MARKUP BASED EXTENSIBILITY FOR USER INTERFACES

Background

With the advent of the computer age, computer and software users have grown accustomed to user-friendly software applications that help them write, calculate, organize, prepare presentations, send and receive electronic mail, make music, and the like. For example, modern electronic word processing applications allow users to prepare a variety of useful documents. Modern spreadsheet applications allow users to enter, manipulate, and organize data. Modern electronic slide presentation applications allow users to create a variety of slide presentations containing text, pictures, data or other useful objects. Modern database applications allow users to store, organize and exchange large amounts of data.

Most software applications provide one or more graphical user interfaces through which a user enters and edits data and from which the user accesses and utilizes various functionalities of the associated software application. A typical user interface includes a work area in which data may be entered, edited, and reviewed. Additionally, user interfaces typically include one or more buttons and/or controls operative for selecting the functionalities provided by the associated software application. For example, buttons or controls may be provided for printing or saving a document, buttons or controls may be provided for applying formatting properties to aspects of a document, and the like.

Often, a third party software developer creates a software add-in that may be added to an existing application for providing functionality not available from the existing application. For example, an add-in software application may provide a feature to a word processing application for adding specialized footnotes or endnotes to a document. Typically, in addition to providing additional functionality, the add-in application provides one or more new user interface components to the existing application user interfaces, such as a new toolbar, button(s), or other control(s), for accessing the additional functionality.

According to one prior method, third party developers are given access to an object model associated with an application's existing user interfaces for allowing customization of the existing user interfaces according to the needs of third party add-in software. Unfortunately, such prior methods have shortcomings because the object

models for given user interfaces are not typically designed around common uses across a variety of different applications, for example, word processing applications, spreadsheet applications, slide presentation applications, and the like, and often such applications exhibit different and perhaps undesirable behaviors in association with custom user interface components.

It is with respect to these and other considerations that the present invention has been made.

Summary

This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the detailed description. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended as an aid in determining the scope of the claimed subject matter.

Embodiments of the present invention solve the above and other problems by providing methods, systems and computer products for exposing a software application user interface programming to third party application add-in developers to allow modification of existing application user interfaces to include new or modified user interface components associated with add-in applications. According to aspects of the invention, an Extensible Markup Language (XML) schema that governs XML programming that may be used to modify a given user interface is exposed to third party developers to allow them to specify changes to the user interface programming according the associated XML schema. For example, if a third party developer desires to add a new button or control to an existing user interface that would be linked to a functionality of an add-in application, the third party developer can modify the existing programming of the user interface according to the grammatical and syntax rules dictated by the associated XML schema. According to an aspect of the invention, XML or other suitable representations of user interface modifications do not necessarily follow the same programming language as the original user interface. Moreover, the original built-in user interface programming may be very complex, and the XML schema exposed according to the present invention may be only a subset of the overall programming for the original user interface. When the modified programming is executed by the host software application, the user interface is rendered with the changes made by the third party developer. For example, if the

programming of the user interface is modified to add a new button, then the new button will be rendered in the user interface in response to the modification to the programming for the user interface so long as the modification is done according to the associated XML schema.

According to aspects of the invention, new user interface components may be added to existing user interfaces and may be linked to associated add-in functionality. The sizes of new user interface components may be automatically scaled to fit available display space as a window in which the user interface is displayed is reduced or enlarged. In addition, according to aspects of the invention, end users of a modified user interface may remove added user interface components if desired. If a particular software add-in is un-installed, the added or modified user interface components associated with the un-installed add-in are not shown in the subsequently rendered user interface.

According to other aspects of the invention, existing user interface components such as buttons or controls may be disabled or may be removed altogether by third parties. In addition, existing user interface components may be repurposed so that the repurposed components exhibit different behaviors when selected.

According to other aspects of the invention, by accessing the schema exposed for modifying an existing user interface, third party contextual user interfaces and contextual user interface buttons or controls may be added to existing user interfaces that are exposed in the existing user interfaces when a document object associated with the added contextual user interface is selected. In addition, controls, which when selected cause the application of one or more add-in functionalities to a selected object, may be added to a gallery or collection of controls that are deployed in the existing user interface for applying one or more functionalities of the application to a selected object.

According to other aspects of the invention, the XML schema may be utilized to build start-from-scratch user interfaces that are customized according to the needs of a third party add-in to the associated software application. When such a start-from-scratch user interface solution is rendered, the resulting user interface may bear little resemblance to the application user interface normally exposed to users for the associated software application. That is, when the associated document is launched, a

customized user interface for providing the user functionality to the launched document is rendered according to the customized user interface provided for the document.

These and other features and advantages, which characterize the present invention, will be apparent from a reading of the following detailed description and a review of the associated drawings. It is to be understood that both the foregoing general description and the following detailed description are explanatory only and are not restrictive of the invention as claimed.

Brief Description of the Drawings

Fig. 1 illustrates an exemplary computing operating environment for embodiments of the present invention.

Fig. 2 is a computer screen display showing an example user interface that may be modified according to embodiments of the present invention.

