED OBJECT
706
CHANGE ALLOWED?
NO
707
DISALLOW APPLICATION OF SELF CONTAINED OBJECT
DONE
YES
708
DETERMINE THE APPROPRIATE CALL TO APPLY THE SELF CONTAINED OBJECT DATA TO THE APPLICATION
709
MAKE THE CALL TO REALIZE THE ACTION MESSAGE IN THE APPLICATION
DONE

FIG. 10

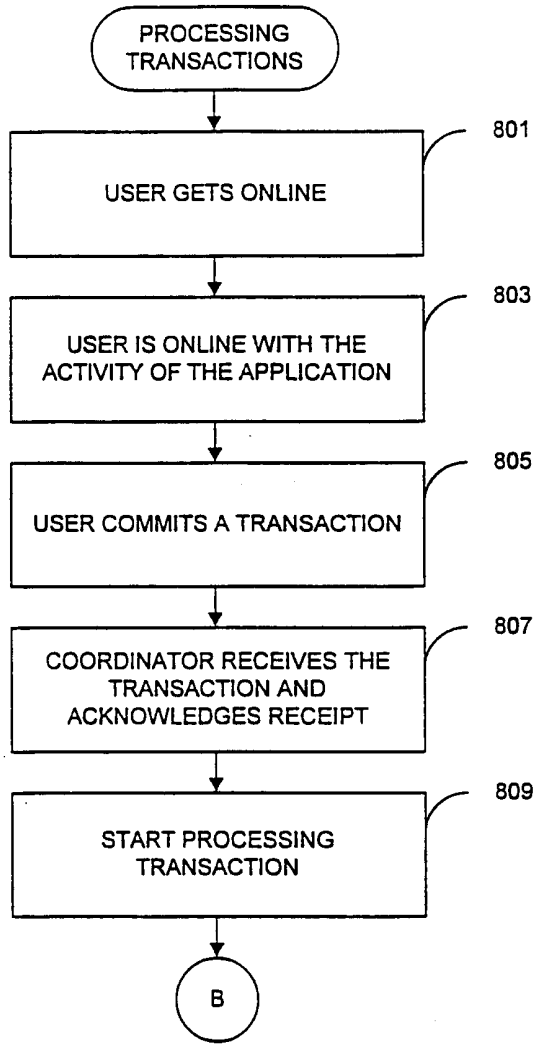
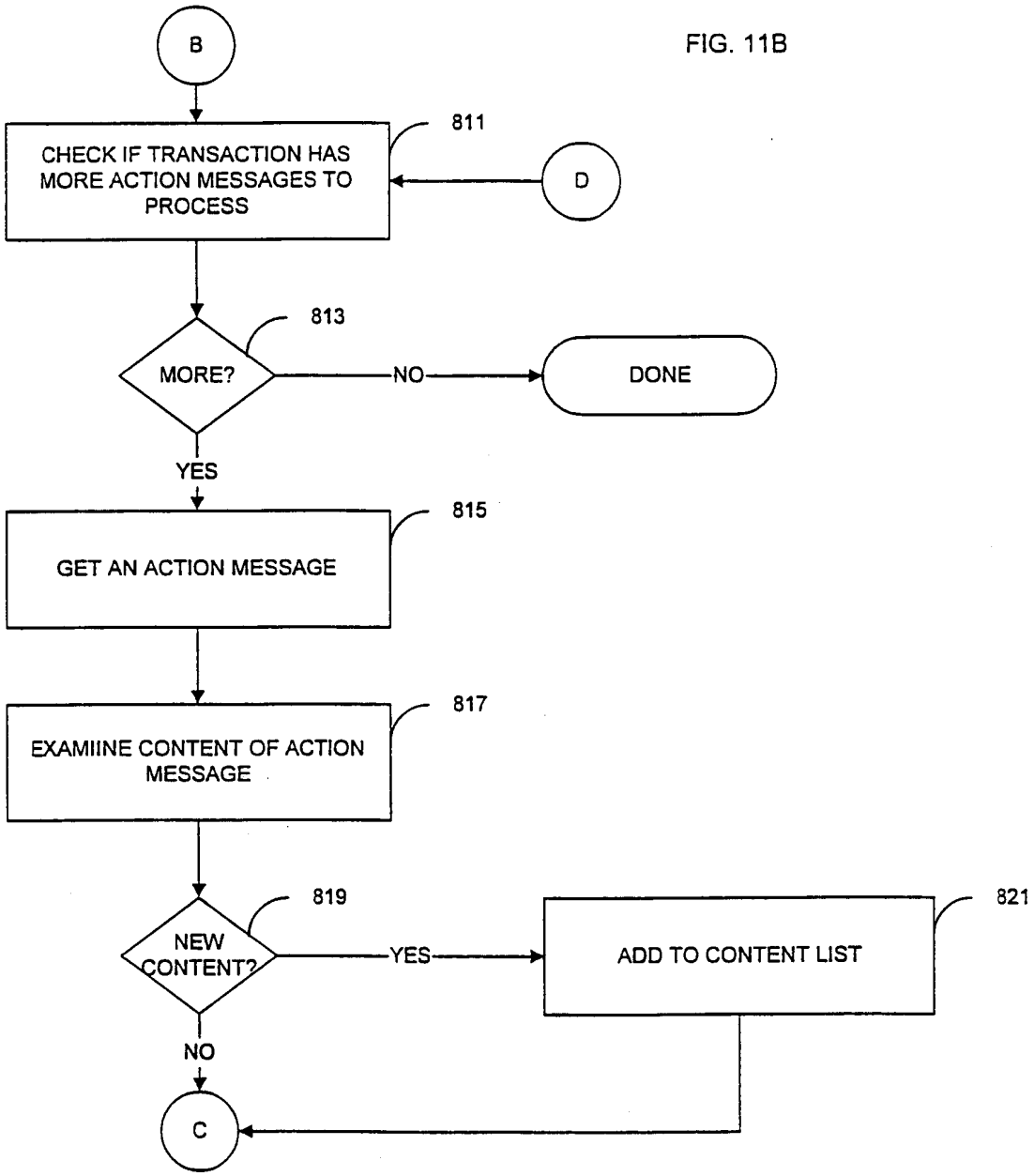

