

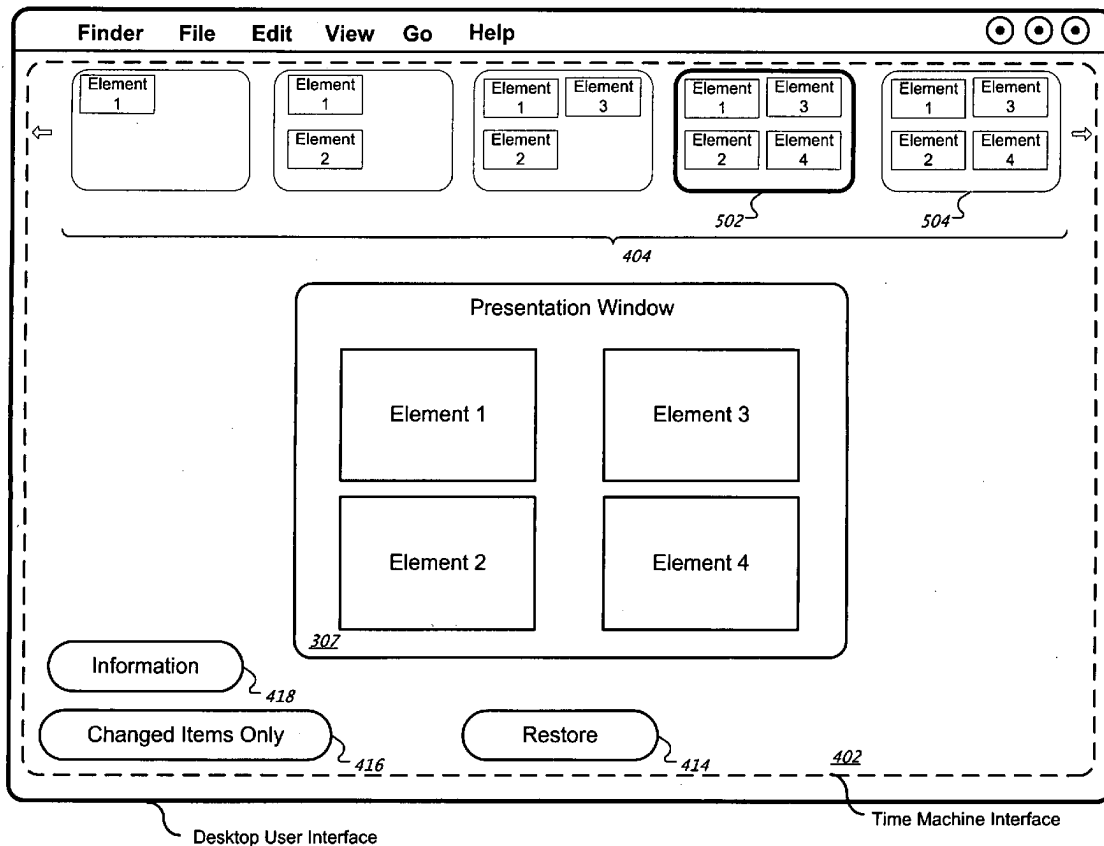


US 20080059894A1

(19) **United States**(12) **Patent Application Publication**
Cisler et al.(10) **Pub. No.: US 2008/0059894 A1**(43) **Pub. Date: Mar. 6, 2008**(54) **CONFLICT RESOLUTION IN RECOVERY
OF ELECTRONIC DATA**(52) **U.S. Cl. 715/762**(76) **Inventors:** **Pavel Cisler**, Los Gatos, CA (US);
Gene Zyril Ragan, Santa Clara,
CA (US); **Dave Lyons**, San Jose,
CA (US)(57) **ABSTRACT**

Systems and methods are provided for resolving conflicts. In one implementation, a method relating to modifying a view in a user interface is provided. The method includes receiving, while a current view is displayed in a user interface, a first user input requesting that a history view associated with the current view be displayed. The history view is displayed in response to the first user input, the history view includes at least a first visual representation of an earlier version of the current view, and the earlier version includes a first element. There is received, while the history view is displayed, a second user input requesting that the current view be modified according to the earlier version, at least with regard to the first element. The method further includes determining, in response to the second user input, whether the current view includes a current version of the first element.

Correspondence Address:
FISH & RICHARDSON P.C.
PO BOX 1022
MINNEAPOLIS, MN 55440-1022

(21) **Appl. No.: 11/499,885**(22) **Filed: Aug. 4, 2006****Publication Classification**(51) **Int. Cl.**
G06F 3/00 (2006.01)

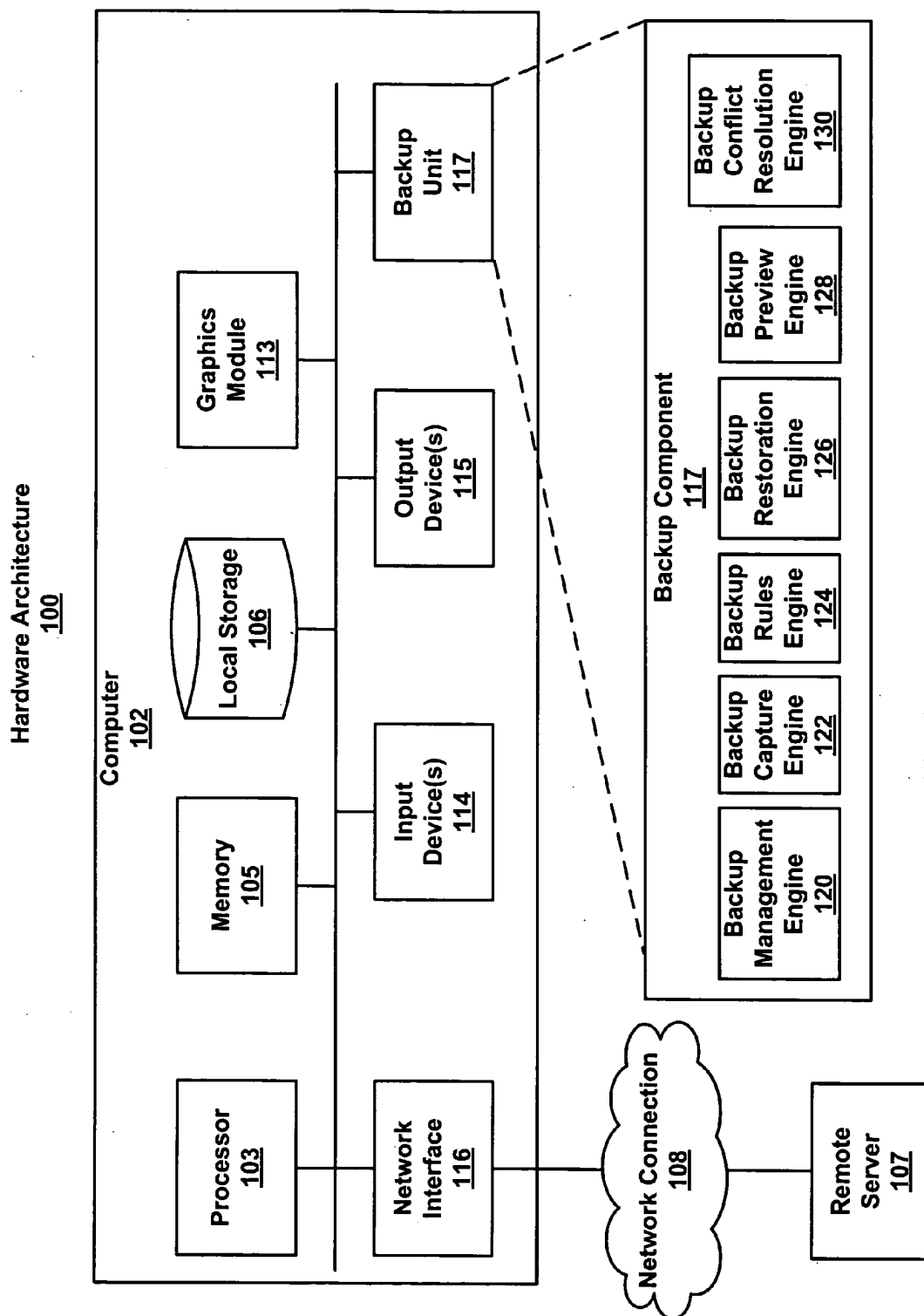
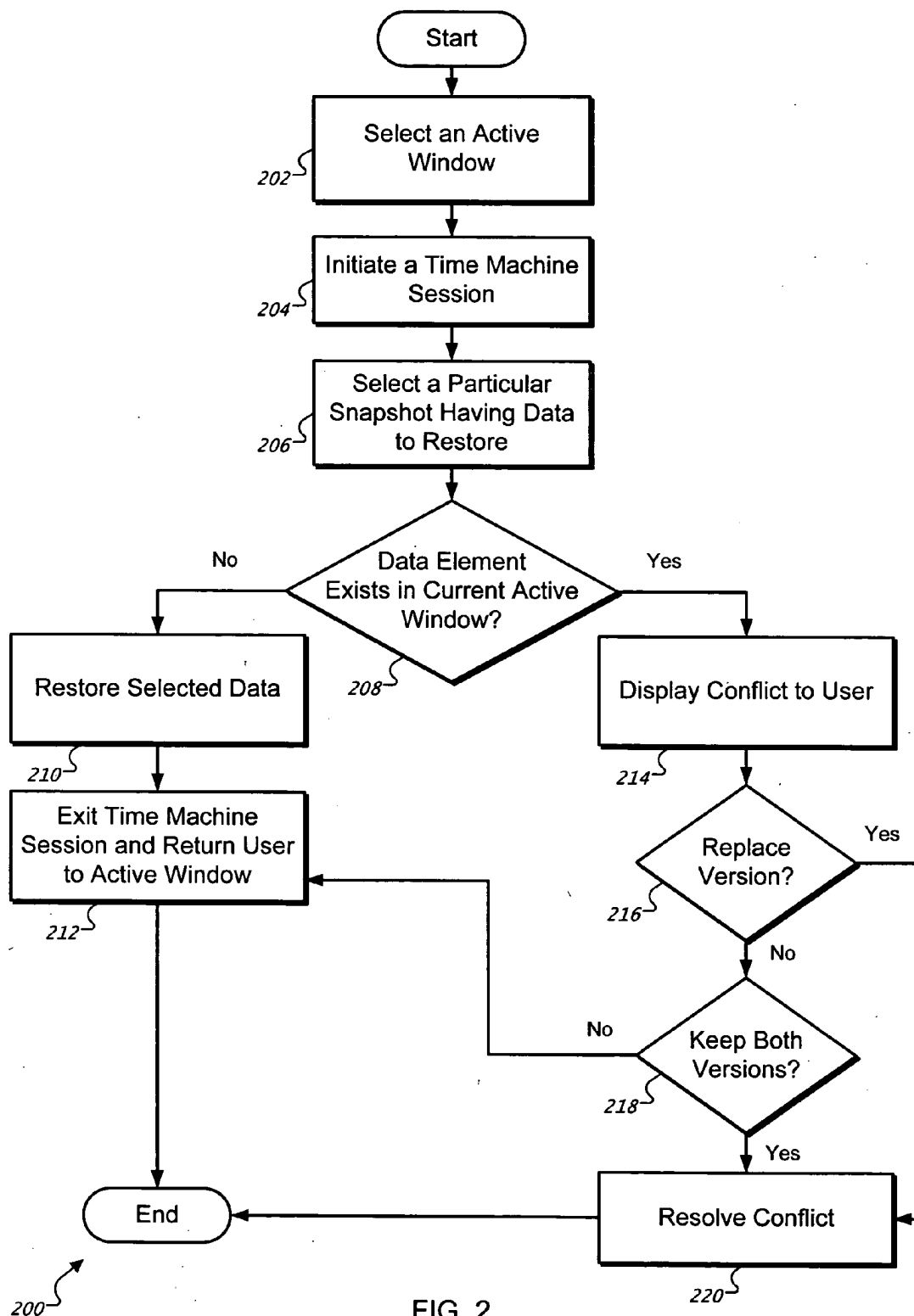


