US 20150095812A1

(19) **United States**

(12) **Patent Application Publication**

Birck et al.

(10) **Pub. No.: US 2015/0095812 A1**

(43) **Pub. Date: Apr. 2, 2015**

(54) **EXTENSIBLE AND CONTEXT-AWARE COMMANDING INFRASTRUCTURE**

(71) Applicant: **Microsoft Corporation**, Redmond, WA (US)

(72) Inventors: **Andrew Birck**, Seattle, WA (US); **Brad Olenick**, Redmond, WA (US); **Leon Ezequiel Welicki**, Issaquah, WA (US); **Nafisa Bhojawala**, Seattle, WA (US); **Stephen Michael Danton**, Seattle, WA (US); **Jonathan Lucero**, Bellevue, WA (US); **Dina-Marie Ledonna Supino**, Seattle, WA (US); **Jesse David Francisco**, Lake Stevens, WA (US); **Vishal R. Joshi**, Redmond, WA (US); **Karandeep Singh Anand**, Redmond, WA (US); **William J. Staples**, Duvall, WA (US); **Madhur Joshi**, Kirkland, WA (US); **Julio O. Casal**, Redmond, WA (US); **Jonah Bush Sterling**, Seattle, WA (US)

(21) Appl. No.: **14/231,873**

(22) Filed: **Apr. 1, 2014**

Related U.S. Application Data

(60) Provisional application No. 61/905,128, filed on Nov. 15, 2013, provisional application No. 61/884,743, filed on Sep. 30, 2013, provisional application No. 61/905,111, filed on Nov. 15, 2013, provisional application No. 61/905,243, filed on Nov. 17, 2013, provisional application No. 61/905,114, filed on Nov. 15, 2013, provisional application No. 61/905,116, filed on Nov. 15, 2013, provisional application No. 61/905,129, filed on Nov. 15, 2013, provisional application No. 61/905,105, filed on Nov. 15, 2013, provisional application No. 61/905,247, filed on Nov. 17, 2013, provisional application No. 61/905,101, filed on Nov. 15, 2013, provisional application No. 61/905,119, filed on Nov. 15, 2013.

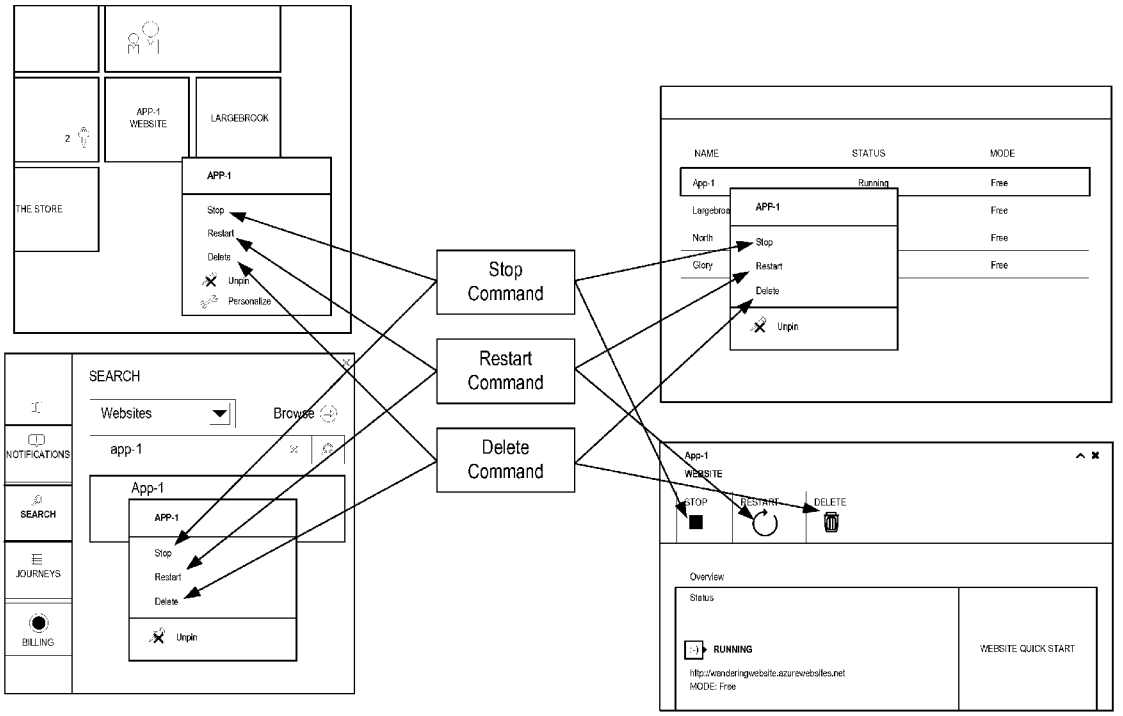
Publication Classification

(51) **Int. Cl.**
G06F 3/0484 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 3/0484*** (2013.01)
USPC **715/762**

(57) **ABSTRACT**

Computing systems in which multiple non-context-sensitive or core commands may be initiated from each of a number of different user interface contexts. There are also multiple context-sensitive mechanism for visualizing the commands depending on which of the multiple possible user interface contexts that the commands appear. At least some embodiments described herein also related to the presentation of dialogs at various stages of the command lifecycle without the system needing to know the underlying operations of the command, and allowing the developer to specify when dialogs are to appear in that lifecycle.



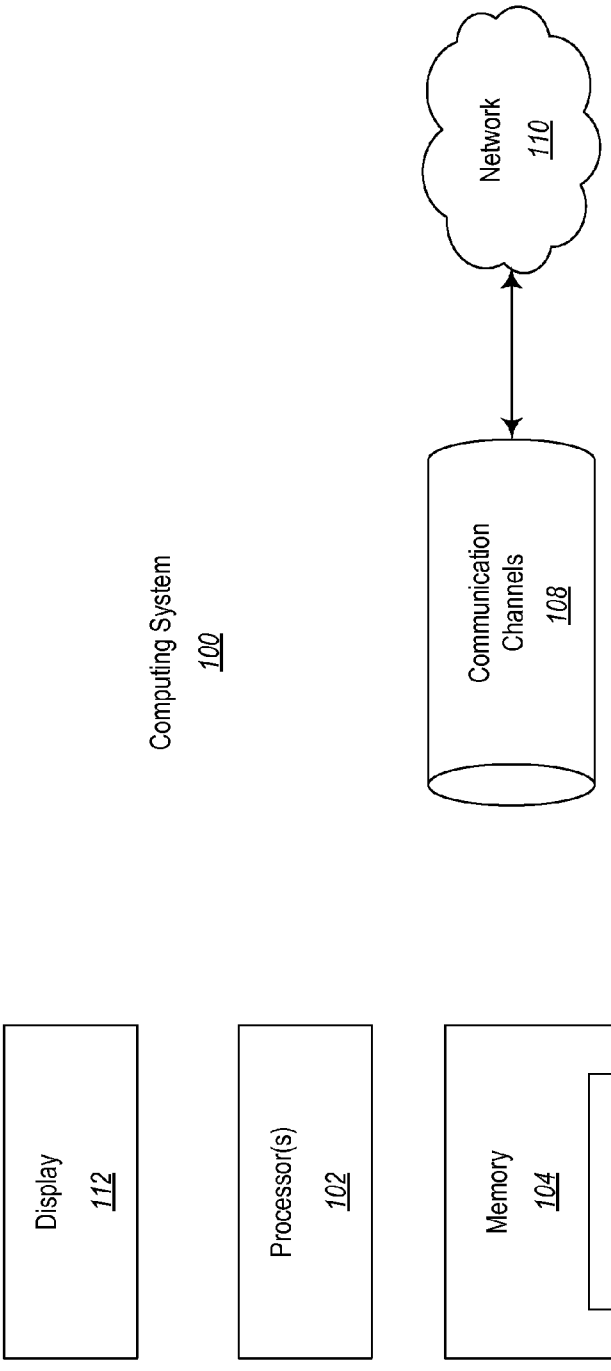

Network
110
Communication Channels
108
Computing System
100
Display
112
Processor(s)
102
Memory
104
Volatile
Non-Volatile

