(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 April 2015 (02.04.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/048206 A1

(51) International Patent Classification:
G06F 9/44 (2006.01)

(21) International Application Number:
PCT/US2014/057324

(22) International Filing Date:
25 September 2014 (25.09.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/884,743	30 September 2013 (30.09.2013)	US
61/905,128	15 November 2013 (15.11.2013)	US
61/905,111	15 November 2013 (15.11.2013)	US
61/905,114	15 November 2013 (15.11.2013)	US
61/905,116	15 November 2013 (15.11.2013)	US
61/905,129	15 November 2013 (15.11.2013)	US
61/905,105	15 November 2013 (15.11.2013)	US
61/905,101	15 November 2013 (15.11.2013)	US
61/905,119	15 November 2013 (15.11.2013)	US
61/905,243	17 November 2013 (17.11.2013)	US
61/905,247	17 November 2013 (17.11.2013)	US
14/231,869	1 April 2014 (01.04.2014)	US

(71) Applicant: **MICROSOFT TECHNOLOGY LICENSING, LLC** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **OLENICK, Brad**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **ZUNINO, Gilles C.J.A.**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **MOY, William Y.**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **BIRCK, Andrew**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **JOSHI, Madhur**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **ALEXANDER, Brendyn**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **COX, Tom**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **ECHERUO, Ifeanyi**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **GOUPIL, Jean-Sebastien**; c/o Microsoft Corporation, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM,

[Continued on next page]

(54) Title: RENDERING INTERPRETER FOR VISUALIZING DATA PROVIDED FROM RESTRICTED ENVIRONMENT CONTAINER

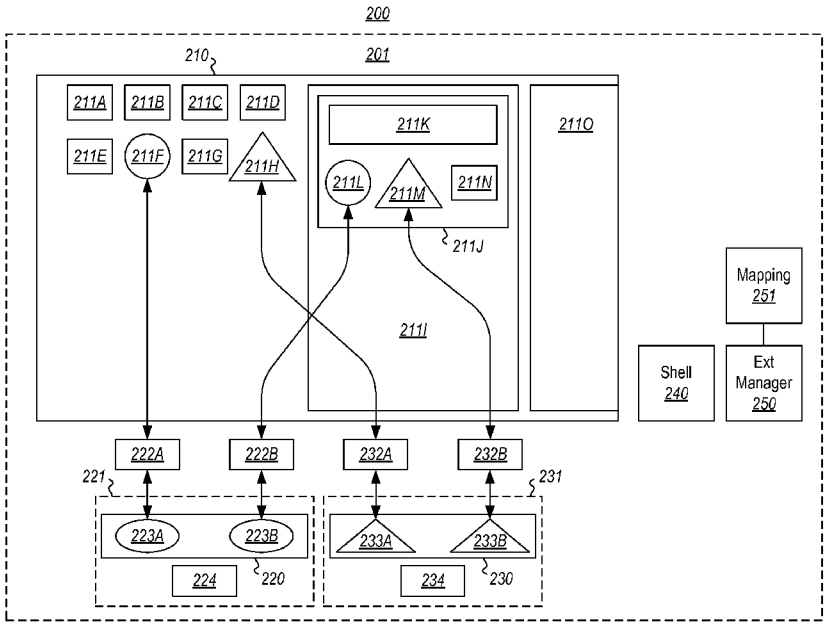


Figure 2

(57) Abstract: At least some embodiments described herein refer to the user of a rendering interpreter that receives data object(s) from code executing within a restricted environment container (such as an IFrame). The restricted environment container prevents the code from adversely impacting the performance and consistency of the user interface system as a whole, and also prevents the code from adversely impacting the performance of code running in other restricted environment containers.

WO 2015/048206 A1

AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

RENDERING INTERPRETER FOR VISUALIZING DATA PROVIDED FROM RESTRICTED ENVIRONMENT CONTAINER

BACKGROUND

[0001] The primary purpose of an IFrame is to render content (hereinafter "untrusted content") within a particular window of a larger user interface in a manner that the window looks like an integrated part of the larger user interface. The content is "untrusted" in that it is generated by untrusted code (often script). The IFrame provides an isolation function that prevents the script from doing harmful actions to objects outside of the IFrame. After all, the script provided within an IFrame is often provided by a third-party entity.

[0002] There is typically one IFrame per window. The main application content is typically displayed in one IFrame. Furthermore, there is a dedicated IFrame for each window having untrusted content. In conventional user interface systems, there might be one or two IFrames (other than the primary IFrame for the main application content) that operate to render untrusted content on respective portions of the display. For instance, in a browser application, the main browser display might be rendered by one IFrame. There might be a small stock update window in another corner of the display that regularly updates a few designated stock prices. The content of that small window is also driven by a separate IFrame. Thus, in conventional approaches, there is a one-to-one correspondence between IFrames and windows that display untrusted content.