Fig. 1



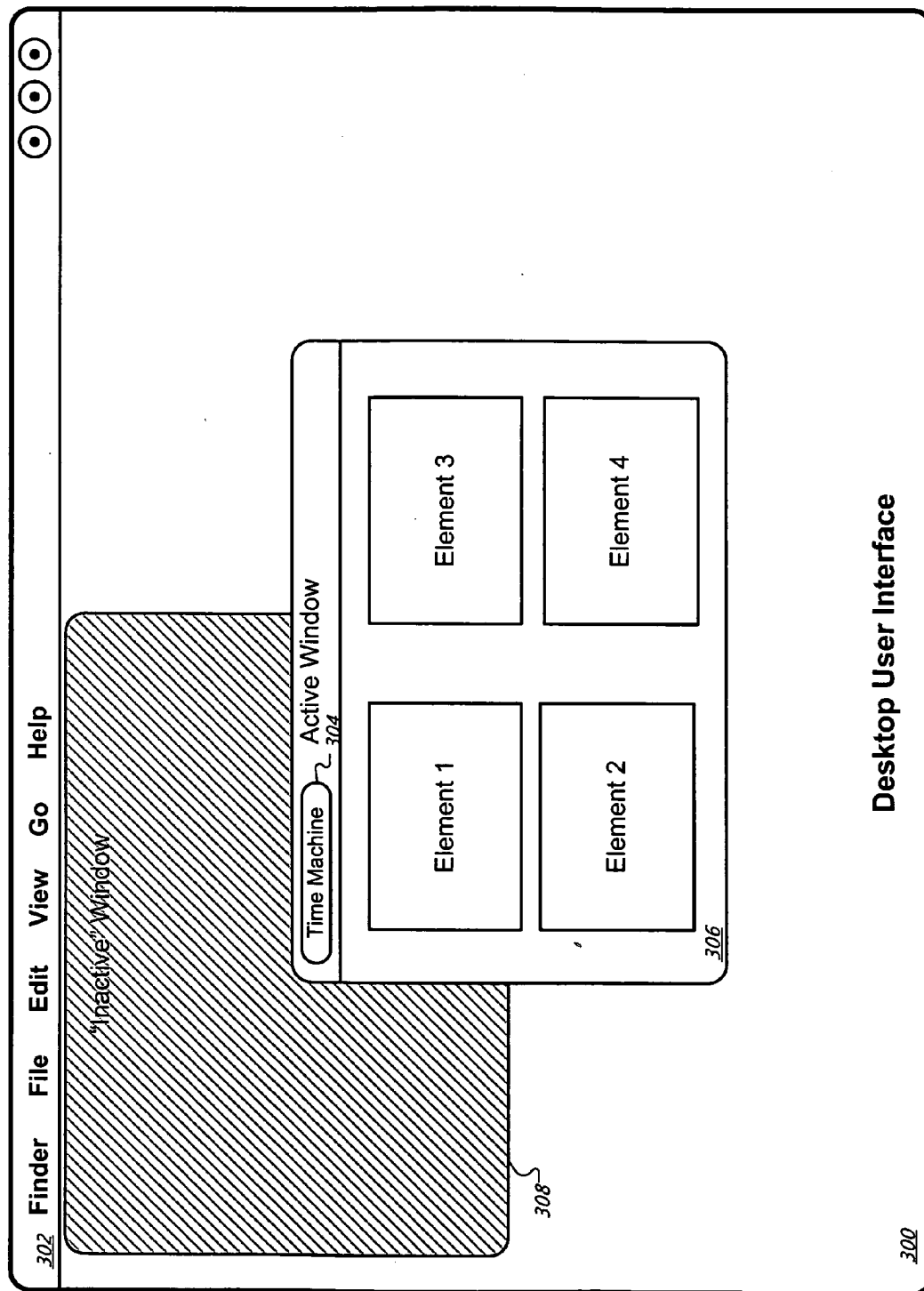


FIG. 3

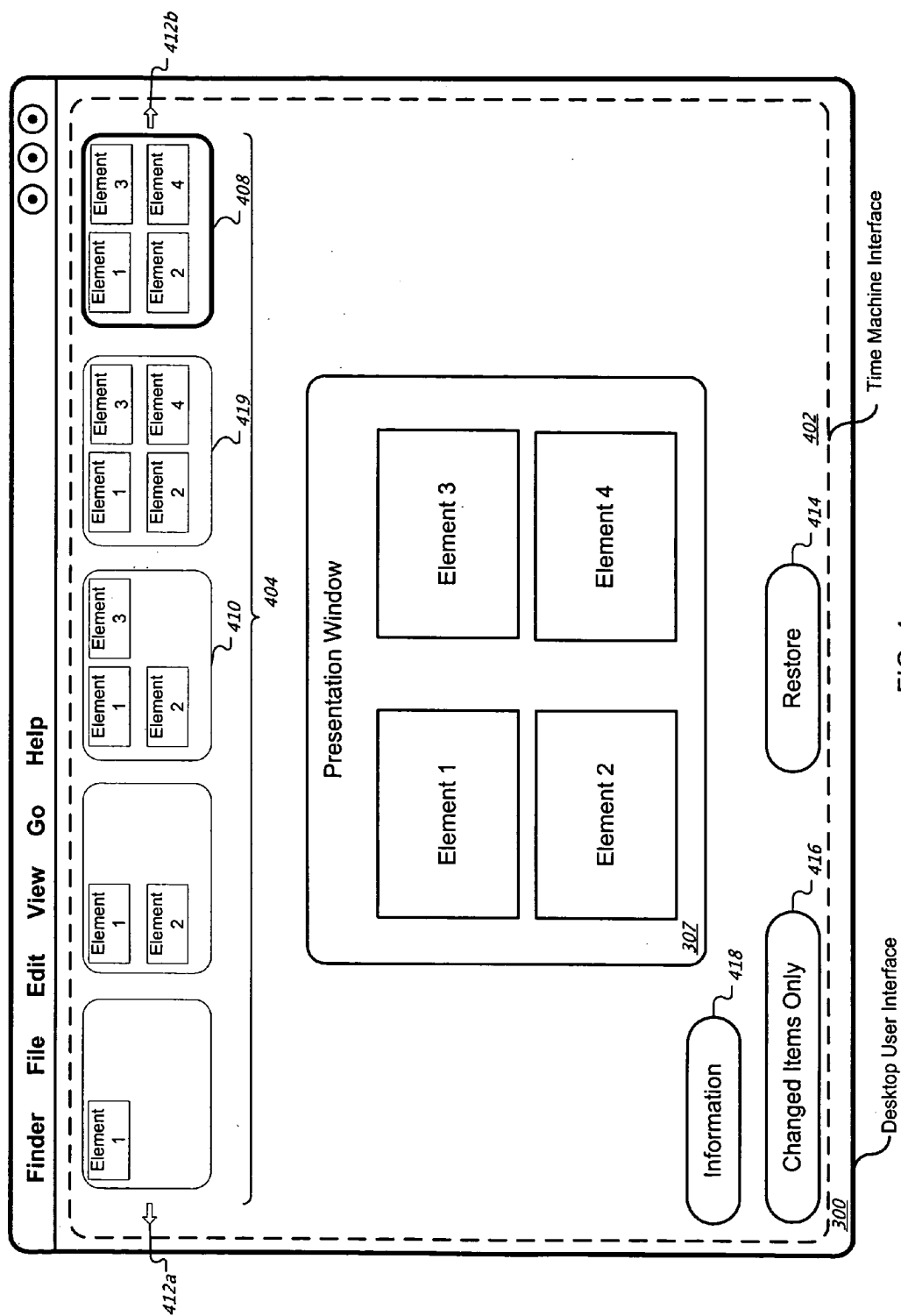


FIG. 4

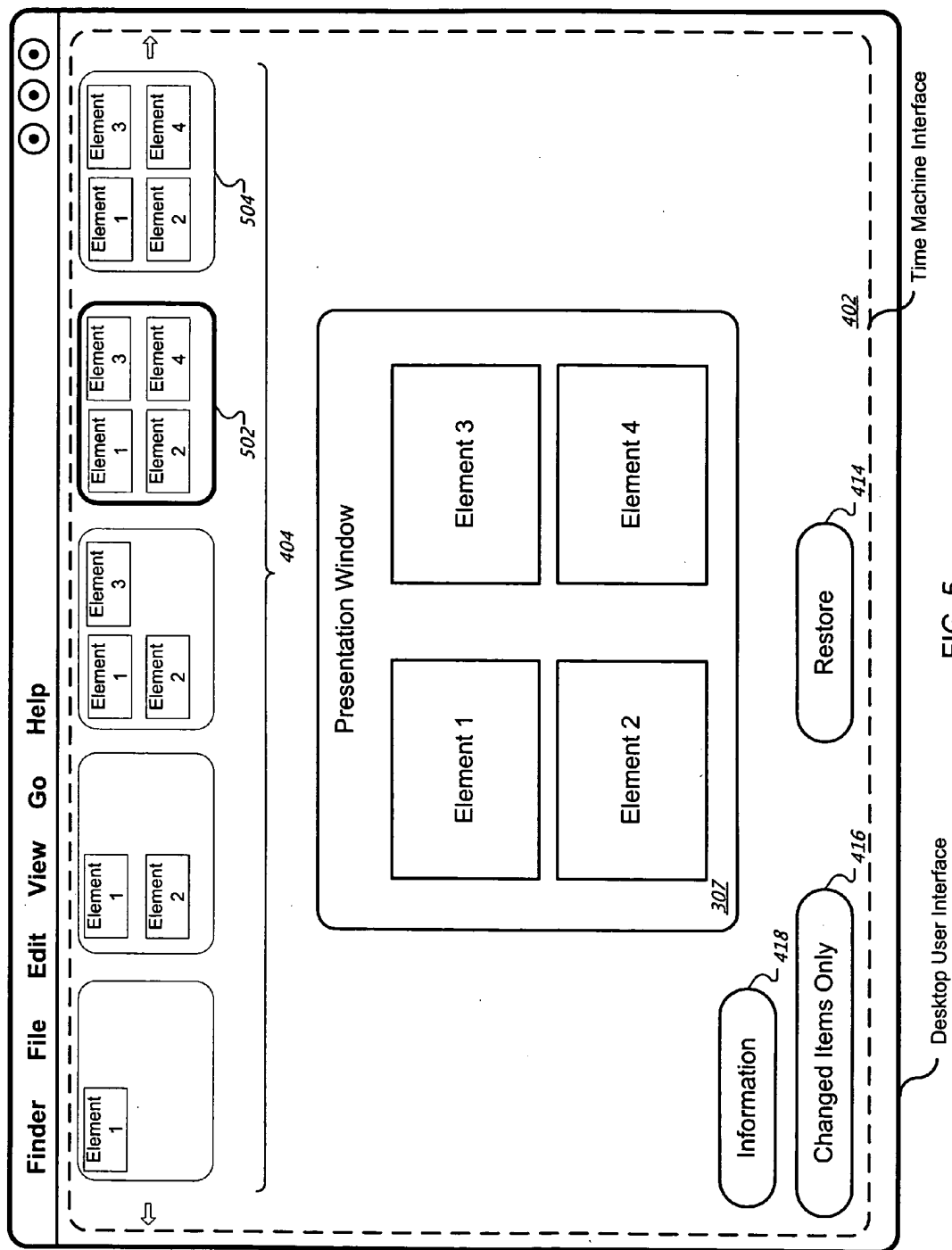


FIG. 5

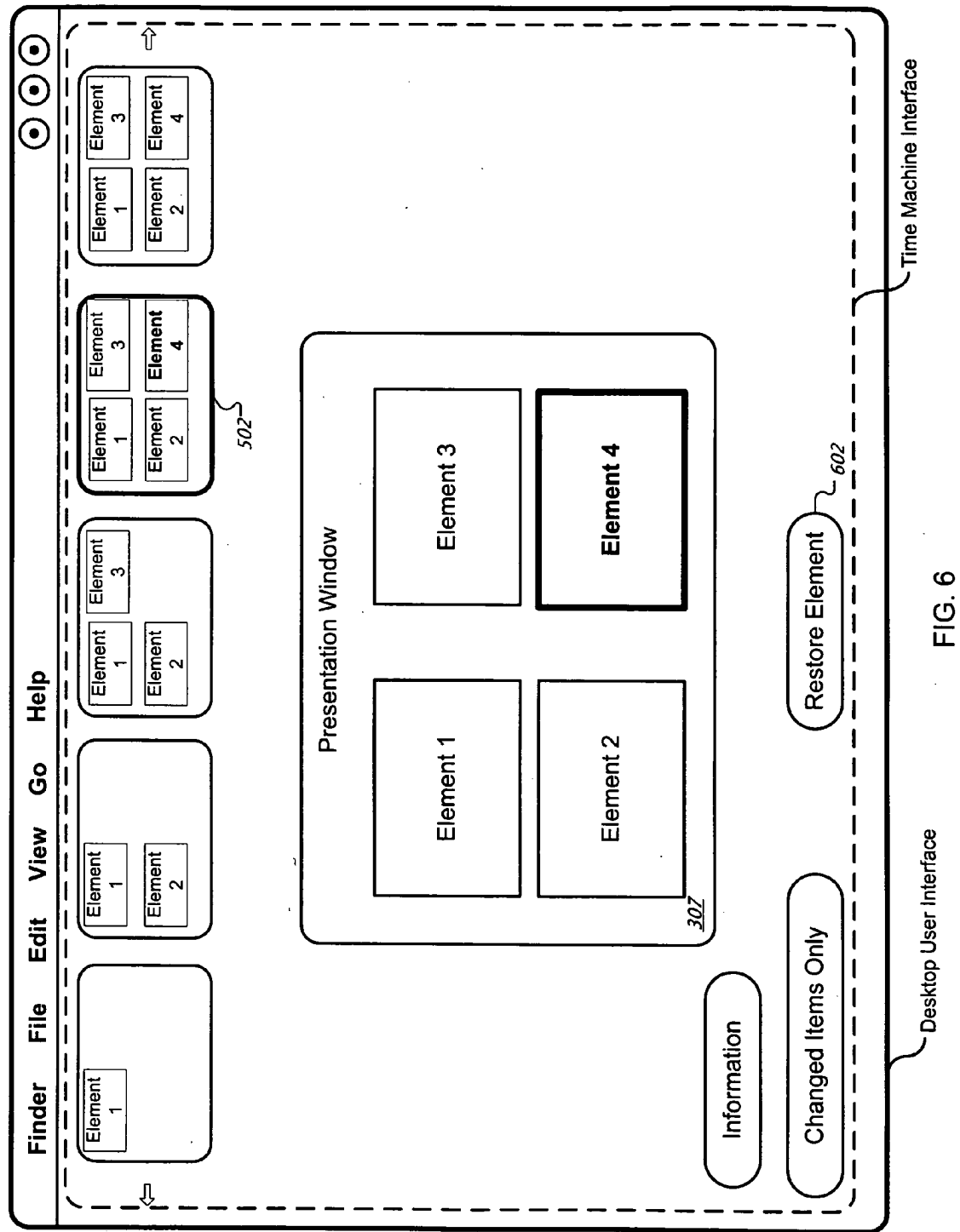


FIG. 6

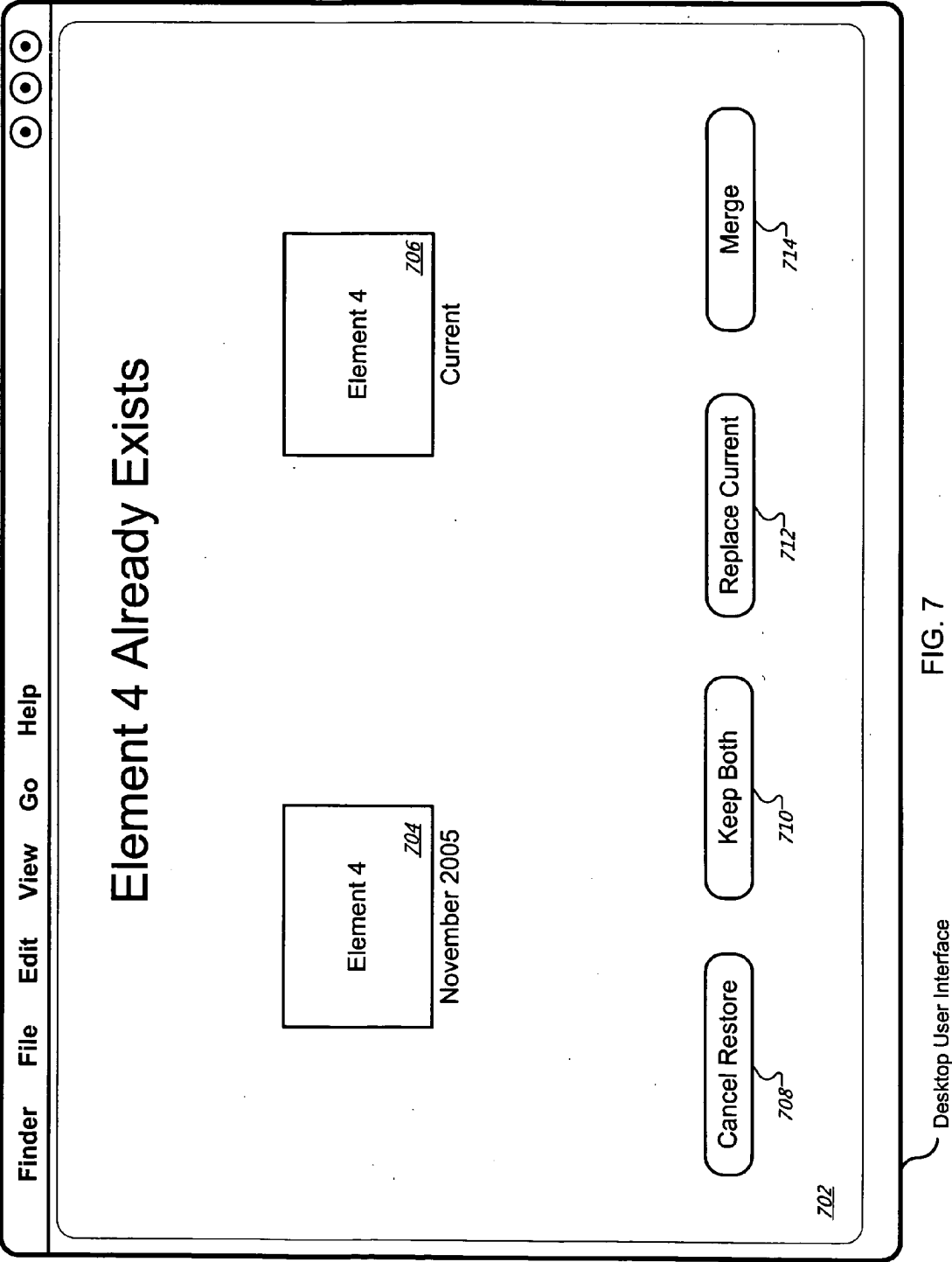


FIG. 7

CONFLICT RESOLUTION IN RECOVERY OF ELECTRONIC DATA

RELATED APPLICATIONS

[0001] This application is generally related to the following jointly owned and co-pending patent applications, each incorporated herein by reference in its entirety:

- [0002] U.S. patent application Ser. No. _____, for "Managing Backup of Content," filed Aug. 4, 2006;
- [0003] U.S. patent application Ser. No. _____, for "User Interface for Backup Management," filed Aug. 4, 2006;
- [0004] U.S. patent application Ser. No. _____, for "Navigation of Electronic Backups," filed Aug. 4, 2006;