Figure 1

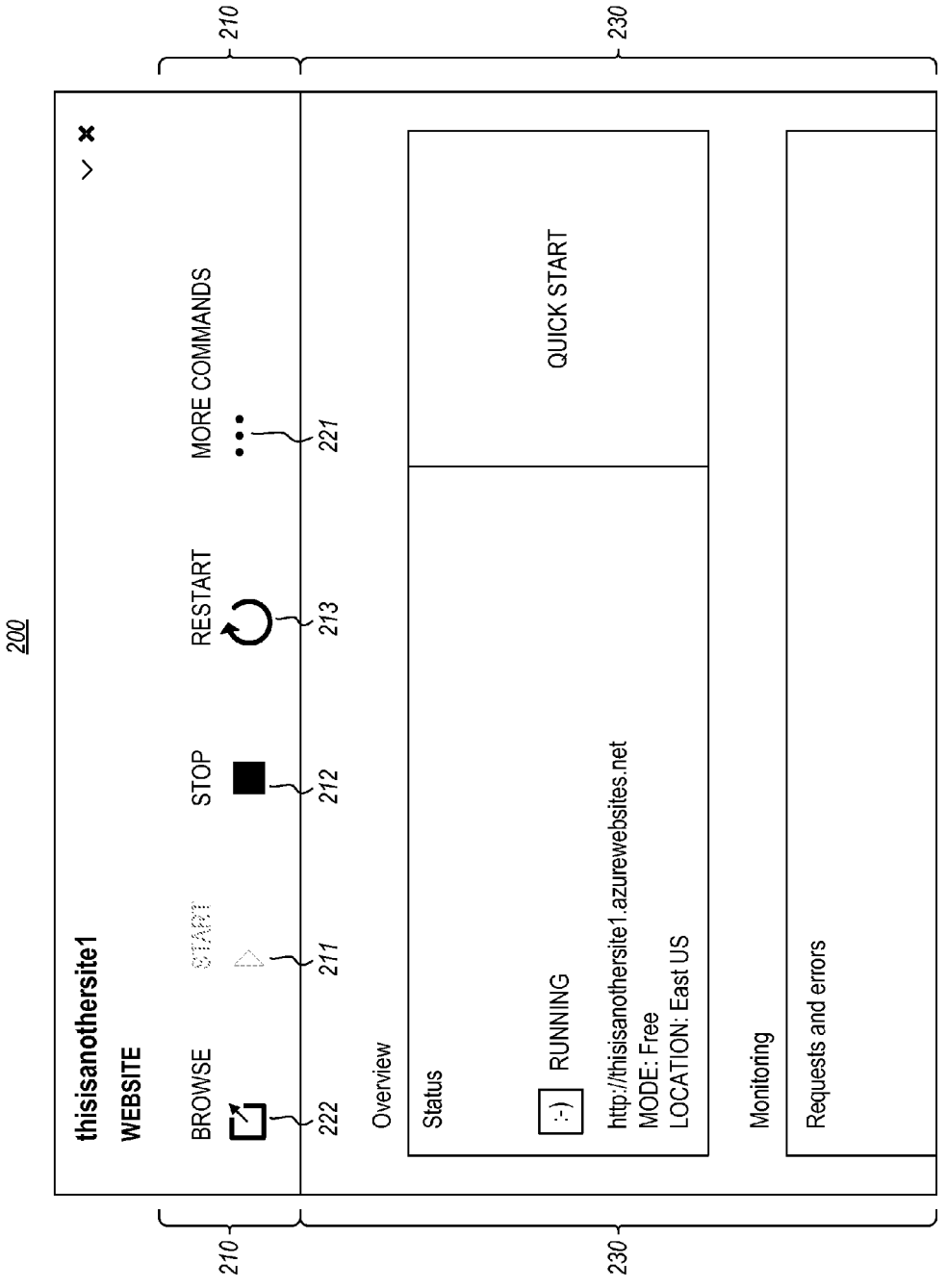

200
210
230
thisisanothersite1
WEBSITE
BROWSE
START
STOP
RESTART
MORE COMMANDS
222
211
212
213
221
Overview
Status
QUICK START
:-)
RUNNING
http://thisisanothersite1.azurewebsites.net
MODE: Free
LOCATION: East US
Monitoring
Requests and errors

Figure 2

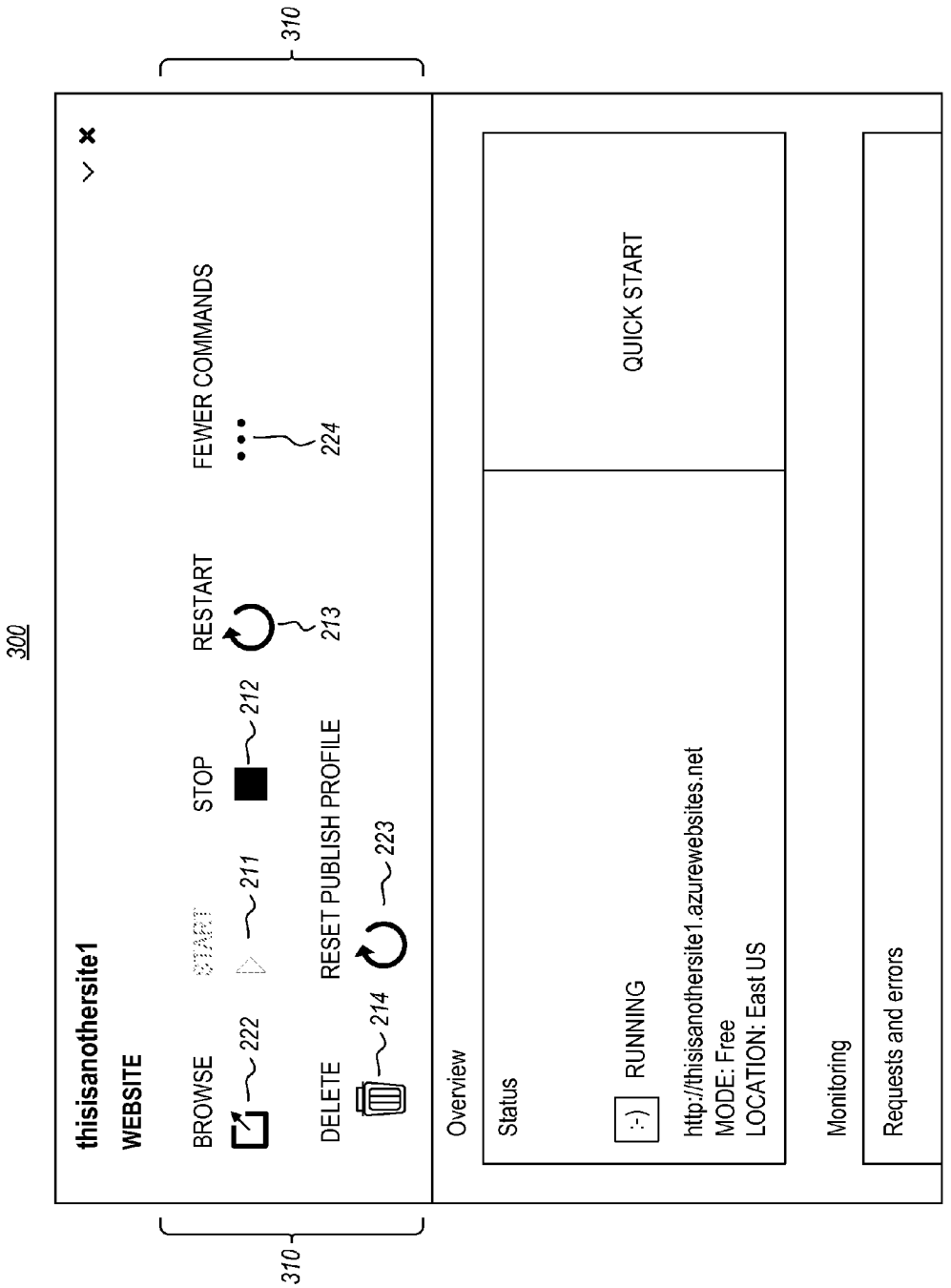

300
thisisanothersite1
WEBSITE
BROWSE
222
START
211
STOP
212
RESTART
213
FEWER COMMANDS
224
DELETE
214
RESET PUBLISH PROFILE
223
310
310
Overview
Status
:-) RUNNING
http://thisisanothersite1.azurewebsites.net
MODE: Free
LOCATION: East US
QUICK START
Monitoring
Requests and errors

