[0149] At step **1660** it is determined whether the "A" setting indicates that the button is to accept Quicklaunch or Favorites features. If YES, the appropriate attributes are applied to the button at a step **1662**. If not, flow proceeds to step **1638** so as to process any further buttons. Once all buttons are processed, flow continues back to step **1314** as shown at a step **1642**.

[0150] These collections of styles that can be applied to a button (attributes, properties, or the like, e.g., a bitmap, a font, tool tip, flyover characteristic) have a particular precedence. Such characteristics are defined in the configuration file. If no such characteristics are found in the configuration file, the theme is searched. If such a characteristic is found in the theme it is applied. If no such characteristic is found in the theme, or it wasn't in the configuration file, the template is searched. If no such characteristic is located in the template, any predetermined default is applied. Thus, softer defaults as well as overrides at the template level, theme level, and user configuration level are available according to this embodiment of the present invention. The flow of **FIG. 16** follows this iteration to determine what attributes to apply to the buttons. For example, the "set tweak amount" for positioning at step **1646** searches the configuration file and the theme to determine whether an modification has been made to the position of the button. In the case in which Zenu™ buttons are nested, in other words, clicking a button opens another level of buttons, each level of buttons has different parent templates, and a theme associated with each level. Thus, the position of buttons of a particular level is determined by their template and their appearance is determined by the theme of that level.

[0151] Turning again to the "waiting for an event" step **1314**, two events can occur: a "button click", which is described in connection with **FIGS. 17 and 18**, or "a dropped file unbutton file event", which is described at **FIG. 19**.

[0152] **FIGS. 17 and 18** describe the process that occurs when a button is clicked.

[0153] This process begins at a step **1702**, and proceeds to determine whether the UDI is in button configuration mode, at a step **1704**. If so, the current settings for the button are displayed in the configuration window at a step **1706**. The process then enters the "wait for event" mode, at a step **1716** (which is equivalent to the wait for event step **1314**). If not in the configuration mode, flow proceeds to a step **1708**, which determines if there is a user defined button command for the button that applies to this theme. If so, the command is executed at a step **1710**, then flow proceeds to step **1716**. If no button command is defined, flow proceeds to step **1712** to determine if there is a built-in command for the button with the theme applied to the UDI as specified in the theme file. If YES, that command is then executed at step **1710**. If not, flow proceeds to step **1714** to determine if a default command for the button is found in the default template file. If so, the default command is executed at step **1710**. Otherwise, flow proceeds to step **1716** to wait for another event.

[0154] **FIG. 18** illustrates the process for executing a command formed at step **1710**. The execute command process begins at a step **1802**. The command string is parsed at a step **1804**. The syntax of the command string as described above in connection with Tables 1 and 2. Next, the first command is evaluated at a step **1806**. If the command is a plug-in, as determined at a step **1808**, the plug-in .dll is loaded to create a plug-in object and a command is executed at a step **1810**. If the command is not a plug-in, it is determined whether the command specifies opening a template or theme, at a step **1812**. If so, the theme, template, or both are opened, at a step **1814**. Next, it is determined whether the command was to close a template, at a step **1816**. If so, the closed template specified is performed, at a step **1818**. As a result of steps **1810**, **1814** and step **1818**, or if the result of step **1816** is NO, it is next determined whether a command has been executed, at a step **1820**. If YES, it is then determined whether there is another command in the string to process, at a step **1822**. If so, the next command is obtained, at a step **1826**, and flow proceeds to evaluate the command, at step **1808**. If the result of the query in step **1820** is NO, a shell execute command is performed, at a step **1824**. Control then proceeds after step **1824** to step **1822**. If no other commands are to be executed in the string flow proceeds to step **1828** to wait for an event, which is the equivalent of "wait for event" step **1314**.

[0155] **FIG. 19** illustrates the process for handling a "dropped file on button event." Flow begins at a step **1902**. Next, a short-cut is created to the file that is dropped on the button, and that short-cut is placed in a Zenu™ short-cut directory, at a step **1904**. Next, a short-cut icon for the file association is placed on the button and is modified according to the theme/layer specific characteristics, at a step **1906**. Finally, the button command is edited based on the current theme, so that when the button is clicked the file is opened if that theme is currently applied, at a step **1908**. The wait for event step is entered again at a step **1910**.

[0156] IV Example Computer System and Computer Program Product Implementations

[0157] The Zenu™ UDI of the present invention can be implemented using hardware, software or a combination thereof and may be implemented in one or more computer systems or other processing systems. In fact, in one embodiment, the invention is directed toward one or more computer systems capable of carrying out the functionality described herein. An example of a computer system **2000** is shown in **FIG. 20**. The computer system **2000** includes one or more processors, such as processor **2004**. Processor **2004** can support various operating systems such as Microsoft® Windows, Unix, Lixux, or the like. The processor **2004** is connected to a communication infrastructure **2006** (e.g., a communications bus, cross-over bar, or network). Various software embodiments are described in terms of this exemplary computer system. After reading this description, it will become apparent to a person skilled in the relevant art(s) how to implement the invention using other computer systems and/or computer architectures.

[0158] Computer system **2000** can include a display interface **2005** that forwards graphics, text, and other data from the communication infrastructure **2002** (or from a frame buffer not shown) for display on the display device **2030**.

[0159] Computer system **2000** also includes a main memory **2008**, preferably random access memory (RAM), and can also include a secondary memory **2010**. The secondary memory **2010** can include, for example, a hard disk drive **2012** and/or a removable storage drive **2014**, representing a floppy disk drive, a magnetic tape drive, an optical disk drive, or the like. The removable storage drive **2014**