Fig. 3 is a simplified block diagram showing a relationship between an example user interface and an XML representation of the example user interface that may be amended for modifying the example user interface according to embodiments of the present invention.

Fig. 4 is a computer screen display showing an example contextual user interface that may be modified according to embodiments of the present invention.

Fig. 5 is a computer screen display showing an example gallery of selectable controls user interface that may be modified according to embodiments of the present invention.

Detailed Description

As briefly described above, embodiments of the present invention are directed to methods, systems, and computer products for exposing the programming of an application user interface to allow modification of an associated user interface to include adding, removing, disabling, enabling and repurposing new or existing user interface components. In the following detailed description, references are made to the accompanying drawings that form a part hereof, and in which are shown by way of illustrations specific embodiments or examples. These embodiments may be combined, other embodiments may be utilized, and structural changes may be made

without departing from the spirit or scope of the present invention. The following detailed description is therefore not to be taken in a limiting sense, and the scope of the present invention is defined by the appended claims and their equivalents.

Referring now to the drawings, in which like numerals refer to like elements through the several figures, aspects of the present invention and an exemplary computing operating environment will be described. Figure 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of program modules that execute in conjunction with an application program that runs on an operating system on a personal computer, those skilled in the art will recognize that the invention may also be implemented in combination with other program modules.

Generally, program modules include routines, programs, components, data structures, and other types of structures that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Embodiments of the invention may be implemented as a computer process (method), a computing system, or as an article of manufacture, such as a computer program product or computer readable media. The computer program product may be a computer storage media readable by a computer system and encoding a computer program of instructions for executing a computer process. The computer program product may also be a propagated signal on a carrier readable by a computing system and encoding a computer program of instructions for executing a computer process.

With reference to Figure 1, one exemplary system for implementing the invention includes a computing device, such as computing device 100. In a basic configuration, the computing device 100 typically includes at least one processing unit 102 and system memory 104. Depending on the exact configuration and type of

computing device, the system memory 104 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. System memory 104 typically includes an operating system 105 suitable for controlling the operation of a networked personal computer, such as the WINDOWS® operating systems from MICROSOFT CORPORATION of Redmond, Washington. The system memory 104 may also include one or more software applications 106, 120 and may include program data 107. This basic configuration is illustrated in Figure 1 by those components within dashed line 108.

According to embodiments of the invention, the application 106 may comprise many types of programs, such as an electronic mail program, a calendaring program, an Internet browsing program, and the like. An example of such programs is OUTLOOK® manufactured by MICROSOFT CORPORATION. The application 106 may also comprise a multiple-functionality software application for providing many other types of functionalities. Such a multiple-functionality application may include a number of program modules, such as a word processing program, a spreadsheet program, a slide presentation program, a database program, and the like. An example of such a multiple-functionality application is OFFICE™ manufactured by MICROSOFT CORPORATION. The add-in software application 120 may comprise any software application that may be added to the applications 106 for enhancing or providing additional functionality to the applications 106 as described herein. In addition, an add-in software application, as described herein, may include document-based software solutions, for example, a spreadsheet document that includes attached toolbars, or a word processing document that contains a macro or other code that adds a toolbar with buttons or controls.

The computing device 100 may have additional features or functionality. For example, the computing device 100 may also include additional data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Such additional storage is illustrated in Figure 1 by removable storage 109 and non-removable storage 110. Computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. System memory 104, removable storage 109 and non-removable storage 110 are all examples of computer storage

media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computing device 100. Any such computer storage media may be part of device 100. Computing device 100 may also have input device(s) 112 such as keyboard, mouse, pen, voice input device, touch input device, etc. Output device(s) 114 such as a display, speakers, printer, etc. may also be included. These devices are well known in the art and need not be discussed at length here.

The computing device 100 may also contain communication connections 116 that allow the device to communicate with other computing devices 118, such as over a network in a distributed computing environment, for example, an intranet or the Internet. Communication connection 116 is one example of communication media. Communication media may typically be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. The term computer readable media as used herein includes both storage media and communication media.

Fig. 2 is an illustration of a computer screen display showing an example user interface that may be modified according embodiments of the present invention. As should be understood by those skilled in the art, the example user interface illustrated in Fig. 2 is for purposes of example and illustration only. That is, embodiments of the present invention may be utilized for a vast array of different user interfaces having different user interface components and different user interface layouts. Thus, the description of the present invention in terms of the example user interface illustrated in Fig. 2 should not be taken as restrictive or limiting in any way of the claimed invention.

The user interface illustrated in Fig. 2 includes a ribbon-shaped user interface for displaying selectable controls associated with task-based functionality available under a given software application, such as the software application 106 illustrated in Fig. 1. A first section 210 of the user interface 200 includes generic selectable controls for functionality not associated with a particular task, such as word processing versus spreadsheet data analysis. For example, the section 210 includes selectable controls for general file commands such as "file open," "file save" and "print." According to one embodiment of the present invention, the selectable controls included in the first section 210 are controls that may be utilized by a variety of software applications, for example, a word processing application, spreadsheet application, slide presentation application, and the like. That is, the selectable controls included in the first section 210 may be controls that are generally found and used across a number of different software applications.