Figure 3

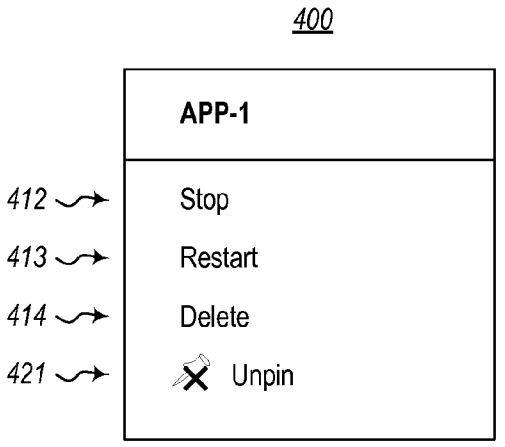


Figure 4

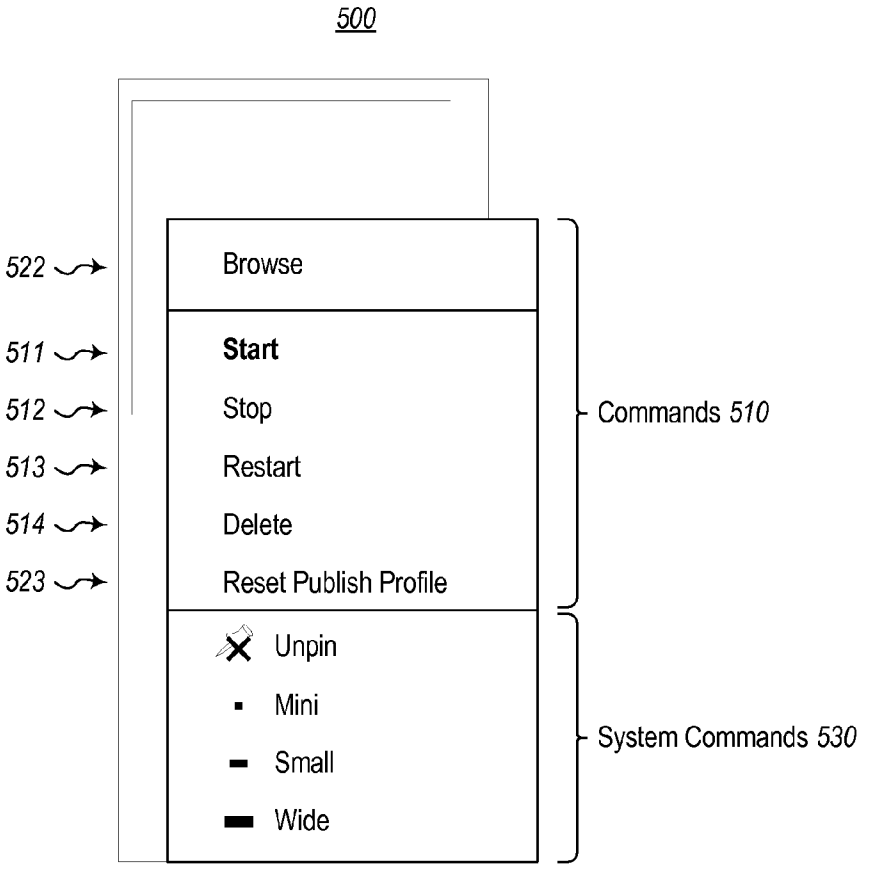


Figure 5

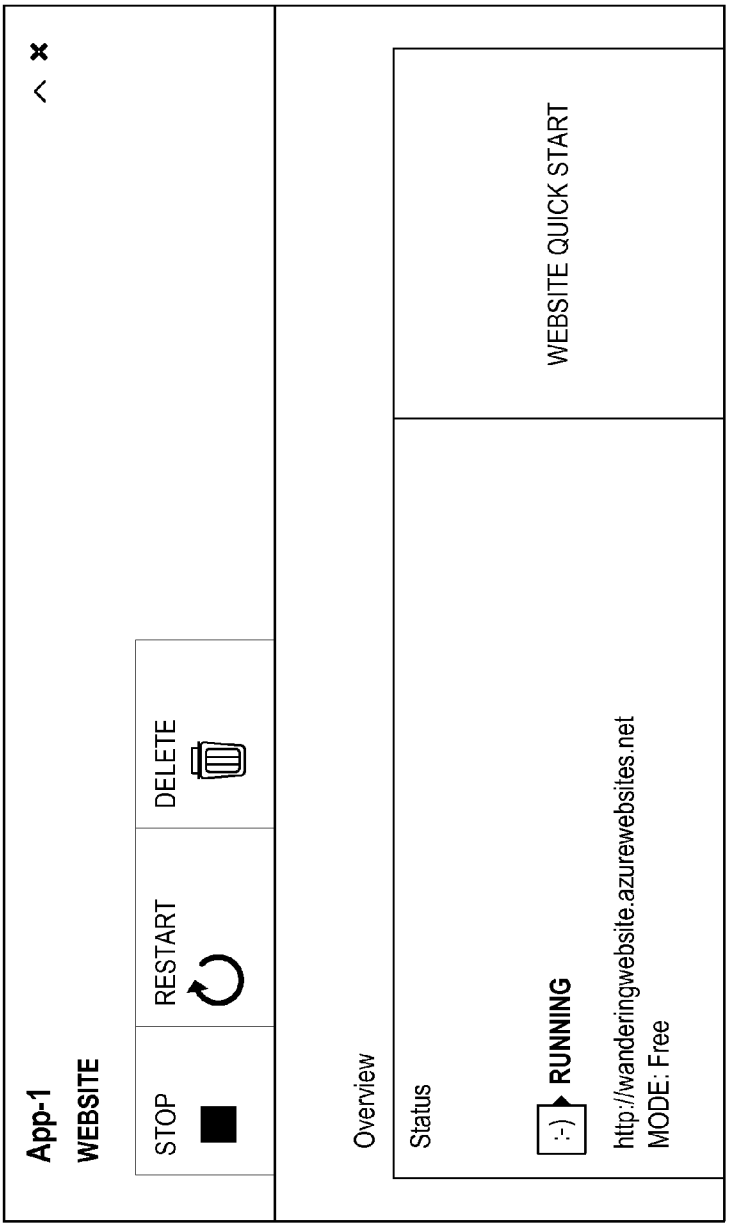

App-1
WEBSITE
STOP
RESTART
DELETE
Overview
Status
:-) RUNNING
http://wanderingwebsite.azurewebsites.net
MODE: Free
WEBSITE QUICK START

