

sponding master page except that the conflicting revision is highlighted and displayed in place of the synchronized revision.

Proceeding to block **710**, a conflict indicator is displayed on the master page of the shared object. The conflict indicator may be a drop down menu, a tab, or any other mechanism that informs a user that a conflict page is available for the master page. The conflict indicator for a conflict page associated with a particular user may be distinct from the conflict indicator for conflict pages associated with other users such that a current user may quickly identify the conflict pages generated by the current user.

Advancing to block **720**, the conflict page is displayed alongside the master page when the conflict indicator is selected. The user is presented with both the synchronized state of the master page and the corresponding conflict page.

Transitioning to block **730**, the user reconciles and merges the conflicting revisions into the master page. In one embodiment, the user may select the content container such that the content container is merged with the master page. In another embodiment, the user may directly implement revisions onto the master page. In yet another embodiment, the user may identify conflicting revisions as irrelevant.

Continuing to block **740**, conflicting revisions that are identified as irrelevant are purged. In one embodiment, conflicting revisions may be identified as irrelevant by a user. In another embodiment, conflicting revisions may be automatically identified as irrelevant. For example, a user may have synchronized several revisions with the master version of the shared object located on a server while ignoring any corresponding conflict pages. The older conflict pages that the user did not reconcile are identified as irrelevant after a predetermined time period has elapsed. Processing then terminates at an end block.

FIG. 8 illustrates an operational flow diagram illustrating a process for synchronizing multiple user revisions to a shared object. The process begins at a start block where different versions of shared object are stored in different locations throughout a system. Moving to block **800**, the shared object is downloaded from a store to a client.

Proceeding to decision block **810**, a determination is made whether the shared object is the current version of the shared object. If the shared object is the current version of the shared object, processing terminates at an end block. If the shared object is not the current version of the shared object, processing continues at block **820**. The shared object may not be the current version because the most recent revision to a content container of the shared object is not available from the store.

Advancing to block **820**, a request tag and client information are assigned to the store to indicate that the client requires the most recent revision data to update the shared object. The client information may include a GUID that identifies the requesting client and a time stamp that identifies the time when the client requested the current version of the shared object from the store.

Transitioning to block **830**, the current version of the shared object is received at the store. The store may receive the current version of the shared object when another client accesses the store with the most recent revision data. The requesting client is informed that the current version of the shared object has been received by the store. Continuing to block **840**, the current version of the shared object is synchronized with the requesting client. Processing then terminates at the end block.

FIG. 9 illustrates an operational flow diagram illustrating a process for seamlessly transitioning from asynchronous to synchronous communication modes. The process begins at a

start block where a peer group is established that identifies users who are authorized to access a shared object.

Moving to block **900**, a client accesses the shared object on a server. The client is automatically connected to other clients that are also accessing the shared object (i.e., the peer group). The shared object is associated with a manifest file. The shared object includes a unique location identifier that identifies the location where the corresponding manifest file is stored in the system.

Proceeding to block **910**, the manifest file is retrieved from the location identified by the unique location identifier. The manifest file identifies the locations where other versions and instances of the shared object are stored within the system. The manifest file includes a peer group identifier for the peer group where a version of the shared object is stored.

Advancing to block **920**, a peer-to-peer network is established when any other client in the peer group accesses a version or instance of the shared object identified by the manifest file. Thus, the client may disconnect from the server and continue to access the shared file on the peer-to-peer network. Processing then terminates at an end block.

FIG. 10 illustrates an operational flow diagram illustrating a process for seamlessly transitioning from synchronous to asynchronous communication modes. The process begins at a start block where a peer-to-peer network is established between at least two users who are authorized to access a shared object.

Moving to block **1000**, a client accesses the shared object on the peer-to-peer network. The shared object is associated with a manifest file. The shared object includes a unique location identifier that identifies the location where the corresponding manifest file is stored in the system.

Proceeding to block **1010**, the manifest file associated with the shared object is retrieved from the location identified by the unique location identifier. The manifest file identifies the locations where other versions and instances of the shared object are stored within the system. Advancing to block **1020**, the client connects to a server. The client determines which other clients are also connected to the server. Transitioning to block **1030**, the client identifies other clients that are authorized to access the shared object from the peer-to-peer network. Continuing to block **1040**, the client connects to an authorized client when the peer-to-peer network is unavailable. Processing then terminates at an end block.

The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

What is claimed is:

1. A computer-implemented method for synchronizing multiple user revisions to a shared object, comprising:
  - accessing a shared object on a server to form an asynchronous server communication mode between a client device and the server;
  - determining a location of a manifest file for the shared object based on a unique location identifier in the accessed shared object;
  - obtaining the manifest file on the client device based on the unique location identifier in the accessed shared object, wherein the manifest file includes a network location identifier for concurrently accessed versions of the shared object in a synchronous peer communication mode that provides real-time communication;
  - based on the obtained manifest file, automatically and seamlessly transitioning from the asynchronous server