Adjacent to the first section 210 of the user interface 200 is a task-based tab section. The tab section includes selectable tabs associated with task-based functionality provided by a given software application. For purposes of example, the task-based tabs illustrated in Fig. 2 are associated with tasks that may be performed using a word processing application. For example, a "Writing" tab 215 is associated with functionality that may be utilized for performing writing tasks. An "Insert" tab 220 is associated with functionality associated with performing insert operations or tasks. A "Page Layout" tab 230 is associated with functionality provided by the associated application for performing or editing page layout attributes of a given document.

As should be appreciated, many other task-based tabs or selectable controls may be added to the tab section of the user interface for calling functionality associated with other tasks. For example, task tabs may be added for text effects, document styles, review and comment, and the like. And, as described above, the user interface 200 may be utilized for a variety of different software applications. For example, if the user interface 200 is utilized for a slide presentation application, tabs contained in the tab section may include such tabs as "Create Slides," "Insert," "Format," "Drawing Effects," and the like associated with a variety of tasks that may be performed by a slide presentation application. Similarly, tabs that may be utilized in the tab section of the user interface 200 for a spreadsheet application 140 may

include such tabs as "Data" or "Data Entry," "Lists," "Pivot Tables," "Analysis," "Formulas," "Pages and Printing," and the like associated with tasks that may be performed using a spreadsheet application.

Immediately beneath the generic controls section 210 and the task-based tab section is a selectable functionality control section for displaying selectable functionality controls associated with a selected tab 215, 220, 230 from the task-based tab section. According to embodiments of the present invention, when a particular tab, such as the "Writing" tab 215 is selected, selectable functionality available from the associated software application for performing the selected task, for example, a writing task, is displayed in logical groupings. For example, referring to Fig. 2, a first logical grouping 240 is displayed under a heading "Clipboard." According to embodiments of the present invention, the clipboard section 240 includes selectable functionality controls logically grouped together and associated with clipboard actions underneath the general task of writing.

Selectable controls presented in the "Formatting" section 250 may include such selectable controls as text justification, text type, font size, line spacing, boldface, italics, underline, etc. Accordingly, functionalities associated with formatting operations are logically grouped together underneath the overall task of "Writing." A third logical grouping 260 is presented under the heading "Writing Tools." The writing tools section 260 includes such writing tools as find/replace, autocorrect, etc. According to embodiments of the present invention, upon selection of a different task-based tab from the tab section, a different set of selectable functionality controls in different logical groupings is presented in the user interface 200 associated with the selected task-based tab. For example, if the "Insert" task tab 220 is selected, the selectable functionality controls presented in the user interface 200 are changed from those illustrated in Fig. 2 to include selectable functionality controls associated with the insert task.

As described herein, often third party application or solution developers provide software applications or modules that may be added to an existing application for adding functionality to the existing application. For example, a software developer may produce a software application or module that may be added to a word processing application for adding additional formatting properties or other functionalities that are not available in the host application (e.g., word processing

application, spreadsheet application, slide presentation application, etc.), or the third party add-in application or module may modify or enhance functionalities already provided by the host application. As described herein, embodiments of the present invention provide access to the programming associated with one or more user interfaces of a host application to allow a third party application developer to modify the user interfaces of the host application to provide for user interface components that are applicable to added or modified functionality provided by a given add-in application or module.

As will be appreciated, embodiments of the present invention are not limited to use in association with add-in applications. For example, a third party developer may utilize aspects of the invention to modify components of an application user interface without regard to added functionality. For example, a third party may desire to change the behavior of a user interface component apart from any added functionality as described herein.

Fig. 3 is a simplified block diagram showing a relationship between an example user interface and an XML representation of the example user interface that may be amended for modifying the example user interface according to embodiments of the present invention. Referring to Fig. 3, a user interface 305 of a host application 106, for example, a word processing application, spreadsheet application, slide presentation application, and the like, is illustrated having a File command button 310, a Tab1 button 315, and a Tab2 button 320. As described above with reference to Fig. 2, the example user interface 305, illustrated in Fig. 3, is for purposes of example only and is not limiting of the vast number of buttons, controls, or other selectable functionalities that may be provided in a given user interface according to embodiments of the present invention. The example user interface 305 includes a first logical grouping of buttons or controls 325 and a second logical grouping of buttons or controls 330. According the example user interface 305, the Tab2 button 320 has been selected for providing the first and second logical groupings of buttons or controls 325, 330.

According to embodiments of the present invention, the user interface 305 is programmed and structured according to a markup language such as the Extensible Markup Language (XML). As should be understood, other languages suitable for programming and structuring a user interface 305 as described herein may be utilized.

As shown in Fig. 3, an XML file 340 is illustrated for providing the XML programming and structure for the example user interface 305. For example, a <UI> root tag is provided having three children tags <FILE>, <TAB1> and <TAB2>. The <TAB2> tag includes two children tags including the <LG1> tag (first logical grouping of buttons or controls) and the <LG2> tag (second logical grouping of buttons or controls). As should be understood, XML file 340 is not intended to illustrate a well-formed XML file, but is provided for purposes of illustration only.