Figure 6A

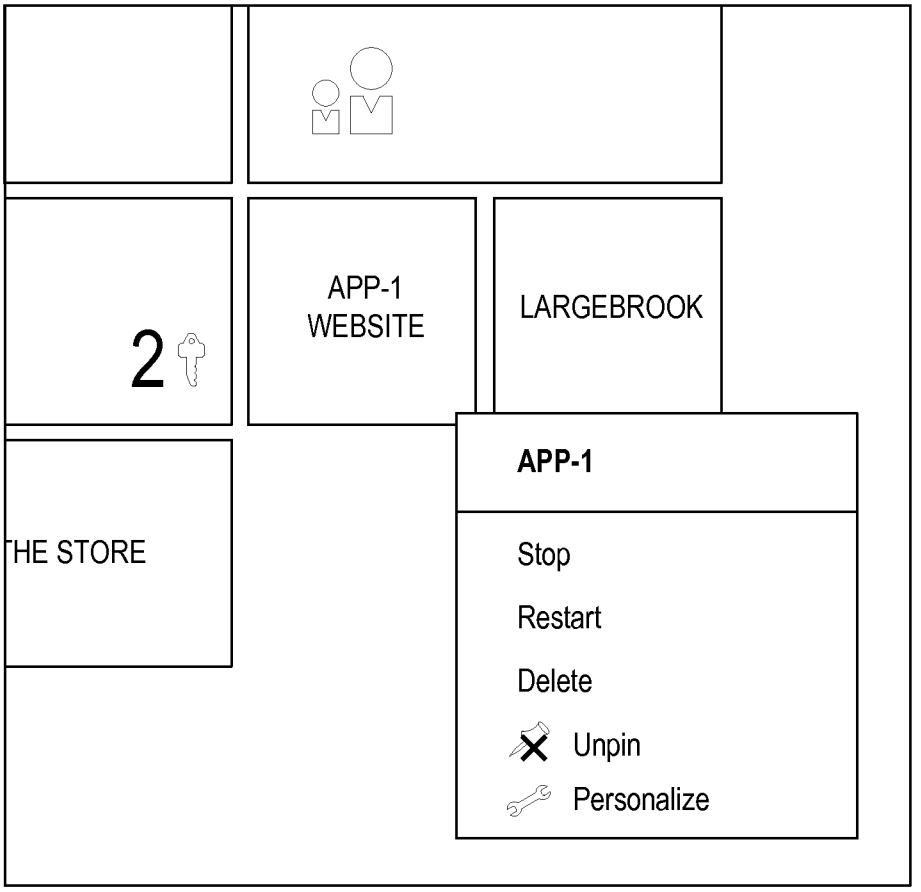

2
APP-1
WEBSITE
LARGEBROOK
THE STORE
APP-1
Stop
Restart
Delete
Unpin
Personalize

Figure 6B

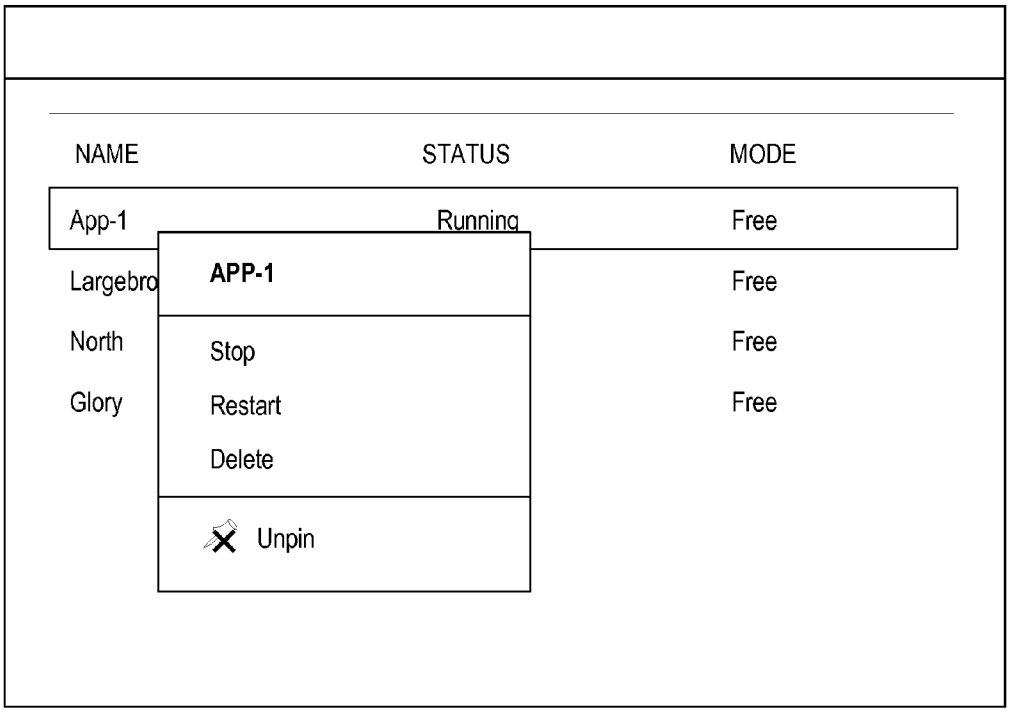


Figure 6C

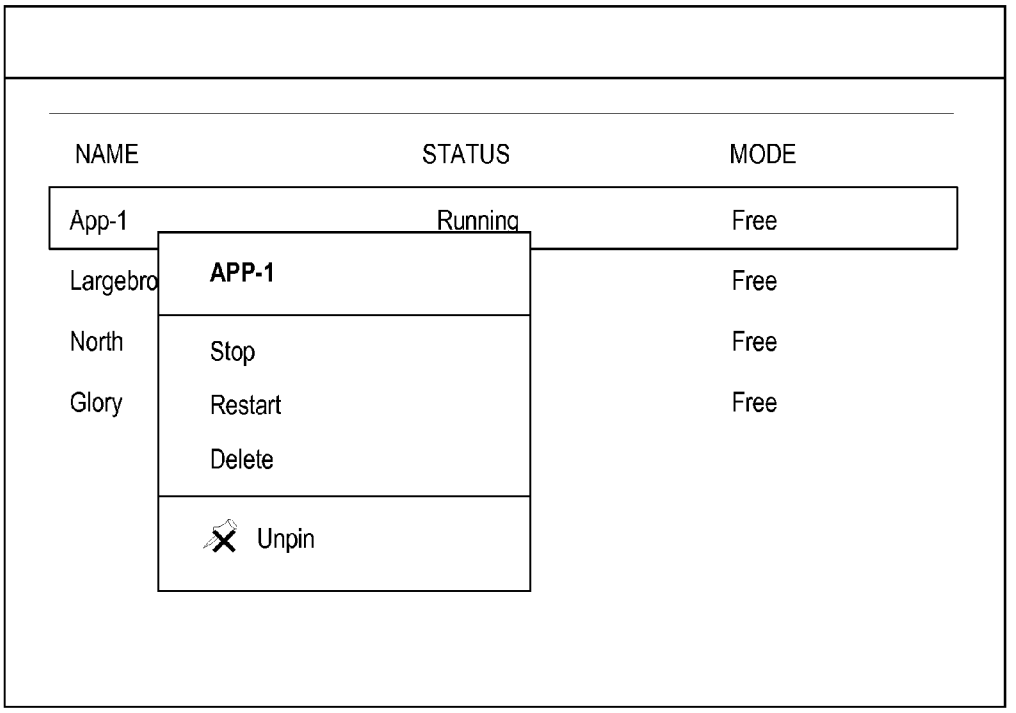


Figure 6D

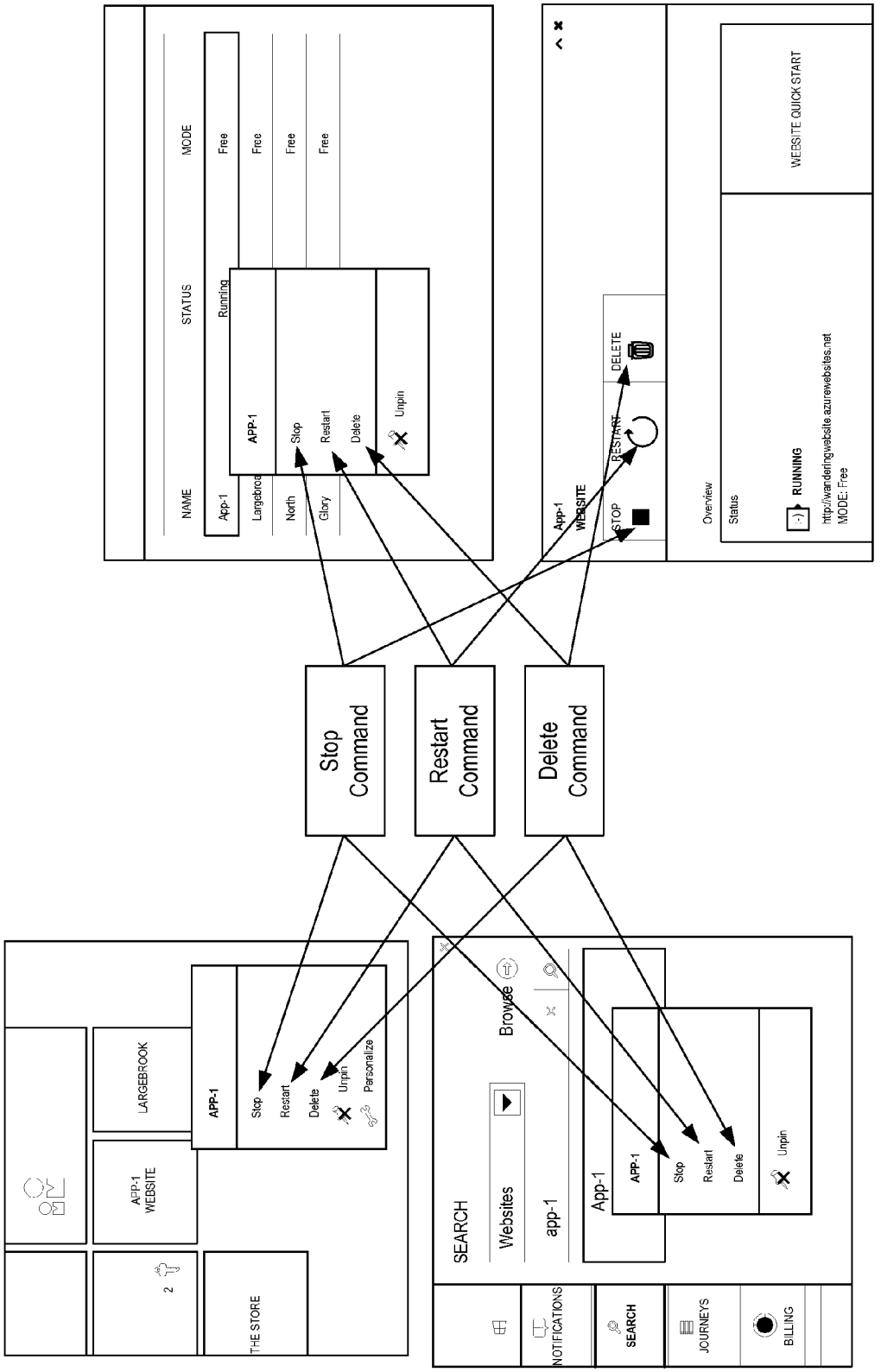

NAME
STATUS
MODE
App-1
Running
Free
APP-1
Largebrook
North
Glory
Stop
Restart
Delete
Unpin
Overview
Status
RUNNING
http://wranderingwebsite.azurewebsites.net
MODE: Free
WEBSITE QUICK START
DELETE
RESTART
STOP
Stop Command
Restart Command
Delete Command
LARGEBROOK
APP-1 WEBSITE
THE STORE
Personalize
SEARCH
Websites
app-1
Browse
NOTIFICATIONS
JOURNEYS
BILLING