- [0005] U.S. patent application Ser. No. _____, for "Architecture for Back Up and/or Recovery of Electronic Data," filed Aug. 4, 2006;
- [0006] U.S. patent application Ser. No. _____, for "Searching a Backup Archive," filed Aug. 4, 2006;
- [0007] U.S. patent application Ser. No. _____, for "Application-Based Backup-Restore of Electronic Information," filed Aug. 4, 2006;
- [0008] U.S. patent application Ser. No. _____, for "System for Multi-Device Electronic Backup," filed Aug. 4, 2006;
- [0009] U.S. patent application Ser. No. _____, for "System for Electronic Backup," filed Aug. 4, 2006;
- [0010] U.S. patent application Ser. No. _____, for "Restoring Electronic Information," filed Aug. 4, 2006;
- [0011] U.S. patent application Ser. No. _____, for "Links to a Common Item in a Data Structure," filed Aug. 4, 2006;
- [0012] U.S. patent application Ser. No. _____, for "Event Notification Management," filed Aug. 4, 2006;
- [0013] U.S. patent application Ser. No. _____, for "Consistent Back Up of Electronic Information," filed Aug. 4, 2006.

TECHNICAL FIELD

[0014] The disclosed implementations relate generally to storing and restoring data.

BACKGROUND

[0015] A hallmark of modern graphical user interfaces is that they allow a large number of graphical objects or items to be displayed on a display screen at the same time. Leading personal computer operating systems, such as Apple Mac OS®, provide user interfaces in which a number of windows can be displayed, overlapped, resized, moved, configured, and reformatted according to the needs of the user or application. Taskbars, menus, virtual buttons and other user interface elements provide mechanisms for accessing and activating windows even when they are hidden behind other windows.