An XML schema document or file (not shown) is provided by the developers of the host application that provides the XML grammatical and syntax rules applicable to XML used for programming and structuring the user interface 305, for example, for adding, removing or otherwise modifying new or existing tags to the XML file 340. For example, the XML schema may dictate the types and names of XML tags that may be added to the XML file 340. For example, the XML schema associated with the user interface 305 may require that each tab tag must include at least one logical grouping child tag. For another example, the XML schema associated with the user interface 305 may dictate that certain XML in the XML file 340 may not be altered, for example, logical grouping tags under a given tab tag of the original user interface 305 may not be altered by a third party.

For another example, the XML schema may dictate certain formatting properties that may or may not be applied to the user interface 305 via changes to the XML file 340. The entire XML schema file 340 and the entire associated XML schema file may be exposed to third party developers. Alternatively, only a subset of the XML file 340 associated with the host application user interface 305 and only a subset of the associated XML schema file may be exposed to third party developers for allowing them to make a limited number and limited types of changes to the host application user interface 305.

The XML file 340 for the host application user interface 305 and the associated XML schema file are exposed to a third party 350 for allowing the third party 350 to make changes to the XML file 340 according to the grammatical and syntax rules provided by the associated XML schema file for modifying the user interface 305, as described herein. For example, the third party 350 may provide a software application add-in that provides an additional functionality not presently available in the host application. For example, the third party may desire to add a

third tab button to the user interface 305 which when selected provides an additional logical grouping of buttons or controls underneath the added third tab button for providing buttons and/or controls for selecting functionality provided by the third party application add-in. For example, a third party application add-in may add a functionality for providing specialized endnotes or footnotes to a word processing application. In order to expose the functionality provided by the add-in to users, the third party developer may need to provide an additional tab in the user interface 305, for example, a "footnotes/endnotes" tab. In addition to adding the "footnotes/endnotes" tab, the third party developer may wish to add an additional logical grouping of buttons or controls that are displayed in the user interface 305 upon selection of the "footnotes/endnotes" tab which will provide buttons and/or controls for selecting various aspects of the added footnotes and endnotes functionality. As should be appreciated, existing buttons or controls may similarly be removed from the user interface 305 by modifying the associated XML file 340.

According to one embodiment, the markup exposed to the third party developer via the XML file 340 is of an incremental nature so that the third party developer is provided limited ability to modify the host application user interface 305. Thus, according to one embodiment, the third party developer does not receive an XML file 340 allowing them to insert their own user interface components at any location. Instead, according to this embodiment, the XML file 340 and the associated schema exposed to the third party developer allow the third party developer to specify a desired location for a new user interface component, for example, insert new component after existing component ABC, and the host application then integrates the amended XML file with the overall XML file representing the host application user interface 305 for rendering the amended user interface 307.

According to one embodiment, the amended XML file 355 provided by each third party developer is identified by a unique XML namespace to prevent one third party application add-in from modifying user interface components supplied by or modified by another third party add-in. Alternatively, the third-party add in file 355 may be identified by other means, for example, by a globally unique identifier (GUID) or by a local filename identification.

Once the XML representation 340 of the host application user interface 305 is accessed by the third party 350, the third party 350 may amend the XML applied to

the user interface 305 in accordance with the associated XML schema file for adding the desired buttons or controls to the user interface 305. The resulting amended XML file 355 illustrates changes made to the XML file 355 for the user interface 305. For example, referring to the amended XML file 355, the third party developer 350 has added a <TAB3> tag 360 and an <LG4> tab (fourth logical grouping of buttons and/or controls) 365. The XML file 355 illustrates the incremental nature of the modifications that may be made to the existing user interface according to embodiments of the present invention. That is, because the example modifications are directed to the addition of a new user interface tab (TAB3) and associated logical grouping of buttons or controls (LG4), the XML file 355 need only make the incremental change of adding the new tab and logical grouping to the existing user interface as opposed to making global changes to the exposed XML for the user interface 305.

Once the desired changes made to the XML file 340 are applied to the host user interface 305, an amended user interface 307 may be rendered, as illustrated in Fig. 3. Referring to the amended user interface 307, the added third tab 370 and the added fourth logical grouping of buttons and/or controls 375 are shown. Once the added tabs and logical grouping of buttons or controls are linked to the associated functionality of the third party application add-in, selection of those added tabs, buttons or controls causes execution of the associated added functionality. Thus, by gaining access to the exposed XML file and associated XML schema for the host application user interface, the third party application developer is able to modify the host application user interface 305 for providing required buttons and/or controls for selecting functionality of the associated add-in application.