Figure 7

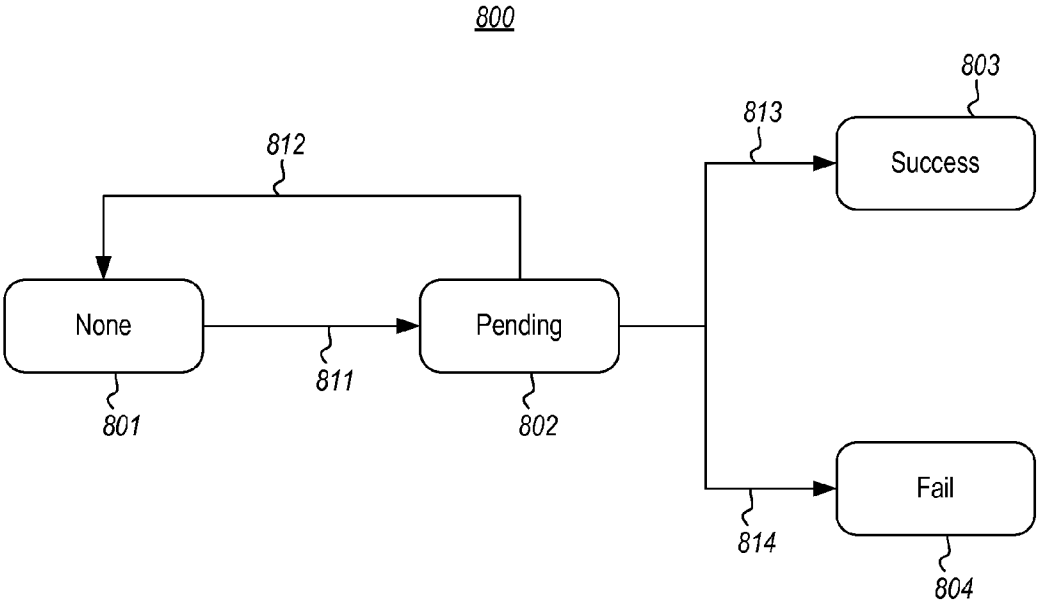


Figure 8

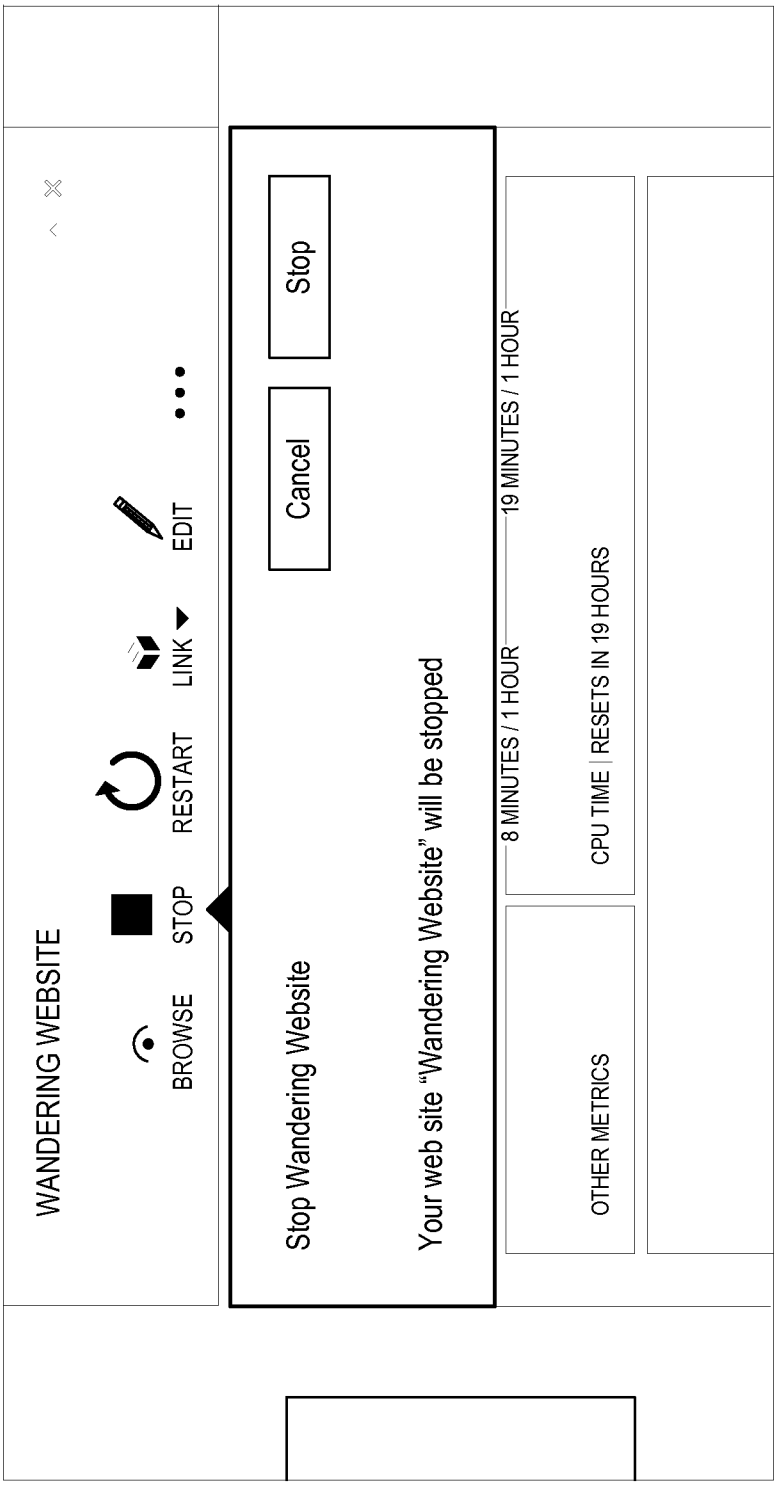

WANDERING WEBSITE
BROWSE
STOP
RESTART
LINK
EDIT
Stop Wandering Website
Cancel
Stop
Your web site "Wandering Website" will be stopped
8 MINUTES / 1 HOUR
19 MINUTES / 1 HOUR
CPU TIME | RESETS IN 19 HOURS
OTHER METRICS

Figure 9

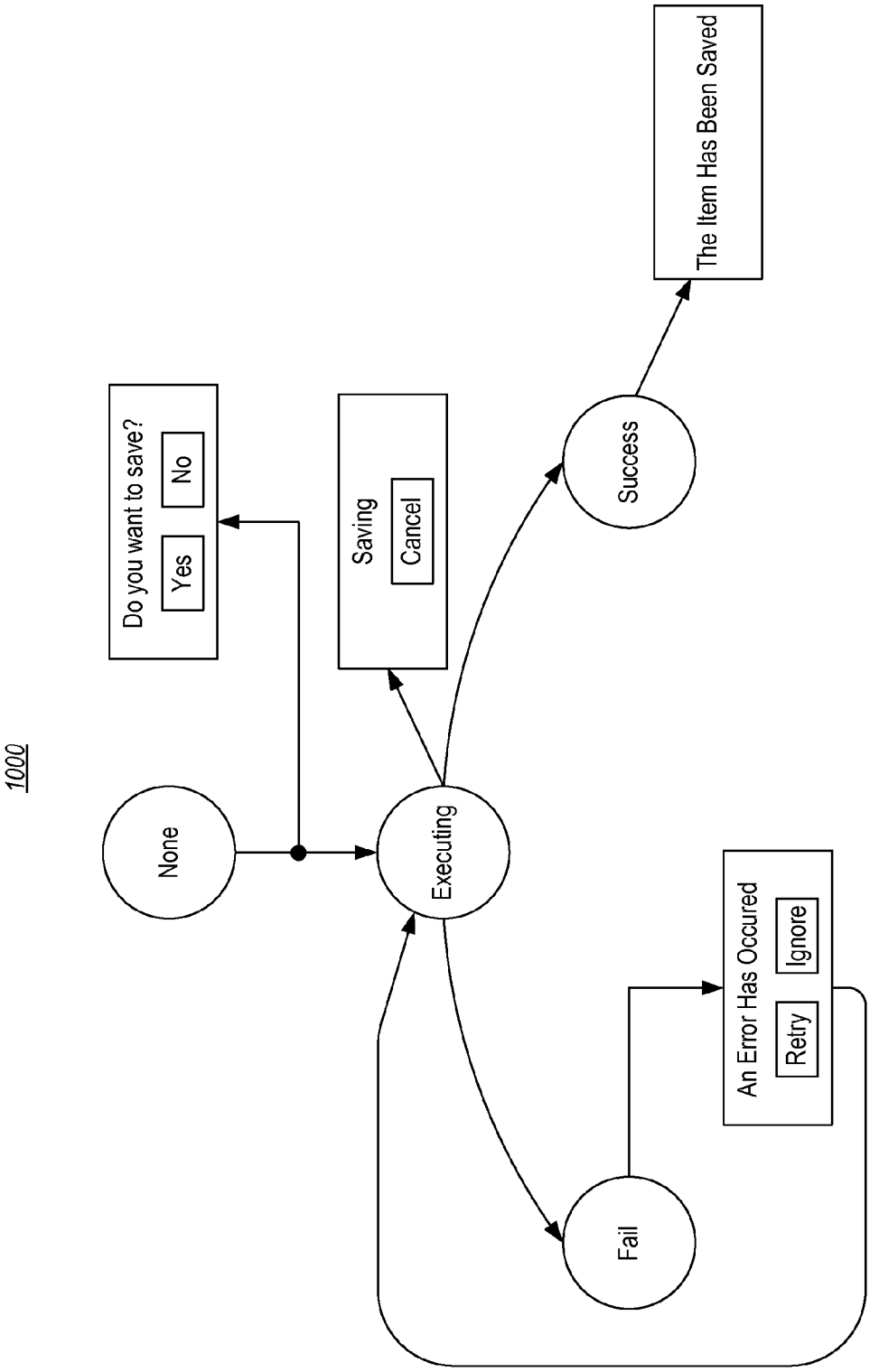

1000
None
Executing
Do you want to save?
Yes
No
Saving
Cancel
Success
The Item Has Been Saved
Fail
An Error Has Occured
Retry
Ignore

Figure 10

EXTENSIBLE AND CONTEXT-AWARE COMMANDING INFRASTRUCTURE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of each of the following provisional patent applications, and each of the following provisional patent applications are incorporated herein by reference in their entirety:

[0002] 1. U.S. Provisional Application Ser. No. 61/905,114, filed Nov. 15, 2013;

[0003] 2. U.S. Provisional Application Ser. No. 61/884,743, filed Sep. 30, 2013;

[0004] 3. U.S. Provisional Application Ser. No. 61/905,111, filed Nov. 15, 2013;

[0005] 4. U.S. Provisional Application Ser. No. 61/905,243, filed Nov. 17, 2013;

[0006] 5. U.S. Provisional Application Ser. No. 61/905,116, filed Nov. 15, 2013;

[0007] 6. U.S. Provisional Application Ser. No. 61/905,129, filed Nov. 15, 2013;

[0008] 7. U.S. Provisional Application Ser. No. 61/905,105, filed Nov. 15, 2013;

[0009] 8. U.S. Provisional Application Ser. No. 61/905,247, filed Nov. 17, 2013;

[0010] 9. U.S. Provisional Application Ser. No. 61/905,101, filed Nov. 15, 2013;

[0011] 10. U.S. Provisional Application Ser. No. 61/905,128, filed Nov. 15, 2013; and

[0012] 11. U.S. Provisional Application Ser. No. 61/905,119, filed Nov. 15, 2013.