BRIEF SUMMARY

[0003] At least some embodiments described herein refer to the user of a rendering interpreter that receives data object(s) (such as a view model) from code executing within a restricted environment container (such as an IFrame). The restricted environment container prevents the code from adversely impacting the performance and consistency of the user interface system as a whole, and also prevents the code from adversely impacting the performance of code running in other restricted environment containers.

[0004] The rendering interpreter processes the data object(s) received from the restricted environment container, and exposes the resulting data to a user interface element that is external to the restricted environment container. The resulting data is structured such that the user interface element visualizes at least some of the resulting data. The code is thereby permitted to safely project content into the user interface element.

[0005] In some embodiments, a single restricted environment container may contain multiple code elements, each responsible for projecting content into a corresponding user

interface element. Accordingly, there is no longer a one-to-one mapping between content windows and restricted environment containers. Instead, in the context in which the user interface application is extended by multiple other entities, a given application extension may be allocated a single restricted environment container, regardless of how many content windows that extension populates. Code associated with that extension then executes within the corresponding restricted environment container. Thus, far more elements of dynamic content may be rendered on a user interface using a smaller number of restricted environment containers. Accordingly, a user interface system may safely gain significant dynamic content contributions from other providers.

[0006] 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

[0007] 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:

[0008] Figure 1 illustrates an example computing system in which the principles described herein may be employed;

[0009] Figure 2 illustrates environment that includes a user interface system that may operate with the computing system of Figure 1; and

[0010] Figure 3 illustrates a mechanism in which code operates on data within an IFrame, and in which data is transferred between the IFrame and a user interface element.

DETAILED DESCRIPTION

[0011] At least some embodiments described herein relate to a computing system that includes a user interface system for presenting a user interface having a variety of user interface elements. While the user interface system presents the user interface elements as a container, at least some of the user interface elements may be populated by applications that extend the features of the user interface elements. For instance, the user interface

system might be a web application. The applications that extend the features of the browser application might be extensions to the web application.

[0012] In order that there is safe and secure isolation between the applications and the user interface system, and between the applications themselves, each application has a dedicated restricted environment container (e.g., an undisplayed iFrame). Each user interface element also includes a rendering interpreter, each configured to process any received data objects (such as a view model) from corresponding code running in a restricted environment container and expose resulting data to a corresponding user interface element. The resulting data provided by any given rendering interpreter is structured such that the corresponding user interface element visualizes at least some of the resulting data, to thereby allow the corresponding code to project content into the corresponding user interface element. Such providing of resulting data will also be referred to hereinafter as "projecting content".

[0013] In the case of the received data object(s) being the rendering interpreter, a copy of the view model may be contained within a corresponding restricted environment container, and another synchronized copy of the same view model may be contained with the user interface system execution environment. As changes are made to one copy of the view model, they are propagated to the other copy of the view model using messaging. Accordingly, the code for an application running in a restricted environment container may project visualizations into appropriate user interface elements without the risk of harm to the user interface system or other applications. Furthermore, since the rendering interpreters can enforce rules on how content is projected into the user interface element, the rendering interpreter tends to provide a consistent look and feel of the user interface. Likewise, when changes are made to the visualization, those changes may be propagated back to the underlying data within the restricted environment container.

[0014] Some introductory discussion of a computing system will be described with respect to Figure 1. Then, example user interfaces, methods and supporting architectures will be described with respect to subsequent figures.

[0015] 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.

[0016] As illustrated in Figure 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).

[0017] 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.

[0018] 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.

[0019] 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.

[0020] 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.

[0021] 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.

[0022] 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.

[0023] 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.

[0024] 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, multi-processor 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.

[0025] Figure 2 illustrates an environment 200 that may operate with the computing system 100 of Figure 1. The environment 200 includes a user interface system 201 that presents a user interface 210. The principles described herein may operate using any user interface that has multiple user interface elements no matter what those user interface elements are, or how distributed. For purposes of example, the user interface 210 has multiple user interface elements 211A through 211O.

[0026] There are one or more applications that extend the features of the user interface system (hereinafter also referred to as "extension applications") and that operate within the user interface system. However, the code for each extension application runs within a dedicated restricted environment container that is dedicated to running code for that extension application. This prevents any given extension application from harming or disrupting the consistency of the user interface system, or harming the performance of another application. For instance, in Figure 2, there are two extension applications 220 and 230 running within corresponding dedicated restricted environment containers 221 and 231.