The following is an excerpt of an example XML file that has been modified by or supplied by a third party developer for adding a "Data Analysis" button or control to the end of a tools menu where the "Data Analysis" button is to be inserted after a logical grouping of buttons or controls labeled as "Data Tools." As illustrated in the example XML, such XML syntax as <insertAfter> is dictated by the associated XML schema file for allowing the third party application developer to insert a button referred to as "Data Analysis" after a logical grouping of buttons or controls referred to as "GroupingDataTools." As should be understood by those skilled in the art, the

example XML that follows is for purposes of illustration only and is not limiting of the claimed invention.

```
<tab id="msoTcidDataTab">
    <chunk id="atpk:MyChunk" insertAfter="GroupingDataTools">
        <bigButton label="Data Analysis" onAction="atpk:dosomething"
image="someImage">
    </chunk>
</tab>
```

Changes to the XML file 340 may be made by third party developers according to different methods. According to one embodiment, the XML file 340 and the associated schema may be accessed by a third party application developer at the application level. For example, a button or control may be provided in the user interface 305 for allowing the third party application developer to access the XML file 340 and the associated schema utilized for programming and structuring the user interface. As should be understood, the XML file 340 and the associated schema file may be provided through other means such as through a developer tool or developer application that may be utilized by the third party application developer for preparing modifications to the XML file 340 in accordance with the associated XML schema file for applying to the host application when the third party application add-in is loaded onto the host application.

According to another embodiment, the user interface 305 associated with a given host application may be customized on a document level. That is, a document-based user interface 305 may contain customizations made to the user interface particular to a given document. For example, a user may create a customized document, for example, a sales template document for use by hundreds or thousands of sales representatives of a large company. Because the document will be used by so many people, it may be desirable to a third party application developer to customize a user interface provided by the host application, for example, a spreadsheet application, so that customized buttons or controls are rendered in the user interface when the particular document is launched.

A file format for such a document-based solution may be utilized in which various functionalities and properties associated with the document exist as related components hosted in a container file. For example, the container file may contain one component representative of user entered data, another component representative of document structure, for example, template structure, another component representative of formatting properties applied to the document, another component representative of document-based user interfaces customized for the document, and so on. A third party application developer may enter such a document container and access the XML representation file and associated XML schema applicable to the user interface component of the document. Once the developer has accessed the XML file and the associated XML schema for the user interface 305, the developer may customize the user interface by modifying the XML in accordance with the associated XML schema, as described above.

According to another embodiment, the modified XML file 355 and the associated XML schema file may be attached as a resource to a component object model (COM) add-in supplied by the third party application developer for providing functionality of the add-in application. When the COM add-in is applied to the host application, for example, a word processing application, the modified XML file 355 is consumed by the host application for rendering the resulting modified user interface 307.

Referring back to Fig. 3, as described above, third party software application add-ins may be utilized for adding new user interface components to the existing user interface of a host application, for example, a word processing application, a spreadsheet application, a slide presentation application and the like. According to one embodiment of the present invention, a start-from-scratch mode is provided where the XML file 340 exposed to the third party application developer removes all but essential user interface components from the associated host application user interface. The associated XML schema then provides the third party application developer the ability to add back original user interface components, for example, the first tab 315, the second tab 320, and the logical grouping controls 325 and 330. In addition, the third party application developer may then add new user interface components according to the associated schema file.

According to this embodiment, a shortcut is provided for allowing the removal of all but essential user interface components from the host application user interface and for allowing the add-in developer to have a great deal of control over the customization of the user interface because the add-in developer is able to start with virtually a blank slate user interface from which to build the customized user interface. According to one embodiment, this shortcut is an XML attribute, which when set to "true," causes a removal of most or all original user interface components.

Some areas of a given host application user interface 305 and some components of the user interface may be restricted from access by add-in developers. That is, some areas of the host application user interface and some components may be specified such that third party add-in developers may not remove or otherwise modify those areas or components. For example, referring to the host application user interface 305 in Fig. 3, it may be desired that the file button 310 for providing certain functionalities across a variety of applications, for example, a word processing application, a spreadsheet application, a slide presentation application, and the like, should not be altered, removed or repurposed in any manner by a third party add-in developer. Thus, the file button 310 may be restricted from access by third party add-in developers. Likewise, once user interface components are added to a given user interface, those components may be designated as restricted to prevent additional third party add-in applications from making changes to those user interface components provided by a previous add-in.

According to one embodiment of the present invention, if a third party add-in application is un-installed from the host application, the customized XML file applied to the host application user interface for adding or otherwise modifying user interface components in association with the un-installed add-in application is parsed and changes to the host application user interface in association with the add-in application are disabled so that changes to the host application user interface are not rendered in the user interface when the associated add-in application is un-installed. Likewise, upon closing a document-based solution that modified the original graphical user interface, the graphical user interface is rendered such that changes to the graphical user interface associated with the modification to the XML representation are not rendered in the graphical user interface

By accessing the XML file 340 representing the host application user interface 305, existing user interface components may be repurposed so that repurposed components subsequently exhibit different behaviors. For example, an existing user interface component may be enabled, disabled, or may be specified for being associated with a different application action. For example, a third party application developer may desire that a given button, for example, a "print" button in a host application user interface may only be utilized for printing a document according to a prescribed print setting associated with the third party application add-in. For another example, a given function button or control may be disabled from use where the function conflicts with the operation of a function added to the host application by an add-in application.