BACKGROUND

[0013] Computing systems and networks have transformed the way we work, play, and communicate. Computing systems obtain there functionality by executing commands on computing resources accessible to the computing system. Commands might be, for instance, initiated by a user. In that case, the user interfaces with a visualization of the command, thereby causing corresponding operations on the computing asset. During various stages of the lifecycle of a command, the user may be presented with dialogs that ask for confirmation, inform of success or failure, or inform of progress of the command.

BRIEF SUMMARY

[0014] At least some embodiments described herein relate to computing systems in which multiple non-context-sensitive or core commands may be initiated from each of a number of different user interface contexts. There are also multiple context-sensitive mechanism for visualizing the commands depending on which of the multiple possible user interface contexts that the commands appear. At least some embodiments described herein also related to the presentation of dialogs at various stages of the command lifecycle without the system needing to know the underlying operations of the command, and allowing the developer to specify when dialogs are to appear in that lifecycle.

[0015] 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 to be used as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0017] FIG. **1** abstractly illustrates an example computing system in which the principles described herein may be employed;

[0018] FIG. **2** illustrates a user interface element in the form of a blade, and in which commands are displayed in a command bar;

[0019] FIG. **3** illustrates a user interface that represents modifications to the user interface that would occur if the user selects the command bar expansion control of FIG. **2**;

[0020] FIG. **4** illustrates an example context menu that represents another example of a context-sensitive mechanism for visualizing controls;

[0021] FIG. **5** illustrates an extended example context menu that represents another example of a context-sensitive mechanism for visualizing controls;

[0022] FIGS. **6**A through **6**D illustrates various visualizations of the same commands across different user interface contexts;

[0023] FIG. **7** illustrates that the commands within FIGS. **6**A through **6**D are indeed the same;

[0024] Even though the user experience for displaying the commands and the context where the command is displayed may be different, the actual command is the same as illustrated in FIG. **7**;

[0025] FIG. **8** illustrates a life-cycle that the system may be aware of for all commands, whether built-in or extrinsic;

[0026] FIG. **9** illustrates an example of a dialog that may appear upon initiating a stop website command; and

[0027] FIG. **10** illustrates that each stage in a life cycle of a command can surface a different dialog, with the application developer indicating whether the corresponding dialog is to appear at each stage in the state machine.

DETAILED DESCRIPTION

[0028] Commanding is a common way of describing behavior in a system, whether distributed or otherwise. Each command represents a unit of functionality that can be applicable to an asset within the system, to the system itself, or to any arbitrary artifact. Commands can be provided by the system (i.e., built-in commands) or by other parties (extrinsic commands).

[0029] In accordance with the principles described herein, commands are provided consistently in an entire system, even though the system itself may be operating a number of different applications composed by entirely different parties. Furthermore, the embodiments described herein help security by running commands in the right isolation mode, such that harmful (but not necessarily malicious) code does not com-

promise the system. Preferably, the command should not block the user interface so they should run asynchronously. As far as the user experience, the embodiments described herein allow commands to be surfaced following the same patterns (e.g. command bar; context menu; etc.) (i.e., also referred to herein as a context-sensitive mechanism for visualization) and provide interactivity options to the users (e.g. dialogs) so they can participate in the operation and also understand the operation's status and result.

[0030] The principles described herein may be implemented using a computing system. For instance, the users may be engaging with the system using a client computing system. The executable logic supporting the system and providing visualizations thereon may also be performed using a computing system. The computing system may even be distributed. Accordingly, a brief description of a computing system will now be provided.

[0031] Computing systems are now increasingly taking a wide variety of forms. Computing systems may, for example, be handheld devices, appliances, laptop computers, desktop computers, mainframes, distributed computing systems, or even devices that have not conventionally been considered a computing system. In this description and in the claims, the term "computing system" is defined broadly as including any device or system (or combination thereof) that includes at least one physical and tangible processor, and a physical and tangible memory capable of having thereon computer-executable instructions that may be executed by the processor. The memory may take any form and may depend on the nature and form of the computing system. A computing system may be distributed over a network environment and may include multiple constituent computing systems. An example computing system is illustrated in FIG. **1**.

[0032] As illustrated in FIG. **1**, in its most basic configuration, a computing system **100** typically includes at least one processing unit **102** and memory **104**. The memory **104** may be physical system memory, which may be volatile, non-volatile, or some combination of the two. The term "memory" may also be used herein to refer to non-volatile mass storage such as physical storage media. If the computing system is distributed, the processing, memory and/or storage capability may be distributed as well. As used herein, the term "executable module" or "executable component" can refer to software objects, routines, or methods that may be executed on the computing system. The different components, modules, engines, and services described herein may be implemented as objects or processes that execute on the computing system (e.g., as separate threads).

[0033] In the description that follows, embodiments are described with reference to acts that are performed by one or more computing systems. If such acts are implemented in software, one or more processors of the associated computing system that performs the act direct the operation of the computing system in response to having executed computer-executable instructions. For example, such computer-executable instructions may be embodied on one or more computer-readable media that form a computer program product. An example of such an operation involves the manipulation of data. The computer-executable instructions (and the manipulated data) may be stored in the memory **104** of the computing system **100**. Computing system **100** may also contain communication channels **108** that allow the computing system **100** to communicate with other message processors over, for example, network **110**.

[0034] The computing system **100** also includes a display **112** on which a user interface, such as the user interfaces described herein, may be rendered. Such user interfaces may be generated in computer hardware or other computer-represented form prior to rendering. The presentation and/or rendering of such user interfaces may be performed by the computing system **100** by having the processing unit(s) **102** execute one or more computer-executable instructions that are embodied on one or more computer-readable media. Such computer-readable media may form all or a part of a computer program product.

[0035] Embodiments described herein may comprise or utilize a special purpose or general-purpose computer including computer hardware, such as, for example, one or more processors and system memory, as discussed in greater detail below. Embodiments described herein also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer system. Computer-readable media that store computer-executable instructions are physical storage media. Computer-readable media that carry computer-executable instructions are transmission media. Thus, by way of example, and not limitation, embodiments of the invention can comprise at least two distinctly different kinds of computer-readable media: computer storage media and transmission media.

[0036] Computer storage media includes RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other tangible medium which can be used to store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer.

[0037] A "network" is defined as one or more data links that enable the transport of electronic data between computer systems and/or modules and/or other electronic devices. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a transmission medium. Transmissions media can include a network and/or data links which can be used to carry or desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. Combinations of the above should also be included within the scope of computer-readable media.

[0038] Further, upon reaching various computer system components, program code means in the form of computer-executable instructions or data structures can be transferred automatically from transmission media to computer storage media (or vice versa). For example, computer-executable instructions or data structures received over a network or data link can be buffered in RAM within a network interface module (e.g., a "NIC"), and then eventually transferred to computer system RAM and/or to less volatile computer storage media at a computer system. Thus, it should be understood that computer storage media can be included in computer system components that also (or even primarily) utilize transmission media.

[0039] Computer-executable instructions comprise, for example, instructions and data which, when executed at a

processor, cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the described features or acts described above. Rather, the described features and acts are disclosed as example forms of implementing the claims.

[0040] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, pagers, routers, switches, and the like. The invention may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. In a distributed system environment, program modules may be located in both local and remote memory storage devices.

[0041] In accordance with principles described herein, a user interface element (often called herein a "part") represents a basic unit of the user interface. Each of at least some of the parts are associated with corresponding controls that the user may interact with to thereby cause the system to execute respective commands. The execution of the command may, for instance, return data to project via the corresponding part. The parts may incorporate extrinsic commands that implement given contracts, and may reason about them.

[0042] In accordance with the principles described herein, commands (also called hereinafter "non-context-sensitive commands" or "core commands") can be associated with resources in the system (such as a website, database, an arbitrary artifact, the system itself, or a piece of the user interface). This association may be persistent, such that when that resource is displayed in different user interface contexts, the non-context-sensitive commands associated with that resource are still available, but displayed using the right context-sensitive mechanism. For instance, the context-sensitive mechanism for visualizing these core commands may be a user experience form factor appropriate for the context in which the part is displayed. Consistency may also be further achieved by having the same context-visualization mechanism used to display the commands of any user interface element that is displayed in a particular user interface context. Commands are thus offered to the user via a well-defined experience that is consistent across the entire system.