[0027] Although not required, the restricted environment containers 221 may be a container usually provisioned to display dynamic content using code executed therein. For instance, each restricted environment container may be, for instance, an IFrame. However, the IFrame is configured so as not to render any displayable content, but just to run code therein, thereby taking advantage of the isolation characteristics of the IFrame.

[0028] Typically, in dynamic web pages, there is one IFrame per user interface element when untrusted code within the IFrame is executed to generate that content. However, this is not scalable when the number of such user interface elements (and corresponding IFrames) rises, as the memory resources of the system become overwhelmed. This would be especially true for a user interface that allows other untrusted applications to populate user interface elements.

[0029] In accordance with the principles described herein, IFrames are apportioned one per extension application, as opposed to one per dynamic user interface element. Accordingly, since any given application extension may populate many more than just a single dynamic interface element with dynamic content, this significantly reduces the number of IFrames needed for user interfaces, thereby facilitating user interfaces with large numbers of user interface elements that contain dynamic content.

[0030] For instance, in the case of Figure 2, there are four user interface elements illustrated as having content driven by execution of untrusted extension applications, including user interface elements 211F, 211H, 211L and 211M. Using the conventional approach, this would require four IFrames. However, this number has been reduced to two IFrames (corresponding to restricted environment containers 221 and 231) in Figure 2. The percentage reduction in IFrame usage can be more significant in a complex user interface with many user interface elements populated by only a few extension applications.

[0031] The user interface system 201 manages a mapping 251 between a given extension application and the user interface element that the application populates. For instance, in Figure 2, the mapping 251 would map an identification of the extension application 220 to the identification of the user interface elements 211F and 211L. The mapping 251 would also map an identification of the extension application 230 to the identification of the user interface elements 211H and 211M.

[0032] A container management component (or an "extension manager component") 250 has access to the mapping, and is responsible for creating and removing restricted environment containers. The container management component creates a new restricted environment container when a user interface element is to be displayed that is to be

populated by code from a particular extension application for which there is not yet a restricted environment container. In addition, the container management component loads the corresponding extension application into the newly created restricted environment component. On the other hand, the container management component 250 releases (i.e., removes) a restricted environment container corresponding to a particular application when the computing system no longer has any user interface state corresponding to any user interface elements that are populated by the particular application. The shell 240 is responsible for adding and removing user interface elements and notifies the container management component 250 of these events.

[0033] Towards the goal of managing a small number of restricted environment containers sufficient to render only those user interface elements that are on screen (and perhaps a little more to facilitate smoother scrolling or other navigation), a virtualization component (which may be part of the shell 240 in Figure 2) releases user interface state of a user interface element from the computing system under at least some circumstances when a user interface element is no longer displayed on the display. This too significantly decreases the use of restricted environment containers to a predictable number that can be managed explicitly by the user interface system.

[0034] The user interface system is responsible for exposing the user interface element in a manner that they can become populated. For instance, the shell 240 determines when a new user interface element is to be rendered, and obtains the template (e.g., markup) for the user interface element. The shell 240 then calls into the appropriate extension application to populate the user interface element. The population occurs by the corresponding code providing one or more data objects (e.g., a view model) to a corresponding rendering interpreter. The rendering interpreter then sends the resulting data sent from the view model to the corresponding user interface element, wherein it is used to populate the user interface element.

[0035] The application 220 includes code 223A and code 223B, which are responsible for populating respective user interface elements 211F and 211L, and may operate based on application data 224. The application 230 includes code 233A and code 233B, which are responsible for populating respective user interface elements 211H and 211M, and may operate using application data 234. In one embodiment, each illustrated item of code may be a single control, or perhaps multiple controls, or even complex controls that are hierarchically structured, or depend on the output from other controls. The controls might have been sanitized and approved for use by the user interface system. For instance, for

intrinsic controls offered by the user interface system, the system provider ensures safe operation. On the other hand, extension applications offer markup templates that are sanitized by the shell 240. The extension manager 250 may have responsibility for loading the code into the appropriate application when and/or after the corresponding application is loaded into the respective restricted environment container.

[0036] As previously mentioned, the results of execution of this code are not provided directly into the execution environment of the user interface system 201. Instead, each code has a corresponding rendering interpreter, which restricts the operations and data that can be performed by the code. This permits the user interface system 201 and other extension applications to be protected from unintended harmful consequence of the extension application, or even from rogue extension applications.

[0037] When the code is to provide content to the respective user interface element, the code executes and provides one or more resulting data objects to its respective rendering interpreter. For instance, in Figure 2, code 223A, 223B, 233A and 233B having corresponding rendering interpreters 222A, 222B, 232A and 232B. As an example, the rendering interpreters might each be view models.