Fig. 4 is a computer screen display showing an example contextual user interface that may be modified according to embodiments of the present invention. As should be understood by those skilled in the art, the contextual user interface illustrated in Fig. 4 is for purposes of example only and is not limiting of the many different layouts and types of content that may be applied to and included in a contextual user interface that may be provided in association with a selected document object. Thus, the contextual user interface illustrated in Fig. 4 is not limiting or restrictive of the claimed invention in any way. For example, the contextual user interface illustrated in Fig. 4, described below, is in the form of a user interface menu that deploys in association with a selected object. However, this embodiment of the present invention may be implemented in other forms such as the addition of buttons or controls, such as tabs, to an original user interface 305 that are deployed in the user interface upon the selection of an object in an electronic document.

Referring to Fig. 4, a document including an embedded picture object 410 is illustrated in a word processing application workspace. According to embodiments of the present invention, the context menu 420 may be launched adjacent to or near a selected object through a variety of methods, including but not limited to, selection of an given document object such as the example picture object 410. The context menu 420 includes selectable functionality controls that are relevant to editing the selected object in the selected document. That is, the context menu 420 is populated with one or more selectable functionality controls that may be utilized for editing a particular

selected object in a selected document. For example, referring to the context menu 420 illustrated in Fig. 4, the context menu is launched in the context of a selected picture object 410. Accordingly, the selectable functionality controls, such as the paste control, copy control, position control, reset picture control, and the like provide functionality to a user for editing attributes of the selected picture object 410. As should be understood by those skilled in the art, if the context menu 420 is launched in the context of another type of object, then the selectable functionality controls populated in the context menu 420 will be related to the other type of object.

According to embodiments of the present invention, by exposing the XML file 340 and associated XML schema for the contextual user interface 420, third party developers may add or modify user interface content contained in the contextual user interface 420 for creating customized contextual user interface content and components associated with add-in functionality that may be applied to a selected document object. For example, referring to Fig. 4, a third party add-in application may provide additional functionality for modifying or formatting the picture object 410 over the functionality provided in the host application contextual user interface 420. As described above with reference to Fig. 3, the third party add-in developer may add, disable, repurpose or otherwise modify user interface components in the contextual user interface 420 in association with the third party application add-in. For example, if the add-in application provides an additional formatting functionality that may be utilized for picture objects 410, an additional formatting button or group of formatting buttons and/or controls may be added to the contextual user interface 420 which will be rendered in the contextual user interface 420 when the picture object 410 is selected for editing. As described above, instead of modifying the contextual user interface 420, new functionality controls, for example, new tabs may be added to a user interface 305 (illustrated in Fig. 3) that are deployed in the user interface 305 upon the selection of an associated object.

Alternatively, in addition to modifying an existing contextual user interface, a new contextual user interface may be added to the host application. For example, if a given add-in application adds one or more functionalities for use on a selected object, an XML file 340 and an associated XML schema may be exposed for creating a new contextual user interface for the associated add-in functions that will be launched in association with a selected object. According to this embodiment, a start-from-

scratch XML file 340 may be provided as described above to allow the creation and deployment of a new contextual user interface 420.

Fig. 5 illustrates a computer screen display showing an example gallery or collection of selectable controls 430 that may be modified according to embodiments of the present invention. As should be understood, the gallery of controls user interface 430 illustrated in Fig. 5 is for purposes of example only and is not restrictive of the different user interface layouts and different user interface content that may be applied to and included in a gallery of controls user interface, as described herein. Referring now to Fig. 5, a pop-out gallery of images is illustrated adjacent to the context menu 420. As should be understood such a gallery of controls may be deployed in other configurations. For example, a gallery of controls according to embodiments of the invention may be deployed in-line in the user interface 200, or the gallery may be deployed as a drop-down user interface under a selected button or control in the user interface 200. According to embodiments each control 435, 440, 445 contained in the gallery of controls represents one or more functionalities, for example, formatting functionalities that will be applied to a selected object if a given control in the gallery is selected.

A given add-in application may add functionality to the host application for providing additional functionalities (e.g., formatting properties) to a selected object and for which one or more additional controls may be desired in a host application gallery of controls. For example, referring to the example gallery of controls illustrated in Fig. 5, a third party add-in application may provide a formatting setting that automatically places the example picture object in the upper left hand corner of the example document and simultaneously applies a different formatting to the text contained in the document in which the picture object is included. If desired, the add-in application developer may modify the host application gallery of controls user interface 430 to provide an additional control showing the application of the add-in formatting properties to the selected document and picture object. As described above with reference to Figs. 3 and 4, the add-in application author may specify the location in the gallery of controls at which the new control should be placed. For example, the add-in developer may specify that the new control should be inserted after the "Top Left" control 440.

In addition to modifying an existing gallery of controls user interface, a new gallery of controls user interface 430 may be added to the host application. For example, if a given add-in application adds one or more functionalities for use on a selected object, an XML file 340 and an associated XML schema may be exposed for creating a new gallery of controls user interface for the associated add-in functions that will be launched in association with a selected object. According to this embodiment, a start-from-scratch XML file 340 may be provided as described above to allow the creation and deployment of a new contextual user interface 420.