[0043] The system provides a set of abstractions through a portal that enable application developers to create commands. A command encapsulates an action in the system. The composition tree describes structure and the commands describe behavior. Commands provide a well-defined surface that the system can reason about to support units of behavior. Commands can be system commands (built-in) and custom commands (provided by the application developer). Commands are offered to the user via a well-defined experience that is consistent across the entire system. This experience is built-in and cannot be redefined by application developers. Application developers can only contribute with new commands, but not with new ways of exposing those commands to the user. Thus, the manner of exposing commands (the command experience) is governed by the system.

[0044] Commands provide application developers and users a consistent model across applications (sometimes referred to as "extensions") compatible with browser capabilities and scalable to all parties to describe behavior in the system. The command may have affinity with portal assets which can make them available everywhere that asset is presented if so desired

[0045] The user interface may be a rich in allowing different user interface elements to be presented in entirely different contexts. For instance, as will be described further below, the user interface might include different contexts such as a favorites area, a blade, and hubs. These represent different places where user interface elements (also called herein "parts") can be displayed. The commands associated with a given asset can be available in all of these different contexts, enabling taking action on a resource in the way that is more convenient (and in addition new commands can be added to accommodate the specifics of each context in case is needed).

[0046] Commands can be associated with different resources in the system. For instance, the resource might be a portion of the user interface itself, such as a part. As another example of a portion of the user interface may be what will be referred to herein as a "blade". A blade is a user interface element that may be placed on a canvas that extends in an extendible direction (such as horizontally). For substantially all of a particular range of the canvas in the dimension of the extendible dimension, the blade may occupy substantially all of the canvas in the dimension perpendicular to the extendible direction of the canvas. The resource might also be associated with an actual asset in the system, such as a website, database, virtual machine, and so forth. Blades or other parts can be also associated with assets creating a transitive relationship between the commands and its container if so desired.

[0047] Commands are visualized through different context-sensitive mechanisms, depending on the user interface context in which the associated part is displayed. Each context-sensitive mechanism supports a particular user experience. For instance, FIG. **2** illustrates a user interface element **200** in the form of a blade. The blade is associated with an asset in the form of a website (called "thisisanothersite1" in the user interface element **200** of FIG. **2**). Commands are presented in a command bar **210** at the top of the blade in this context. Thus, the command bar **210** represents a context-sensitive mechanism for visualizing commands when the commands are displayed in the context of a blade.

[0048] In FIG. **2**, the command bar **210** is illustrated as visualizing three non-context-sensitive commands including the start command **211**, the stop command **212**, and the restart command **213**. The start command **211** is deemphasized as not selectable since the web site has already running as evidenced within the status window **230**. As will be seen from the subsequent windows, the command **211**, **212** and **213** are "non-context-sensitive" in that regardless of the user interface context in which the non-context-sensitive commands are displayed, at least the selectable non-context-sensitive commands (in this case stop command **212** and the restart command **213**) will still be displayed.

[0049] The command bar **210** also includes an overflow control **221** (also called hereinafter a "command bar expansion control **221**") that is presented when there are more commands associated with the blade than the blade can display in the available space. FIG. **3** illustrates a user interface **300** that represents modifications to the user interface **200** that would occur if the user selects the command bar expansion control **221**. Note that the command bar **210** is augmented to be an augmented command bar **310** that shows a second row of commands. For instance, a non-context-sensitive delete command **214** is illustrated in the second row.

[0050] The selectable non-context-sensitive commands **212** through **214** may be presented regardless of where commands appears in the user interface context. For instance, when the user interface element is associated with a resource, the non-context sensitive commands may be basic commands associated with the resource that the user might like to initiate regardless of the user interface context in which the resource is presented. For instance, a user might like to start, stop, restart, or delete a web site from any one of a number of different user interface contexts.

[0051] Referring again to FIGS. **2** and **3**, the command bars **210** and **310** also include context-sensitive commands. For instance, there is an overflow indicator **221** (hereinafter referred to as a "command bar expand command" **221**). Furthermore, there is a browse command **222**, which is related to the underlying asset (e.g., the web site), but which is specific to a presentation in a particular user interface context. For instance, a user might like to browse to the web site when the web site is associated with the blade (since there is more space available to usefully browse), but the user might not be so interested in browsing if they are working in the context of a smaller user interface portion that is associated with that web site. Another context-sensitive command is illustrated as the reset publish profile command **223** in FIG. **3**. FIG. **3** illustrates another context-sensitive command in the form of a command bar collapse control **224**, which when selected returns the user interface **300** to that of the user interface **200** of FIG. **2**.

[0052] In one embodiment, context-sensitive commands including one or more of 1) non-selectable non-context-sensitive commands (such as the start command **211**), 2) commands that are associated with an underlying resource, but which are not to be performed in every user interface context (such as the browse command **222** and the reset publish profile command **223**), and 3) and commands that are associated with the user interface element itself, but not the underlying resource (such as the overflow indicator **221** or command bar expansion command).

[0053] FIGS. **2** and **3** illustrate user interface elements when web commands are associated with a blade, which is one example of a user interface context. However, web commands may be displayed in other user interface contexts. For instance, suppose that the web commands are in a smaller user interface part that is within the favorites area, within an activity pane or grid. In those user interface contexts, the active context sensitive commands may be displayed in a context menu. FIG. **4** illustrates an example context menu **400**.

[0054] The context menu **400** again visualizes the active non-context-sensitive commands, including the stop command **412** (corresponds to stop command **212** of FIG. **2**), the restart command **413** (corresponding to the restart command **213** of FIG. **2**), and the delete command **414** (corresponding to the delete command **214** of FIG. **3**). Because the state of the underlying resource (i.e., the web site) was persisted, the system recognized that the start command is not a selectable command given the current state of the resource. Thus, when accessing commands for that same web site via another user interface context, the active non-context-sensitive commands are again displayed. The context menu **400** also includes a single context-sensitive command in the form of an unpin command **421**, which would remove the associated user interface element from the user interface context.

[0055] In this example, the blade user interface element **200** is one example of a user interface context with the command bar **210** being an associated context-sensitive visualization for the commands. The context menu **400** is another example of the associated context-sensitive visualization for the commands, which is associated with other user interface contexts (such as smaller parts, favorites areas, grids, activity panes, and so forth).

[0056] FIG. **5** illustrates another user interface element **500** that represents a more extended context menu. This user interface element **500** might appear when accessing commands to operate on the web site from yet another user interface context. Accordingly, the user interface element **500** represents yet a third example context-sensitive mechanism for visualizing the web commands.

[0057] The user interface element **500** again displays the non-context sensitive commands including the stop command **512** (corresponding to the stop command **212** and **412** in FIGS. **2** and **4**, respectively), the restart command **513** (corresponding to the restart command **213** and **413** in FIGS. **2** and **4**, respectively, and the delete command **514** (corresponding to the delete command **214** and **414** in FIGS. **3** and **4**, respectively). Note that the underlying resource is the web site, and the state of the web site has been preserved. Accordingly, the start command **511** (corresponding to the start command **211** of FIG. **2**) is displayed, but in deemphasized form. The browse command (corresponding to browse command **222** of FIG. **2**), and the reset publish profile command **523** (corresponding to the reset publish profile command **223** of FIG. **2**), are also displayed, even though they are context-sensitive commands.

[0058] The commands **511** through **514**, **522** and **523** are application commands **510** (also referred to herein as "extrinsic commands") being offered by application developers and not underlying system. The extended context menu **500** also includes system commands **530**, such as an unpin command **531**, and size selection commands **522** through **524**. Such system commands **530** are offered by the system regardless of the underlying resource, so long as the commands were selected within the given user interface context that generated the extended context menu **500**.

[0059] The built-in commands provide general infrastructure services (pin/unpin parts, resizing parts, restoring layout, and so forth) and are general in that they apply across all usage domains. Commands provided by application developers are domain specific. For instance, an example set of extrinsic commands for a web site application might include "start", "stop", "delete website" and so forth.

[0060] Commands are authored by application developers by leveraging a set of artifacts (interfaces and bases classes provided by the system) that expose the command contract to the application developers. This allows the application developer to provide the actual behavior of the command (what happens when the command is executed), provide dialogs

(which are optional) that will display at different moments of the command's life cycle, and influence the command life-cycle.

[0061] The non-context-sensitive commands can follow a resource in multiple contexts. For example, commands associated with a website can be present in the website's blade (see FIG. **6**A), in the website startboard part (see FIG. **6**B), when the website is displayed in a grid (see FIG. **6**C), in the notifications panel, when the website is part of a search result (see FIG. **6**D) or anywhere the website is surfaced. Note that the status of the underlying resource is considered in each of FIGS. **6**A through **6**D, in that a start command is not offered given that the web site has already started. Furthermore, note that the stop command, the restart command, and the delete command are offered regardless of the user interface context in which the commands and associated resource are visualized. Each of FIGS. **6**A through **6**B illustrate the non-context-sensitive commands being displayed via a different context-sensitive mechanism as a result of being in a different user interface context. Even though the user experience for displaying the commands and the context where the command is displayed may be different, the actual command is the same as illustrated in FIG. **7**.

