

[0037] As aforementioned, each GSO also constitutes a persistent, real-time conferencing session between the members of the GSO who are connected to the server. Hence, any modification to the set of fixed properties or the set of variable properties of a GSO is not only stored in the underlying database but also automatically broadcast to the other members of that GSO by sending notifications to the clients currently connected. The receipt of a Notification at the client causes the client to transition from Idle state 40 to Update Object state 44, where the contents of the object data structure are updated, and the state returns to Idle.

[0038] The server process not only sends notifications about modifications to GSOs but also broadcasts the creation of new GSOs and the deletion of existing GSOs. The default behavior could be that any modification to a GSO is broadcast. Alternatively, a subscription object, associated with each client or each client/GSO pair, could be used control the amount of notifications sent out by the server process. With such an arrangement, clients can subscribe or unsubscribe to properties they are interested in hearing about.

[0039] Referring now to FIG. 5, a flow diagram is provided illustrating several exemplary steps that may be taken at the collaboration server of the invention to control broadcast of generic shared objects for conferencing and content management purposes. At step 50, the server receives a request for creating/modification/deletion/addition of a property of a GSO. At step 52, the server determines whether the client is authorized to perform the activity on the GSO. This determination is made by comparing the client against the access control list for the GSO. If the client is unauthorized, then in one embodiment the server merely drops the request at step 51. Alternatively the server could respond with a negative acknowledgement (NACK) or some indication to the client that the request is not granted. If, however, the client is authorized, then at step 54 the server executes the request on the object, and at step 56 forwards a Response package to the requesting client. As mentioned above, the Response package may include a copy of the modified object, or may simply include a message to the client that the modification is accepted. At step 58 the server sends a Notification package to each client on the access control list of the GSO (if connected) to inform the members of the modification.

[0040] In case modifications are propagated to related GSOs, at step 59, the server searches the relationship database at step 60 to identify related GSOs and the corresponding clients that are interested in changes to the modified GSO. At step 62, the server forwards the modifications to the identified interested clients. The server may accumulate the modifications, only sending them out at the desired subscription times of the client. Alternatively, the server may send them to the client at predefined time intervals (so as not to overload the network). Also, the server may send out modifications only to clients that are subscribed to certain types of modifications. In addition, some modifications may be prioritized so that they are immediately forwarded to the clients. Thus, various techniques of controlling transmission of the modifications to the clients may be used, and the present invention is not limited to any particular technique.

[0041] Although the above embodiment has described the object server as functioning on a dedicated device, this is not a requirement of the invention. Rather, the Generic Server

functionality may be implemented on any device, whether it is a client or a server. For example, referring now to FIG. 6, an example is provided of the support of GSOs in a peer-to-peer networked environment. In the peer-to-peer embodiment, each client runs an instance of the collaboration server and the collaboration servers talk to each other to broadcast modifications. For example, two clients, 112 and 212 each include functionality previously described on the GSO server. For example, client 124 includes navigation logic and relation database 124, access control logic 126 and database 118. Where only two client devices are coupled, it may be that the navigation logic and relation database format are simplified, indicating whether or not the partner client has access to the object or not. However, for systems wherein three or more clients communicate peer-to-peer, the logic is substantially similar to that previously described. Each client would have a network interface that integrates the broadcast functionality, described with regard to FIG. 5, with the API of FIG. 4. Thus, whenever one client seeks to modify an object, it notifies the other clients of the modification, and awaits receipt from the other client that the request has been processed at the peer device.

[0042] An example of how Generic Shared Objects (GSOs) may be managed by a GSO server of the present invention to seamlessly provide synchronous, real-time collaboration and asynchronous collaboration between multiple users will now be described with reference to FIG. 7. Bob is a project lead and he works with Dan on a project on "Casual Displays". Catherine is a web designer in their company who is responsible for the external web site. At 81, Bob receives an email from Catherine containing a draft for a project description that she would like to put on their external web site. She wants some feedback from Bob. Before getting back to her, Bob wants to discuss the design of that web page with Dan. Instead of forwarding the message to Dan via email, Bob decides to start a new activity by creating a shared object based on this message. He right-clicks on the original message in his inbox, selects "share", enters Dan's email address, and hits "Share". At 82, a new shared message object (with Bob and Dan as members) shows up in Bob's activity tree in the right window pane (screen A). Bob right-clicks on the shared object and adds a new shared message to the initial one, because he wants to let Dan know that he would like to discuss this with him. At 83, Bob's message shows up as a reply to the initial message similarly to a newsgroup thread.

[0043] A few hours later, Dan returns to his desktop, which is running the client, and notices Bob's newly created shared messages. He opens one message and while he is reading it, at 84 Bob sees that Dan is looking at the messages because the shared object is lit green along with Dan's name underneath the object. Bob takes this as an opportunity to begin a discussion with Dan within the context of the shared object. He right-clicks on the initial message and adds a chat object to this activity at 85. A chat window D pops up on Dan's desktop and they chat. In their chat conversation, Bob and Dan continue talking about the web page over the phone. At some point during the discussion, Bob wants to show directly how to change the web page. He right-clicks on the chat object in his activity tree and adds a shared screen object (86). A transparent window allows Bob to select and "screen scrape" any region on his desktop. He freezes the transparent window over Catherine's draft web page. The screen shot pops up on Dan's desktop. Bob and