PROCESSING TRANSACTIONS
801
USER GETS ONLINE
803
USER IS ONLINE WITH THE ACTIVITY OF THE APPLICATION
805
USER COMMITS A TRANSACTION
807
COORDINATOR RECEIVES THE TRANSACTION AND ACKNOWLEDGES RECEIPT
809
START PROCESSING TRANSACTION
B

FIG. 11A

FIG. 11B


B
CHECK IF TRANSACTION HAS MORE ACTION MESSAGES TO PROCESS
811
D
MORE?
813
NO
DONE
YES
GET AN ACTION MESSAGE
815
EXAMIINE CONTENT OF ACTION MESSAGE
817
NEW CONTENT?
819
YES
ADD TO CONTENT LIST
821
NO
C

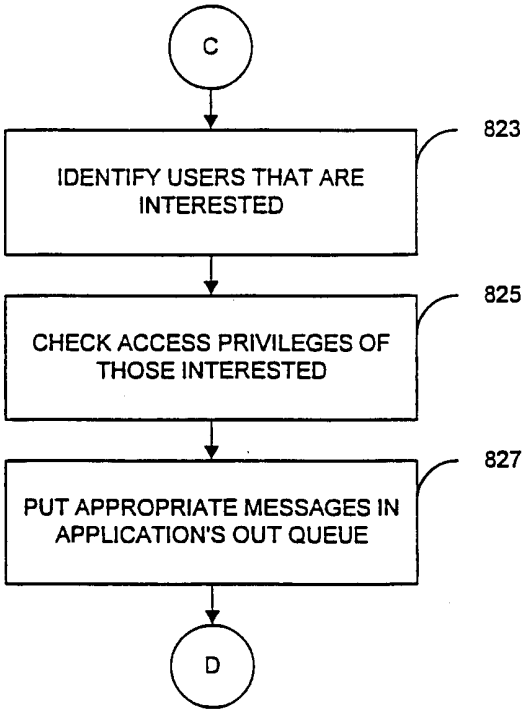

C
IDENTIFY USERS THAT ARE INTERESTED
823
CHECK ACCESS PRIVILEGES OF THOSE INTERESTED
825
PUT APPROPRIATE MESSAGES IN APPLICATION'S OUT QUEUE
827
D