[0016] With the sophisticated tools available, users are encouraged not only to create and save a multitude of items in their computers, but to revise or otherwise improve on them over time. For example, a user can work with a certain file and thereafter save its current version on a storage device. The next day, however, the user may have had second thoughts about the revisions, or may have come up with new ideas, and therefore open the file again.

[0017] The revision process is usually straightforward if the user wants to add more material to the file or make

changes to what is there. But it is typically more difficult for a user who has changed his/her mind about changes that were previously made and wants the file back as it was once before. Application programs for word processing typically let the user "undo" previous edits of a text, at least up to a predefined number of past revisions. The undo feature also usually is configured so that the previously made revisions must be undone in reverse chronological order; that is, the user must first undo the most recently made edit, then the second-most recent one, and so on. If the user saves and closes the document and thereafter opens it again, it may not be possible to automatically undo any previous edits.

SUMMARY

[0018] Systems and methods are provided that relate to a conflict in restoring an interface view including processes and interfaces for resolving conflicts between an element to restore and a currently existing element. Specifically, when a user attempts to restore an element from a snapshot that has a current version as well, a conflict is generated. The user is alerted to the conflict and can choose, for example, to retain the current element, retain the snapshot element, or retain both elements.

[0019] In general, in one aspect, a method relating to modifying a view in a user interface is provided. The method includes receiving, while a current view is displayed in a user interface, a first user input requesting that a history view associated with the current view be displayed. The history view is displayed in response to the first user input, the history view includes at least a first visual representation of an earlier version of the current view, and the earlier version includes a first element. There is received, while the history view is displayed, a second user input requesting that the current view be modified according to the earlier version, at least with regard to the first element. The method further includes determining, in response to the second user input, whether the current view includes a current version of the first element.

[0020] Implementations of the method can include one or more of the following features. The determination can indicate that the current view does not include a current version of the first element, and the method can further include modifying, in response to the determination, the current view to include at least the first element. The determination can indicate that the current view does include a current version of the first element, and the method can further include displaying a conflict resolution view in response to the determination.

[0021] The conflict resolution view can provide user-selectable conflict resolution by at least one option of canceling the request that the current view be modified, replacing the current version of the first element with the first element included in the earlier version, and modifying the current view to include both the current version of the first element and the first element included in the earlier version. The user-selectable conflict resolution can provide the option of merging, in the current view, the current version of the first element and the first element included in the earlier version. The user-selectable conflict resolution can include a rollover effect graphically indicating a result of selecting each of the options. For the option of canceling the request that the current view be modified, the rollover effect can indicate that the current version of the first element will be kept. For the option of replacing the current version of the

first element with the first element included in the earlier version, the rollover effect can indicate that the current version of the first element will be discarded. For the option of modifying the current view to include both the current version of the first element and the first element included in the earlier version, the rollover effect can indicate that both the current version of the first element and the first element included in the earlier version will be kept.

[0022] The conflict resolution view can contain the current version of the first element and the first element included in the earlier version. The first element can be one selected from the group including a folder, a file, an item, an information portion, a playlist, a directory, an image, system parameters, and combinations thereof.

[0023] In general, in one aspect, a computer program product is provided. The computer program product generates on a display device a graphical user interface. The graphical user interface includes a view display area for presenting a current view and a history view display area for presenting a history view associated with the current view. The history view includes at least a first visual representation of an earlier version of the current view and providing that a user can initiate a restoration of the current view with at least a portion of the earlier version. The graphical user interface also includes a conflict resolution function for alerting the user to a conflict in performing the restoration.

[0024] Implementations can include one or more of the following features. The conflict resolution function can include at least one option of an input control to cancel restoration, an input control to keep both conflicting elements, an input control to perform the restoration, and an input control to merge both conflicting elements. A rollover effect can be provided that graphically indicates a result of selecting any of the options. The first visual representation can be included in a timeline presented in the history view, the timeline can include several visual representations of earlier versions of the current view. The history view can further include an input control for modifying the timeline to include only at least one of the visual representations whose corresponding earlier version differs from the current view. The conflict resolution function can be included in a conflict resolution view that presents visual representations of respective elements in the current version and the earlier version that give rise to the conflict.

[0025] In general, in one aspect, a method is provided. The method includes defining a criterion for capturing a state of a view of a user interface. The state of the view is captured in accordance with the criterion. A prompt is received to suspend presentation of a current view and present a captured view. A determination is made as to if a conflict exists between the captured view and the current view. If a conflict does not exist, the captured view is reinstated into the current view of the user interface.

[0026] Implementations can include one or more of the following features. If a conflict is detected, the conflict can be resolved prior to reinstating the captured view. The conflict can be deemed to exist if the determination indicates that information present in the current view would be lost if the captured view were reinstated into the current view.

[0027] In general, in one aspect, a method is provided. A current view of a user interface is evaluated. A determination is made that an undesirable change to the current view of the user interface has occurred relative to a past view of the user interface. The current view of the user interface is sus-

pended. One or more past view of the user interface are presented. A determination is made as to if a conflict exists between the captured view and the current view. If there is no conflict, a past view of the user interface is reinstated into the current view.

[0028] In general, in one aspect, a method is provided. The method includes presenting an element in a user interface, the element having a current state and being presented in a current view of a user interface. A state of the element is captured over time. A current view of the user interface including the element is suspended. A past view of the element is reinstated including resolving a conflict between the current view and past view with respect to the element.

[0029] Particular embodiments of the subject matter described in this specification can be implemented to realize one or more of the following advantages. A user can resolve conflicts when restoring data from a snapshot. Data that is present in a current state and in a snapshot can be restored without loss of data. The user can determine which version of the data to maintain as a current version.

[0030] The details of the various aspects of the subject matter described in this specification are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the invention will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0031] FIG. 1 is a block diagram of a hardware architecture for previewing, installing, managing and operating widgets in a display environment.

[0032] FIG. 2 is a flow diagram of a method illustrating a conflict scenario.

[0033] FIG. 3 is a screen shot depicting a desktop user interface prior to initiation of a time machine session.

[0034] FIG. 4 is a screen shot depicting an initial state for a time machine user interface.

[0035] FIG. 5 is a screen shot depicting a time machine user interface where a user has selected a particular snapshot.

[0036] FIG. 6 is a screen shot depicting a time machine user interface after a snapshot has been selected for restore.

[0037] FIG. 7 is a screen shot depicting a conflict management screen.

DETAILED DESCRIPTION

[0038] FIG. 1 is a block diagram of a hardware architecture 100 for modifying a user interface view. As used herein, a view refers to an item, element, or other content, capable of being stored and/or retrieved in an interface, that can be subjected to a back up operation by a backup component 117. For example, a user interface view can contain any number of icons, files, folders, application state information and/or machine state information, etc. The architecture 100 includes a personal computer 102 communicatively coupled to a remote server 107 via a network interface 116 and a network 108 (e.g., local area network, wireless network, Internet, intranet, etc.). The computer 102 generally includes a processor 103, memory 105, one or more input devices 114 (e.g., keyboard, mouse, etc.) and one or more output devices 115 (e.g., a display device). A user interacts with the architecture 100 via the input and output devices 114, 115.

[0039] The computer 102 also includes a local storage device 106 and a graphics module 113 (e.g., graphics card) for storing information and generating graphical objects, respectively. The local storage device 106 can be a computer-readable medium. The term “computer-readable medium” refers to any medium that includes data and/or participates in providing instructions to a processor for execution, including without limitation, non-volatile media (e.g., optical or magnetic disks), volatile media (e.g., memory) and transmission media. Transmission media includes, without limitation, coaxial cables, copper wire, fiber optics, and computer buses. Transmission media can also take the form of acoustic, light or radio frequency waves.

[0040] While modifications of a user interface view are described herein with respect to a personal computer 102, it should be apparent that the disclosed implementations can be incorporated in, or integrated with, any electronic device that has a user interface, including without limitation, portable and desktop computers, servers, electronics, media players, game devices, mobile phones, email devices, personal digital assistants (PDAs), embedded devices, televisions, other consumer electronic devices, etc.

[0041] The computer 102 includes a backup component 117 that allows for the storage of versions of the computer's files to be saved, either within the local storage 106 or in an external storage repository (e.g., a backup repository). A backup management engine 120 coordinates the collection, storage, and retrieval of file versions performed by the backup component 117. A backup capture engine 122 locates changed files which are in need of a backup operation. The backup capture engine 122 then stores copies of these files in a targeted storage repository. The backup capture engine 122 can track multiple version copies of each file included in the backup repository.