[0062] FIG. **8** illustrates a life-cycle **800** that the system may be aware of for all commands, whether built-in or extrinsic. The life-cycle **800** may be tracked by, for example, a command state tracking module, which may be a single module or a collection of modules.

[0063] Before the command is initiated, there is no operation, which is the none state **801** in FIG. **8**. When the command is initiated, the state transitions **811** to a pending state **802**. In the pending state, the command is in process, and the results are pending. If the command is cancelled (transition **812**), then the operation ends transitioning to the None state **801**. If the operation completes and is successful (transition **813**), the result is success (the "Success" state **803** in FIG. **8**). Otherwise, if the operation is not successful (transition **814**), the result is failure (the "Failure" state **804** in FIG. **4**). The system understands this lifecycle even without understanding what specifically the underlying operation(s) of the command are doing.

[0064] The developer can specify whether or not constrained user interface elements (or dialogs) are to appear at each of the transition **811** through **814** for each command. Accordingly, when making a transition **811** through **814**, the system can check to determine whether a dialog is to appear as part of the transition. For instance, such dialogs could ask users for confirmations, inform of progress, or inform of the result of an operation, all depending on which transition **811** through **814** is being made, and what the resource associated with the command is.

[0065] FIG. **9** illustrates an example of a dialog **900**. In this example, the system is aware that the user has selected a stop command. The system may then track the overall lifecycle **800** of the stop command even though the system might not be aware of all that is involved in stopping the web site. The system is also aware of the resource type being operated upon (i.e., a web site) as well as an identifier for that resource ("Wandering").

[0066] Immediately upon receiving the stop command, the stop command begins transitioning (as represented by transition **811**) from the Non state **801** to the Pending state **802**. However, as part of this transition, the author verifies that the browser developer has not indicated that a dialog is to appear at this point for this type of resource (e.g., website), and/or that the web site developer has not indicated that a dialog is to appear at this point for that particular resource (e.g., the "Wandering website"). The browser developer and/or the website author may also specify a dialog template in cases in which there are multiple templates that could be used for that transition and resource type.

[0067] Here, the system verifies that a dialog is to appear, and thus presents dialog **900**. The dialog **900** may be generated knowing nothing more than which transition is involved (and potentially also a dialog template to use which may also be specified by the developer). The dialog **900** may then populate the dialog template using the name of the resource (e.g., "Wandering" website), and then present the dialog to the user. Thus, the presentation of dialogs may be consistent throughout the system regardless of the command being executed, or the resource being operated upon, even without the system knowing the specifics of the underlying operations that support the command.

[0068] These dialogs are data-driven and extremely constrained to provide a uniform user experience across applications. Dialogs include confirmation (with yes/no buttons), show progress (deterministic and non-deterministic), show success, and show failure (with a retry button). Application developers can configure the command to surface the dialogs at certain points in the lifecycle of the operation.

[0069] As illustrated in FIG. **10**, each stage in the life cycle can surface a different dialog, with the application developer indicating whether the corresponding dialog is to appear at each stage in the state machine.

[0070] In some cases, the portal can provide abstractions that application developers can use to create intrinsic commands that the system will recognize (at least to the point of being able to track the state machine of FIG. **10**). An example can be extensible abstractions for "Save" and "Discard" commands that when used in forms are subject to the validation state and changes to the underlying form.

[0071] Commands are executed asynchronously by the system. Commands provided by application authors are executed leveraging the system's isolation model to ensure that they do not compromise the overall portal (as the execution is isolated within the application that owns the command).

[0072] The application developer creates a small set of commands that are available in multiple contexts. All capabilities for his commands (execution logic, dialogs, and so forth) are preserved and the necessary user experience is adapted to the constraints of where the command is rendered. This makes possible that users can interact with a resource at any place in the user interface. There is no single location where "actions" can be executed but rather any place in the portal allows rich interactions with resources.

[0073] Accordingly, a system has been described that provides consistency in how commands are visualized, as well as how dialogs associated with the command lifecycle are visualized. This is true regardless of there being user interface elements of different applications within the system.

[0074] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes

which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. A computer program product comprising one or more computer-readable storage media having thereon computer-executable instructions that are structured such that, when executed by one or more processors of a computing system, cause the computing system to instantiate and/or operate the following:

a plurality of non-context-sensitive commands that may be initiated from each of a plurality of user interface contexts; and

a plurality of context-sensitive mechanisms for visualizing the plurality of non-context-sensitive commands depending on which of the plurality of user interface contexts in which the plurality non-context sensitive commands appear.

2. The computer program product in accordance with claim **1**, the plurality of non-context-sensitive commands being associated with a resource upon which each of the plurality of non-context-sensitive commands acts when selected.

3. The computer program product in accordance with claim **1**, the one or more computer-readable storage media having thereon computer-executable instructions that are structured such that, when executed by one or more processors of a computing system, cause the computing system to further instantiate and/or operate the following:

one or more context-sensitive commands visualized with the plurality of non-context-sensitive commands, but which may differ depending on which of the plurality of user interface contexts in which the plurality of non-context-sensitive commands appear.

4. The computer program product in accordance with claim **1**, the different contexts in the user interface including a first set of one or more contexts associated with a first application, and a second set of one or more contexts associated with a second application.

5. The computer program product in accordance with claim **4**, the first set of one or more contexts associated with the first application including a first context of a particular context type, and the second set of one or more contexts associated with the second application also including a second context also of the same particular context type.

6. The computer program product in accordance with claim **5**, any context of the particular context type being associated with a particular context-sensitive mechanism for visualizing the plurality of non-context-sensitive commands, such that the mechanism for visualizing the plurality of non-context-sensitive commands is the same in the first context of the particular context type and associated with the first application as the mechanism for visualizing the plurality of non-context-sensitive commands in the second context of the particular context type and associated with the second application.

7. The computer program product in accordance with claim **1**, the plurality of non-context-sensitive commands being a first plurality of non-context-sensitive commands, the one or more computer-readable storage media further having thereon computer-executable instructions that are structured such that, when executed by one or more processors of the computing system, cause the computing system to instantiate and/or operate the following:

a second plurality of non-context-sensitive commands that may be initiated from each of the plurality of user interface contexts; the plurality of context-sensitive mechanisms also for visualizing the second plurality of non-context-sensitive commands,

such that for a given user interface context, the same context-sensitive mechanism is used to visualize the first plurality of non-context-sensitive commands as would be used to visualize the second plurality of non-context-sensitive commands in that given user interface context.

8. The computer program product in accordance with claim **1**, the plurality of non-context-sensitive commands being selectable from a user interface element, the execution of at least one of the plurality of non-context-sensitive commands resulting in a change in data displayed in the user interface element.

9. The computer program product in accordance with claim **1**, the plurality of context-sensitive mechanisms being intrinsic to a system, and not alterable by applications running within the system.

10. A method for executing a command from a user interface element, the method comprising:

an act of initiating the command; and

an act of tracking the command at a plurality of stages;

for each of the plurality of stages, determining whether a dialog is indicated as to be displayed, and if so, displaying a dialog for the corresponding stage that is consistent across a plurality of commands.

11. The method in accordance with claim **10**, the method being performed by a command state tracking module, and the command being a first command, the method further comprising:

an act of initiating a second command; and

an act of tracking the second command at a plurality of stages of the second command that are the same as the plurality of stages of the first command;

for each of the plurality of stages of the second command, determining whether a dialog is indicated as to be displayed, and if so, displaying a dialog for the corresponding stage that is consistent across a plurality of commands.

12. The method in accordance with claim **11**, the first and second commands both being intrinsic commands.

13. The method in accordance with claim **11**, the first and second commands both being extrinsic commands.

14. The method in accordance with claim **11**, one of the first and second commands being an extrinsic command, and the other of the first and second commands being an intrinsic command.

15. The method in accordance with claim **10**, wherein the act of determining whether a dialog is to be displayed is performed at least at each transition between the plurality of stages.

16. The method in accordance with claim **15**, the determination of whether or not to display a dialog being provided by a developer.

17. The method in accordance with claim **15**, the act of displaying the dialog performed at each transition being a function of the transition.

18. The method in accordance with claim **10**, the act of displaying the dialog being a function of a resource that is being operated upon.

19. A computer program product comprising one or more computer-readable storage media having thereon computer-executable instructions that are structured such that, when executed by one or more processors of a computing system,

cause the computing system to perform a method for executing a command from a user interface element, the method comprising:

an act of initiating the command; and

an act of tracking the command at a plurality of stages;

for each of the plurality of stages, an act of determining whether a dialog is indicated as to be displayed, and if so, displaying a dialog for the corresponding stage that is consistent across a plurality of commands.

20. The computer program product in accordance with claim **19**, the plurality of commands including a plurality of extrinsic commands and a plurality of intrinsic commands.

* * * * *