After user interface components are added to the host application user interface as described above with reference to Figs. 3, 4, and 5, according to one embodiment of the present invention, the host application may automatically scale the sizes of the added user interface components as display space for the modified user interface is reduced or enlarged. For example, if a logical grouping of buttons or controls is added to a host application user interface, and the logical grouping of buttons or controls contains three large buttons associated with add-in functionality, the host application may automatically scale the displayed buttons if the window size containing the user interface is reduced. For example, if the window size containing the user interface is reduced such that the three large example buttons may no longer be displayed without crowding other user interface components, the three large buttons may be replaced with three smaller versions of the three large buttons. If the window size is further reduced, the smaller versions of the three buttons may be removed altogether, and a small text identification for the three functionalities associated with the three buttons may be utilized in place of the three large buttons.

Customized user interface components may be refreshed automatically by the host application when conditions affecting the customized user interface components change. For example, if a customized user interface component is added to a host application user interface 305 that provides a logo or picture associated with a given type of information (first data), the logo or picture may be disabled and automatically refreshed when associated information or data changes. For example, an add-in weather application may provide a button or control to the host application user interface which when selected provides a weather forecast for a specified area. The button or control in the user interface may be decorated with a picture associated with the current weather, for example, a sunny picture for sunny weather, a cloudy picture

for cloudy weather, and so on. According to this embodiment of the present invention, if the current weather condition changes such that the picture or logo provided for the added button or control is no longer applicable, the current picture or logo may be automatically disabled, and the button or control may be refreshed with a different picture or logo that is applicable to the current information, for example, the current weather conditions.

According to embodiments, the host application keeps track of the identity of each added or modified user interface component relative to the software application add-in responsible for the added or modified user interface component. According to one embodiment, upon hovering over or focusing on an added user interface component, for example, hovering a mouse pointer over an added user interface component, a tool tip or other dialog may be presented to identify the software application add-in responsible for the added or modified user interface component and for directing the user to help content available for describing or providing other information about the software application add-in associated with the added or modified user interface component.

As described herein, software application user interface programming is exposed to allow modification of existing application user interfaces to include adding, removing, disabling, enabling and repurposing new or existing user interface components associated with add-in applications. It will be apparent to those skilled in the art that various modifications or variations may be made in the present invention without departing from the scope or spirit of the invention. Other embodiments of the present invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein.

WE CLAIM:

1. A method for modifying a graphical user interface, comprising:

providing a ribbon-shaped graphical user interface comprising a first tab, the first tab comprising a first logical group of controls, the first logical group of controls comprising one or more selectable controls for selecting one or more functionalities of a host software application, the ribbon-shaped graphical user interface including a gallery that provides one or more selectable controls, each of the selectable controls in the gallery comprising an image that represents a functionality that is applied if the selectable control is selected;

providing an Extensible Markup Language (XML) representation of the ribbon-shaped graphical user interface, the XML representation including a first XML element corresponding to the first tab of the ribbon-shaped graphical user interface, a second XML element corresponding to the first logical group of controls;

receiving an add-in application for providing one or more additional functionalities to the host software application;

receiving a modification to the XML representation, the modified XML representation including a first add-in XML element corresponding to an add-in control that provides a user with access to functionality provided by the add-in application; and

rendering the ribbon-shaped graphical user interface by the host application such that the gallery in the ribbon-shaped graphical user interface is modified to display the add-in control, the add-in control comprising an image that represents a functionality applied if the add-in control is selected.

2. The method of claim 1,

wherein the modified ribbon-shaped graphical user interface comprises one or more additional selectable controls associated with the one or more additional functionalities provided by the add-in application; and

wherein rendering the modified ribbon-shaped graphical user interface comprises: rendering the modified ribbon-shaped graphical user interface such that the modified ribbon-shaped graphical user interface displays the one or more

additional selectable controls within a second tab, the second tab displayed within the ribbon-shaped graphical user interface.

3. The method of claim 1, wherein the modified XML file specifies a change to a behavior of a selectable control contained in the ribbon-shaped graphical user interface.

4. The method of claim 1,

wherein one or more selectable controls contained in the ribbon-shaped graphical user interface are enabled; and

wherein the one or more selectable controls are disabled in the modified ribbon-shaped graphical user interface.

5. The method of claim 1, further comprising: upon receiving a focus on a selectable control contained in the modified ribbon-shaped graphical user interface, displaying an identification of the add-in application.

6. The method of claim 1, wherein a XML schema restricts one or more selectable controls contained in the ribbon-shaped graphical user interface from modification.

7. The method of claim 1,

wherein the ribbon-shaped graphical user interface includes a contextual user interface for displaying one or more selectable controls applicable to a selected object; and

wherein the modified ribbon-shaped graphical user interface comprises one or more additional selectable controls to the contextual user interface.

8. The method of claim 1, further comprising: after un-installation of the add-in application from the host application, rendering the ribbon-shaped graphical user interface instead of the modified ribbon-shaped graphical user interface.

9. The method of claim 1, further comprising: after closing a document-based solution that modified the ribbon-shaped graphical user interface, rendering the ribbon-shaped graphical user interface instead of the modified ribbon-shaped graphical user interface.

10. The method of claim 1, further comprising automatically scaling any user interface components added in the modified ribbon-shaped graphical user interface such that the any user interface components added in the modified ribbon-shaped graphical user interface are scaled to fit a display space in which the modified ribbon-shaped graphical user interface is displayed.