[0042] A backup rules engine 124 contains settings for the details of capturing and storing the earlier versions. For example, the backup rules engine 124 may determine the frequency of the backup capture, the storage location for the backup versions, the types of files that are eligible for backup capture, and the events that trigger a backup capture (periodic or event-driven, etc.). A user can invoke a backup restoration engine 126 to restore previous versions of files that were captured by the backup capture engine 122. A backup preview engine 128 is configured to provide a preview of at least one of the captured earlier versions. A backup conflict resolution engine 130 may be configured to resolve a conflict that is presented for a particular restoration. For example, the backup conflict resolution engine 130 can detect that the restoration of a particular item to a previous version thereof would result in some portion of information being lost from the current version of the item. In some implementations, the conflict resolution engine 130 can detect this condition, alert the user about it, present one or more possible actions, and receive the user's selection of how to proceed, to name a few examples.

[0043] Systems and methods are provided for identifying and addressing conflicts in modifying a user interface view. The systems and methods can be stand alone, or otherwise integrated into a more comprehensive application. In the materials presented below, an integrated system and method for modifying a user interface view is disclosed. However, one of ordinary skill in the art will recognize that the engines, methods, processes and the like that are described

can themselves be an individual process or application, part of an operating system, a plug-in or the like. In one implementation, the system and methods can be implemented as one or more plug-ins that are installed and run on the personal computer 102. The plug-ins are configured to interact with an operating system (e.g., MAC OS® X, WINDOWS XP, LINUX, etc.) and to perform the various functions, as described with respect to the Figures. A system and method for modifying a user interface view can also be implemented as one or more software applications running on the computer 102. Such a system and method can be characterized as a framework or model that can be implemented on various platforms and/or networks (e.g., client/server networks, wireless networks, stand-alone computers, portable electronic devices, mobile phones, etc.), and/or embedded or bundled with one or more software applications (e.g., email, media player, browser, etc.)

[0044] FIG. 2 is a flow diagram of a method 200 illustrating a conflict scenario. The method 200 can be performed in connection with presenting a user with a list of options in a “time machine” user interface that allows the user to selectively restore a currently displayed window, or part thereof, a state of an application or state of the system, preferences (e.g., user or system preferences), and the like to an earlier state that is presented by the time machine. Particularly, the method can resolve a conflict that arises when the user seeks to restore. For example, the user may restore all or part of any element that includes one or more folders, files, items, information portions, directories, images, system parameters, playlists, and the like. The time machine user interface may contain several snapshots in time displaying a visual representation of elements. In one implementation, a snapshot refers to a backup element stored in a historical archive that includes a backup of selected items or content as specified by the backup component 117. A conflict may occur when a user selects a particular snapshot for restoration that differs from a current version of an element. For example, a conflict can occur based on a version difference such as a difference in the file creation date, modified data content, or a file location change. In other words, a conflict may be deemed to exist when the restoration of the snapshot into the current view would cause information present in the current view to be lost. In contrast, if restoring the snapshot would merely add information to that already present in the current view, in any way, this may be deemed as not presenting a conflict.

[0045] The method 200 begins when a user selects 202 an active window in a user interface, and initiates 204 a time machine session for the selected window. For example, the user can initiate the backup management engine 120 to initiate the time machine session.

[0046] Initiating 204 a time machine session can, in one implementation, create a timeline display automatically within the time machine session. Alternatively, the time machine timeline can be displayed when a prompt is received from the user (e.g., a mouseover). The timeline display includes graphic descriptions of previous versions of the active window. In one implementation, the previous versions included in the timeline display may have been retrieved according to a schedule or according to a set of rules (e.g., events which trigger a snapshot). The criteria can be preexisting or can be user-defined. For example, snapshots can be taken simply as a function of time (e.g., hourly, daily, or some other time period). Additionally, in one

implementation, the criteria for taking a snapshot can be programmatically defined or dynamically generated.

[0047] Alternatively, in one implementation, the snapshots can be taken when a triggering event occurs. For example, adding, deleting, or modifying an element may trigger the snapshot to be taken. However, some modifications may not be significant enough to trigger a snapshot. For example, changing the name of a file may not be a significant change and may be disregarded when creating a snapshot. In one implementation, a snapshot can be triggered, however, the snapshot is generated without all of the changed data according to some criteria. In another example, for a music database application such as iTunes, there are file changes indicating play counts for particular songs that may be unimportant to the actual content in the file, and therefore disregarded when creating a snapshot. In one implementation, the backup capture engine 122 captures the snapshot(s) at one or more times specified by the backup rules engine 124.

[0048] When an active window has been selected 202, and a time machine session has been initiated 204, the user may select 206 a particular snapshot having data to restore. In some implementations, the user may select a portion of the earlier version represented by the snapshot for restoration, as opposed to restoring the entire earlier version. Upon the selection, the method 200 verifies 208 whether or not at least one element to be restored exists in the current active window. In one implementation, this step is performed by the backup conflict resolution engine 130. If one or more elements to be restored do not exist in the current active window, the selected data is restored 210, the time machine session is exited, and the user is returned 212 to the active window. This provides, for example, that the user can employ the time machine to go back and restore one or more pieces of information that were part of the earlier version, but have since been removed or deleted.

[0049] If, in contrast, an element to be restored exists in the current active window, the conflict is displayed to the user 214. In one implementation, this step is performed by the backup conflict resolution engine 130. It will be understood that “exists” here includes also the scenario where a different, albeit recognizable, version of the element exists in the current version. The conflict is shown with an option to replace 216 a version of the element. If the user chooses to replace the current version of the element, the conflict is resolved 220. If the user chooses not to replace the current version, an option to keep 218 both versions is presented to the user. Keeping both versions may include providing the current window with both of them, or merging their contents into a new element, for example. If the user chooses to keep both versions, the conflict is resolved 220. In one implementation, the user-selected option is then performed by the backup restoration engine 126. Otherwise, the method cancels the restore operation, exits the time machine session, and returns 212 the user to the active window. In an alternative implementation, a set of rules can be applied to resolve the conflict.

[0050] A preview of the snapshot can be provided, for example so that the user can review the contents and decide whether to make the restoration. In some implementations, the preview is provided by the backup preview engine 128.

[0051] Accordingly, there may be performed a method or process that allows the user to restore a previously captured user interface state. Such a method may include defining a

criterion for capturing a state of a view of a user interface, capturing the state of the view in accordance with the criterion, receiving a prompt to suspend presentation of a current view and present a captured view, determining if a conflict exists between the captured view and the current view, and if not, reinstating the captured view into the current view of the user interface.

[0052] As another example, a method may include evaluating a current view of a user interface, determining that an undesirable change to the current view of the user interface has occurred relative to a past view of the user interface, suspending the current view of the user interface, presenting one or more past views of the user interface, determining if a conflict exists between the captured view and the current view, and if not, reinstating a past view of the user interface into the current view.

[0053] In another example, a method includes presenting an element in a user interface, the element having a current state and being presented in a current view of a user interface, capturing over time, a state of the element, suspending a current view of the user interface including the element, reinstating a past view of the element including resolving a conflict between the current view and past view with respect to the element.

[0054] FIG. 3 is a screen shot depicting one example of a user interface 300 (e.g., a desktop user interface) prior to initiation of a time machine session. The user interface 300 (also referred to herein as “desktop”) is a user interface as may be provided by an operating system. The user interface 300 has a background, a menu bar 302 and may also include windows, icons, and other elements. The user interface 300 has a view display area that may depict a visual representation of elements associated with a currently selected view.

[0055] The user interface 300 may have multiple applications running, any or all of which can be presented in a separate window. In one implementation, the user activates a time machine session by for example, selecting a “Time Machine” icon 304 located in an active window 306. Other means for activating the time machine are possible. For simplicity, one active window 306 and one inactive window 308 are shown. The active window 306 may contain various types of elements. For example, elements 1-4 shown in the active window 306 may include folders, files, information, playlists, or data of various types. In some implementations, the active window may be scaled upon entering the time machine to fit within the presentation interface. For example, if an application is running in full screen mode (e.g., iTunes) when the time machine session is initiated, the application window may be scaled down to fit within the time machine interface.