FIG. 11C

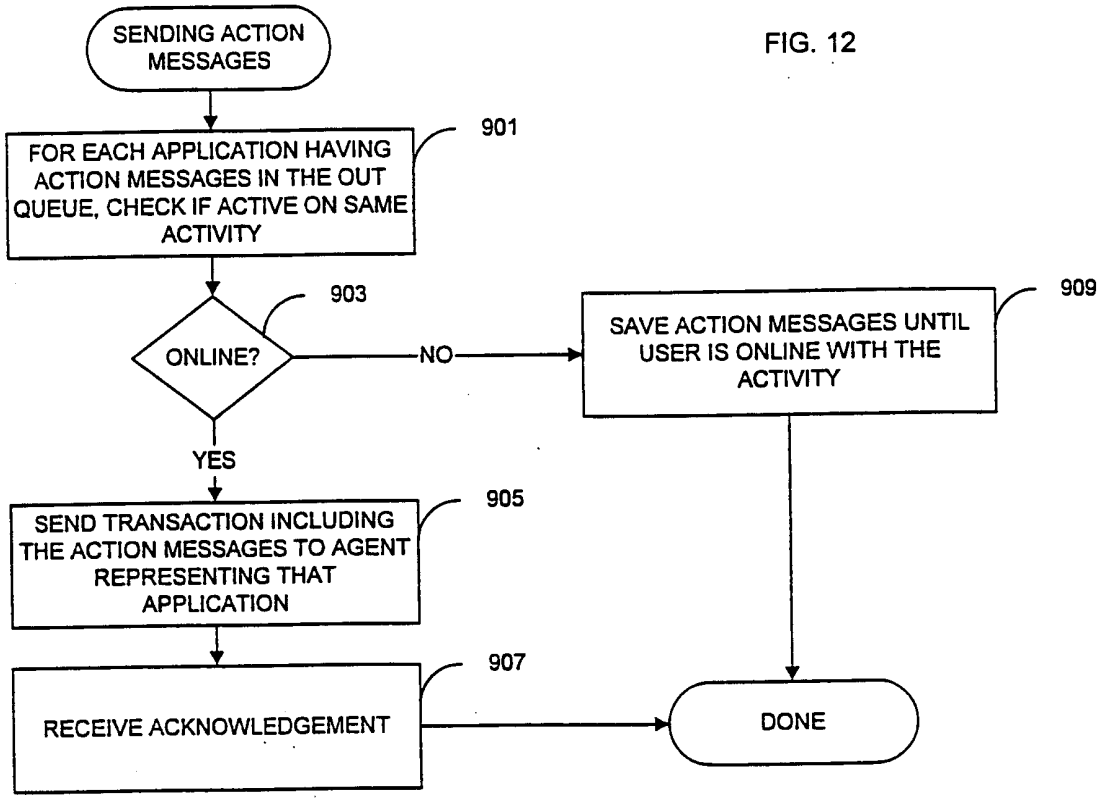

SENDING ACTION MESSAGES
FOR EACH APPLICATION HAVING ACTION MESSAGES IN THE OUT QUEUE, CHECK IF ACTIVE ON SAME ACTIVITY
901
ONLINE?
903
NO
SAVE ACTION MESSAGES UNTIL USER IS ONLINE WITH THE ACTIVITY
909
YES
SEND TRANSACTION INCLUDING THE ACTION MESSAGES TO AGENT REPRESENTING THAT APPLICATION
905
RECEIVE ACKNOWLEDGEMENT
907
DONE

FIG. 12