11. The method of claim 1,

wherein the modified ribbon-shaped graphical user interface includes a selectable control that was not in the ribbon-shaped graphical user interface, the selectable control associated with an additional functionality provided by the add-in application, wherein the selectable control is associated with a first data; and

upon receiving a change to the first data, refreshing the modified ribbon-shaped graphical user interface to automatically update the selectable control.

12. A computer storage medium containing computer executable instructions, which when executed by a computer, cause the computer to:

receive an add-in application for providing one or more additional functionalities to a host software application on the computer, wherein an initial XML file represents a ribbon-shaped initial graphical user interface in a graphical user interface of the host software application, the initial graphical user interface including a first tab, the first tab comprising a gallery of selectable controls, each of the controls in the gallery comprising an image that represents a functionality that is applied if the selectable control is selected;

receiving an amended XML file, the amended XML file representing a ribbon-shaped modified graphical user interface, wherein in addition to the controls in the gallery in the initial graphical user interface, the gallery in the modified graphical user interface includes one or more new selectable controls that provide access to the one or more additional functionalities provided by the add-in application, each of the new

selectable controls comprising an image that represents a functionality that is applied if the new selectable control are selected; and

rendering, by the host software application, the modified graphical user interface such that the graphical user interface displays the modified graphical user interface.

13. The computer storage medium of claim 12, further comprising: upon receiving a focus on one of the new selectable controls, displaying in the modified graphical user interface identification information about modifications to the initial graphical user interface.

14. The computer storage medium of claim 12,

wherein the initial graphical user interface includes a contextual user interface for displaying one or more selectable controls applicable to a selected object; and

wherein the modified graphical user interface includes one or more additional selectable controls in the contextual user interface.

15. The computer storage medium of claim 12,

wherein the gallery in the initial graphical user interface provides one or more selectable controls for providing one or more functionalities to a selected object.

16. A method for modifying a graphical user interface, the method comprising:

receiving, at a computing device, an add-in application for providing one or more additional functionalities to a host software application on the computing device,

wherein an initial XML file represents a ribbon-shaped initial graphical user interface of the host software application, the initial graphical user interface including a gallery that provides one or more selectable controls, each of the selectable controls in the gallery comprising an image that represents a functionality that is applied if the selectable control is selected,

receiving, at the computing device, an amended XML file, the amended XML file representing a ribbon-shaped modified graphical user interface of the host software application,

wherein in the modified graphical user interface, the gallery includes one or more new selectable controls that provide access to the one or more additional functionalities provided by the add-in application, the new selectable controls comprising an image that represents a functionality that is applied if the new selectable control is selected.

17. The method of claim 16, further comprising: receiving, at the computing device, a selection of the gallery.

18. The method of claim 16, further comprising:

in response to closing a document-based solution that modified the initial graphical user interface, rendering the initial graphical user interface instead of the modified graphical user interface.

19. The method of claim 16, further comprising:

disabling one or more selectable controls in the modified graphical user interface.

20. A method for modifying a graphical user interface, comprising:

providing a ribbon-shaped graphical user interface comprising a first tab, the first tab comprising a first logical group of controls, the first logical group of controls comprising one or more selectable controls for selecting one or more functionalities of a host software application;

providing an Extensible Markup Language (XML) representation of the ribbon-shaped graphical user interface, the XML representation including a first XML element corresponding to the first tab of the ribbon-shaped graphical user interface and a second XML element corresponding to the first logical group of controls;

receiving an add-in application for providing one or more additional functionalities to the host software application;

receiving a modification to the XML representation, the modified XML representation including a first add-in XML element corresponding to an add-in control that provides a user with access to functionality provided by the add-in application; and

rendering ribbon-shaped graphical user interface by the host application such that the ribbon-shaped graphical user interface is modified to display the add-in control;

after un-installation of the add-in application from the host application, rendering the ribbon-shaped graphical user interface instead of the modified ribbon-shaped graphical user interface.

21. The method of claim 20,

wherein the modified ribbon-shaped graphical user interface comprises one or more additional selectable controls associated with the one or more additional functionalities provided by the add-in application; and

wherein rendering the modified graphical user interface comprises: rendering the modified ribbon-shaped graphical user interface such that the modified ribbon-shaped graphical user interface displays the one or more additional selectable controls within a second tab, the second tab displayed within the ribbon-shaped graphical user interface.

22. The method of claim 20, further comprising: upon receiving a focus on a selectable control contained in the modified ribbon-shaped graphical user interface, displaying an identification of the add-in application.

23. The method of claim 20,

wherein the ribbon-shaped graphical user interface includes a gallery of selectable controls for providing one or more selectable controls for providing one or more functionalities to a selected object; and

wherein the modified ribbon-shaped graphical user interface comprises one or more additional selectable controls to the gallery of selectable controls user interface.

24. The method of claim 20, further comprising: after closing a document-based solution that modified the ribbon-shaped graphical user interface, rendering the ribbon-based graphical user interface instead of the modified ribbon-shaped graphical user interface.