[0056] FIG. 4 is a screen shot for one implementation depicting an initial state for a time machine user interface 402 after the Time Machine icon 304 has been selected. The time machine interface 402 includes a presentation window 307, a timeline 404, and function buttons. The timeline 404 presents a history view associated with the current view. The time machine interface 402 is shown in FIG. 4 as a layer differentiated from the desktop by a dashed line around the perimeter of the user interface. In one implementation, the user interface 300 and associated active window 306 is obscured or otherwise hidden while the time machine is active.

[0057] The presentation window 307 may show the currently selected snapshot, or a portion thereof. Alternatively,

the presentation window **307** may show a selected one of several available snapshots. In one example, a user may operate the time machine interface so as to sequentially select each snapshot and may restore elements from these selected snapshots until the current version engulfs the entire history of the timeline **404**, or some portion thereof. For example, the user may select a snapshot in the timeline **404** and merge its contents with the current active window contents by restoring each element individually, and keeping both versions in some cases, to achieve a merged list of elements. Additional display options for the presentation window **307** will be discussed in detail in FIG. 7 of this description.

[0058] The timeline **404** may include a number of snapshots representing slices of time for active window elements. Each snapshot provides a screenshot representation of an earlier version of the active window **306** at a particular point in time. The snapshots can represent periodic backup points in time or can represent snapshots of the active window where a change has occurred. FIG. 4 shows a scenario where the snapshots represent changes over time.

[0059] In some implementations the timeline **404** includes a visual representation of elements, and can appear across the top portion of the time machine interface **402** (as shown). Alternatively, the timeline may not appear in the top portion of the time machine interface **402** until a user moves their cursor (or otherwise activates) to the top portion of the interface.

[0060] In alternative implementations, all snapshots may be shown according to when they are taken regardless of changes between snapshots. In another alternative implementation, the snapshot may indicate a range of time covered by a single snapshot where no changes have occurred (e.g., Jun. 1-Aug. 8, 2005). In another implementation, the rate of snapshots can vary over time. For example, there may be a number of snapshots in the newest archives, but the snapshots become more spaced in time as they reach further into the past. For example, hourly snapshots for the current day may turn into daily snapshots, which turn into weekly snapshots that become monthly snapshots as their file dates reach further into the past. The transition to later snapshots can be achieved in one implementation, by selecting a representative snapshot from a group of snapshots and deleting the rest.

[0061] As shown in FIG. 4, the most recent snapshot **408** is shown with a highlighted border. This represents the present state of the contents of the presentation window **307**. The other snapshots shown in the timeline **404** represent states of the active window **306** in the past from which there have been changes to the content of the active window **306**. For example, the snapshot **410** shows a snapshot of the window which includes three elements instead of four elements. Between the prior snapshot **410**, and the current snapshot **408**, element **4** has been added.

[0062] The other snapshots in the timeline **404** show other configurations of past states of the active window **306**. For example, different numbers of elements show additions and deletions have occurred over time. The time machine interface **402** may also show modifications that have been made between different versions of folders, files, or items, and the like.

[0063] Arrow buttons **412a** and **412b** shown to the left and right of the snapshots allow the user to navigate additional snapshots not shown, thus there can be a large number of snapshots to select from.

[0064] The time machine interface **402** includes a restore button **414**, a changed items only button **416**, and an information button **418**. Other buttons are possible. The restore button **414**, when selected, restores the window to the selected state represented by the selected snapshot presented in the presentation window **306** and exits the time machine **402** (e.g. restoring the user interface **300**). A user may select a snapshot and then select the restore button **414** to modify the current version of the element selected. The changed items only button **416** filters the snapshots to show only those that differ from the current state. That is, the changed items only button **416** does not refer to the incremental changes between snapshots in the timeline **404**, rather, it refers to omitting those snapshots whose states are identical to the current state of the active window **306**. For example, if the most recent snapshot **408** is identical to a snapshot **410** that occurs earlier in time, selecting the changed items only button **416** will cause the time machine to cease displaying one of the copies, e.g., remove snapshot **410** from the timeline. This may help the user locate a previous version from which to restore contents different than those in the current version.

[0065] The information button **418** provides information regarding the selected snapshot. In one implementation, selecting the information button **418** opens a panel display. The panel display provides, in one implementation, information including: the date and time the snapshot was made; the location of actual elements of a snapshot; the size of the snapshot; and a comment section.

[0066] While the previous example involved changes that were visualized through the presence or absence of elements in the window, the following example involves also differences at a deeper level. FIG. 5 is a screen shot depicting a time machine user interface where a user has selected a particular snapshot **502**. While both a current version **504** and the snapshot version **502** of the window include element **4**, the element **4** in the snapshot **502** is a prior version of element **4** in the current version **504**. Upon the selection, the presentation window **307** changes to reflect the contents of the selected snapshot **502**. For example, if snapshot **502** has versions of a word processing file from last week, the user may wish to go back to this version if an error occurs in the current version of the file. The user may choose an appropriate version and then select the restore button **414** to return to the older file. However, since a version of the file exists in the current document, a conflict will arise.

[0067] FIG. 6 is a screen shot depicting a time machine user interface after the snapshot **502** has been selected for restore. The presentation window **307** changes to reflect the newly selected snapshot. For example, if element **4** is a file that has been corrupted in the current version, the user could select the restore button **414** (shown in FIG. 5) and retrieve an earlier version of element **4**. However, restoring the entire window restores everything in the current version with an earlier version of each element, thereby overwriting potentially useful information. In some implementations, the time machine user interface allows the user to analyze changed content before a restore is performed. For example, the restore button **414** (shown in FIG. 5) may change to a restore element button **602** after the snapshot **502** has been selected.

The restore element button **602** may restore a selected element and leave the rest of the file structure intact. In this example, when the restore button **414** is selected, a conflict is generated because the current version of the active window **306** already includes all elements in the snapshot version **502**. Therefore, a conflict management screen will be provided to resolve the conflict.

[0068] FIG. 7 is a screen shot depicting a conflict management screen **702**. In one implementation, the screen **702** is generated by the backup conflict resolution engine **130**. The conflict management screen **702** may be provided when the user attempts to restore an already existing element. The conflict management screen **702** notifies the user that the element already exists and can also present representations of the conflicting elements including the date of the snapshot element. For example, if a conflict arises, the conflict management screen **702** can graphically display the differences between the elements that are conflicting, such as element **4** (**704**) that was modified in November 2005 and element **4** (**706**) that is an updated version of the same element. When the user is notified about a conflict, one or more options may be presented: cancel restore **708**, keep both **710**, replace current **712**, and merge **714**. The cancel restore button **708** cancels an element restore and returns the user to the time machine interface **402** (FIG. 4). This means that no change is made in the current version. The keep both button **710** allows the user to keep both elements, for example, by renaming the snapshot version of element **4**. After selection of the keep both button, the time machine can exit back to the time machine interface **402**, but with both the restored earlier version of element **4** (renamed) and the current element **4**. The replace current button **712** allows a user to replace the current version of an element with a previous version. For example, if information was unintentionally deleted from a document, the replace current version button **712** can be used to restore the data using a previously saved version of the document. The time machine can exit and returns to the time machine interface **402**, however, the element **4** shown is the restored version.

[0069] In some implementations, the merge button **714** may be used to combine multiple snapshots to create the current version. In one implementation, the elements may be folders where it is possible to merge the contents of the folders into a single folder. For example, if a current folder has files A, B, and C, and the earlier snapshot includes files B, C, D, and E, then the merge would only restore the files of the snapshot that were not present in the current version. The merge option allows for creating the most current version of a file or folder using multiple previous versions. In this example, the merge button **714** may be selected and the folder would include the current A, B, and C files plus the snapshot D and E files.

[0070] In some implementations, when the user moves the cursor over one of the option buttons, a rollover effect can be generated which graphically indicates the result of a selection of that button. For example, if the user moves the cursor over the replace current button **712**, the graphics above can change to show the replacement effect. For example, a red "X" may appear over the current element **4** (**706**) to indicate the element will be discarded when the conflict is resolved. Similarly, a green checkmark may appear in the snapshot element **4** (**704**) to indicate what will occur when the conflict is resolved. For example, if the restore is cancelled, the rollover effect may use a green

checkmark to indicate that the current version of the element will be kept. In some implementations, the rollover effect may graphically show a merge by visually displaying the combination of a current version of an element and an earlier version of the same element to indicate both files will be kept. In some implementations, an arrow can be shown pointing from the snapshot **704** to the current version **706** to indicate that snapshot **704** is replacing the current version **706**.

[0071] In one implementation, when a conflict is identified, a user can select a particular rule from a set of rules to use in resolving the conflict. In another implementation, an identified conflict can be automatically resolved according to one or more rules without user interaction. The rules can be user selected and/or ordered, programmatically defined, or otherwise generated. For example, the rules can state that a conflict is always resolved in favor of the newer version of an element.

[0072] In the above description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding. It will be apparent, however, to one skilled in the art that implementations can be practiced without these specific details. In other instances, structures and devices are shown in block diagram form in order to avoid obscuring the disclosure.

[0073] In particular, one skilled in the art will recognize that other architectures and graphics environments may be used, and that the present examples can be implemented using graphics tools and products other than those described above. In particular, the client/server approach is merely one example of an architecture for providing the functionality described herein; one skilled in the art will recognize that other, non-client/server approaches can also be used.

[0074] Some portions of the detailed description are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0075] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0076] An apparatus for performing the operations herein may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD ROMs, and magnetic optical disks, read only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0077] The algorithms and modules presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatuses to perform the method steps. The required structure for a variety of these systems will appear from the description. In addition, the present examples are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings as described herein. Furthermore, as will be apparent to one of ordinary skill in the relevant art, the modules, features, attributes, methodologies, and other aspects can be implemented as software, hardware, firmware or any combination of the three. Of course, wherever a component is implemented as software, the component can be implemented as a standalone program, as part of a larger program, as a plurality of separate programs, as a statically or dynamically linked library, as a kernel loadable module, as a device driver, and/or in every and any other way known now or in the future to those of skill in the art of computer programming. Additionally, the present description is in no way limited to implementation in any specific operating system or environment.

[0078] The subject matter described in this specification can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer readable medium for execution by, or to control the operation of, data processing apparatus. The instructions can be organized into modules (or engines) in different numbers and combinations from the exemplary modules described. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more of them. The term "data processing apparatus" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[0079] While this specification contains many specifics, these should not be construed as limitations on the scope of what may be claimed, but rather as descriptions of features

specific to particular implementations of the subject matter. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0080] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

[0081] The subject matter of this specification has been described in terms of particular embodiments, but other embodiments can be implemented and are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In certain implementations, multitasking and parallel processing may be advantageous. Other variations are within the scope of the following claims.

What is claimed is:

1. A method relating to modifying a view in a user interface, the method comprising:
 - receiving, while a current view is displayed in a user interface, a first user input requesting that a history view associated with the current view be displayed;
 - displaying the history view in response to the first user input, the history view including at least a first visual representation of an earlier version of the current view, the earlier version including a first element;
 - receiving, while the history view is displayed, a second user input requesting that the current view be modified according to the earlier version, at least with regard to the first element; and
 - determining, in response to the second user input, whether the current view includes a current version of the first element.
2. The method of claim 1, wherein the determination indicates that the current view does not include a current version of the first element, further comprising modifying, in response to the determination, the current view to include at least the first element.
3. The method of claim 1, wherein the determination indicates that the current view does include a current version of the first element, further comprising displaying a conflict resolution view in response to the determination.

4. The method of claim 3, wherein the conflict resolution view provides user-selectable conflict resolution by at least one option of:

- canceling the request that the current view be modified; replacing the current version of the first element with the first element included in the earlier version;
- modifying the current view to include both the current version of the first element and the first element included in the earlier version.

5. The method of claim 4, wherein the user-selectable conflict resolution provides the option of merging, in the current view, the current version of the first element and the first element included in the earlier version.

6. The method of claim 4, wherein the user-selectable conflict resolution includes a rollover effect graphically indicating a result of selecting each of the options.

7. The method of claim 6, wherein, for the option of canceling the request that the current view be modified, the rollover effect indicates that the current version of the first element will be kept.

8. The method of claim 6, wherein, for the option of replacing the current version of the first element with the first element included in the earlier version, the rollover effect indicates that the current version of the first element will be discarded.

9. The method of claim 4, wherein, for the option of modifying the current view to include both the current version of the first element and the first element included in the earlier version, the rollover effect indicates that both the current version of the first element and the first element included in the earlier version will be kept.

10. The method of claim 3, wherein the conflict resolution view contains the current version of the first element and the first element included in the earlier version.

11. The method of claim 1, wherein the first element is one selected from the group consisting of: a folder, a file, an item, an information portion, a playlist, a directory, an image, system parameters, and combinations thereof.

12. A computer-readable medium having stored thereon instructions which, when executed by a processor, causes the processor to perform operations comprising:

- receiving, while a current view is displayed in a user interface, a first user input requesting that a history view associated with the current view be displayed;
- displaying the history view in response to the first user input, the history view including at least a first visual representation of an earlier version of the current view, the earlier version including a first element;
- receiving, while the history view is displayed, a second user input requesting that the current view be modified according to the earlier version, at least with regard to the first element; and
- determining, in response to the second user input, whether the current view includes a current version of the first element.

13. A computer program product tangibly embodied in an information carrier, the computer program product including instructions that, when executed, generate on a display device a graphical user interface comprising:

- a view display area for presenting a current view;
- a history view display area for presenting a history view associated with the current view, the history view including at least a first visual representation of an earlier version of the current view, the history view

providing that a user can initiate a restoration of the current view with at least a portion of the earlier version; and

- a conflict resolution function for alerting the user to a conflict in performing the restoration.

14. The computer program product of claim 13, wherein the conflict resolution function includes at least one option of:

- an input control to cancel restoration;
- an input control to keep both conflicting elements;
- an input control to perform the restoration; and
- an input control to merge both conflicting elements.

15. The computer program product of claim 14, wherein a rollover effect is provided that graphically indicates a result of selecting any of the options.

16. The computer program product of claim 13, wherein the first visual representation is included in a timeline presented in the history view, the timeline including several visual representations of earlier versions of the current view.

17. The computer program product of claim 16, wherein the history view further includes an input control for modifying the timeline to include only at least one of the visual representations whose corresponding earlier version differs from the current view.

18. The computer program product of claim 13, wherein the conflict resolution function is included in a conflict resolution view that presents visual representations of respective elements in the current version and the earlier version that give rise to the conflict.

19. A method comprising:

- defining a criterion for capturing a state of a view of a user interface;
- capturing the state of the view in accordance with the criterion;
- receiving a prompt to suspend presentation of a current view and present a captured view;
- determining if a conflict exists between the captured view and the current view; and
- if not, reinstating the captured view into the current view of the user interface.

20. The method of claim 19, further comprising if a conflict is detected, resolving the conflict prior to reinstating the captured view.

21. The method of claim 20, wherein the conflict is deemed to exist if the determination indicates that information present in the current view would be lost if the captured view were reinstated into the current view.

22. A computer-readable medium having stored thereon instructions which, when executed by a processor, causes the processor to perform operations comprising:

- defining a criterion for capturing a state of a view of a user interface;
- capturing the state of the view in accordance with the criterion;
- receiving a prompt to suspend presentation of a current view and present a captured view;
- determining if a conflict exists between the captured view and the current view; and
- if not, reinstating the captured view into the current view of the user interface.

23. A method comprising:
evaluating a current view of a user interface;
determining that an undesirable change to the current view of the user interface has occurred relative to a past view of the user interface;
suspending the current view of the user interface;
presenting one or more past views of the user interface;
determining if a conflict exists between the captured view and the current view; and
if not, reinstating a past view of the user interface into the current view.

24. A computer-readable medium having stored thereon instructions which, when executed by a processor, causes the processor to perform operations comprising:
evaluating a current view of a user interface;
determining that an undesirable change to the current view of the user interface has occurred relative to a past view of the user interface;
suspending the current view of the user interface;
presenting one or more past views of the user interface;
determining if a conflict exists between the captured view and the current view; and
if not, reinstating a past view of the user interface into the current view.

25. A method comprising:
presenting an element in a user interface, the element having a current state and being presented in a current view of a user interface;
capturing over time, a state of the element;
suspending a current view of the user interface including the element; and
reinstating a past view of the element including resolving a conflict between the current view and past view with respect to the element.

26. A computer-readable medium having stored thereon instructions which, when executed by a processor, causes the processor to perform operations comprising:
presenting an element in a user interface, the element having a current state and being presented in a current view of a user interface;
capturing over time, a state of the element;
suspending a current view of the user interface including the element;
reinstating a past view of the element including resolving a conflict between the current view and past view with respect to the element.

* * * * *