[0038] Each rendering interpreter is configured to process any received data objects from corresponding code running in a restricted environment container, and expose resulting data to its corresponding user interface element. The resulting data is structured such that the corresponding user interface element visualizes at least some of the resulting data, to thereby allow the corresponding code to project content into the corresponding user interface element.

[0039] In the embodiment in which the IFrame is used as a restricted environment container, and the received one or more data objects is a view model, there may be two copies of the view model, one operating within the IFrame and one operating openly within the user interface system environment. Figure 3 illustrates a mechanism 300 in which code 311 operates on data 312 within an IFrame 310 and provides the resulting objects as a view model 313A that is also contained within the IFrame 310. There is also a copy 313B of the view model operating within the user interface system execution environment (which incidentally, might also be an IFrame). It is that copy 313B which the rendering interpreter accesses to thereby populate the corresponding user interface element 321 corresponding to the code 311.

[0040] The copies 313A and 313B of the view model are kept synchronized by sending messages to and receiving messages from a messaging component 320. For instance, when

the code 311 provides alters the view model 313A within the IFrame 310, the messaging component 320 detects the change, and likewise messages the view model 313B operating outside the IFrame 310. The rendering interpreter may then access the view model 313B to project content into the user interface element 321. In reverse, if a change is made to the state of the user interface element 321, that change is reflected within the view model 313B outside of the IFrame 310. The view model 313B then messages the messaging component 320. The messaging component 320 messages the view model 313A inside the IFrame 310 within the altered object(s). The code 311 then accesses the view model 313A, and the application data 312 is updated.

[0041] Accordingly, embodiments described herein allow a user interface system to be enriched with the contributions of several application extensions, without those application extensions adversely affecting consistency in the user interface system, and without the application extensions harming functionality of the user interface system or each other.

[0042] 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.

CLAIMS

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 a rendering interpreter configured to do the following in response to receiving one or more data objects to be rendered from code executing within a restricted environment container:

an act of processing the received one or more data objects and exposing resulting data to a user interface element on a user interface provided by a user interface system, the resulting data structured such that the user interface element visualizes at least some of the resulting data, to thereby allow the code to project content into the user interface element.

2. The computer program product in accordance with Claim 1, the restricted environment container being suitable for displaying content if reconfigured.

3. The computer program product in accordance with Claim 1, the restricted environment container being an IFrame, and the received one or more data objects being a view model.

4. The computer program product in accordance with Claim 1, a copy of the one or more data objects being within the restricted environment container, and a synchronized copy of the one or more data objects being within a user interface system environment.

5. The computer program product in accordance with Claim 1, the code being code that runs as part of an application that extends features of the user interface system, the user interface element exposed on the user interface by the user interface system, but populated by the code that runs as part of the application.

6. The computer program product in accordance with Claim 1, wherein the user interface system comprising a plurality of restricted environment containers, each dedicated to running code from a corresponding application that expands features of the user interface system.

7. The computer program product in accordance with Claim 6, wherein the computer-executable instructions are further structured such that, when executed by the one or more processors of the computing system, further cause the computing system to instantiate and/or operating the following:

a container management component that instantiates a new restricted environment container when a user interface element is to be displayed that is to be populated by code from a particular application for which there is not yet a restricted environment container.

8. The computer program product in accordance with Claim 6, wherein the computer-executable instructions are further structured such that, when executed by the one or more processors of the computing system, further cause the computing system to instantiate and/or operating the following:

a virtualization component that releases user interface state of a user interface element from the computing system under at least some circumstances when a user interface element is no longer displayed on the display, but is still in the user interface.

9. The computer program product in accordance with Claim 8, wherein the computer-executable instructions are further structured such that, when executed by the one or more processors of the computing system, further cause the computing system to instantiate and/or operating the following:

a container management component that releases a restricted environment container corresponding to a particular application when the computing system no longer has any user interface state corresponding to any user interface elements that are populated by the particular application.

10. A computing system operating thereon the following:

a user interface system for presenting a user interface having a plurality of user interface elements;

a plurality of applications that expand features of the user interface system;

a plurality of restricted environment containers, each dedicated to running code of a corresponding one of the plurality of restricted environment containers,

a plurality of rendering interpreters, each configured to process any received data objects from corresponding code running in a restricted environment container and to expose resulting data to a corresponding user interface element on the user interface provided by a user interface system, the resulting data structured such that the corresponding user interface element visualizes at least some of the resulting data, to thereby allow the corresponding code to project content into the corresponding user interface element,

wherein at least some of the plurality of applications populate corresponding groups of one or more of the plurality of user interface elements.

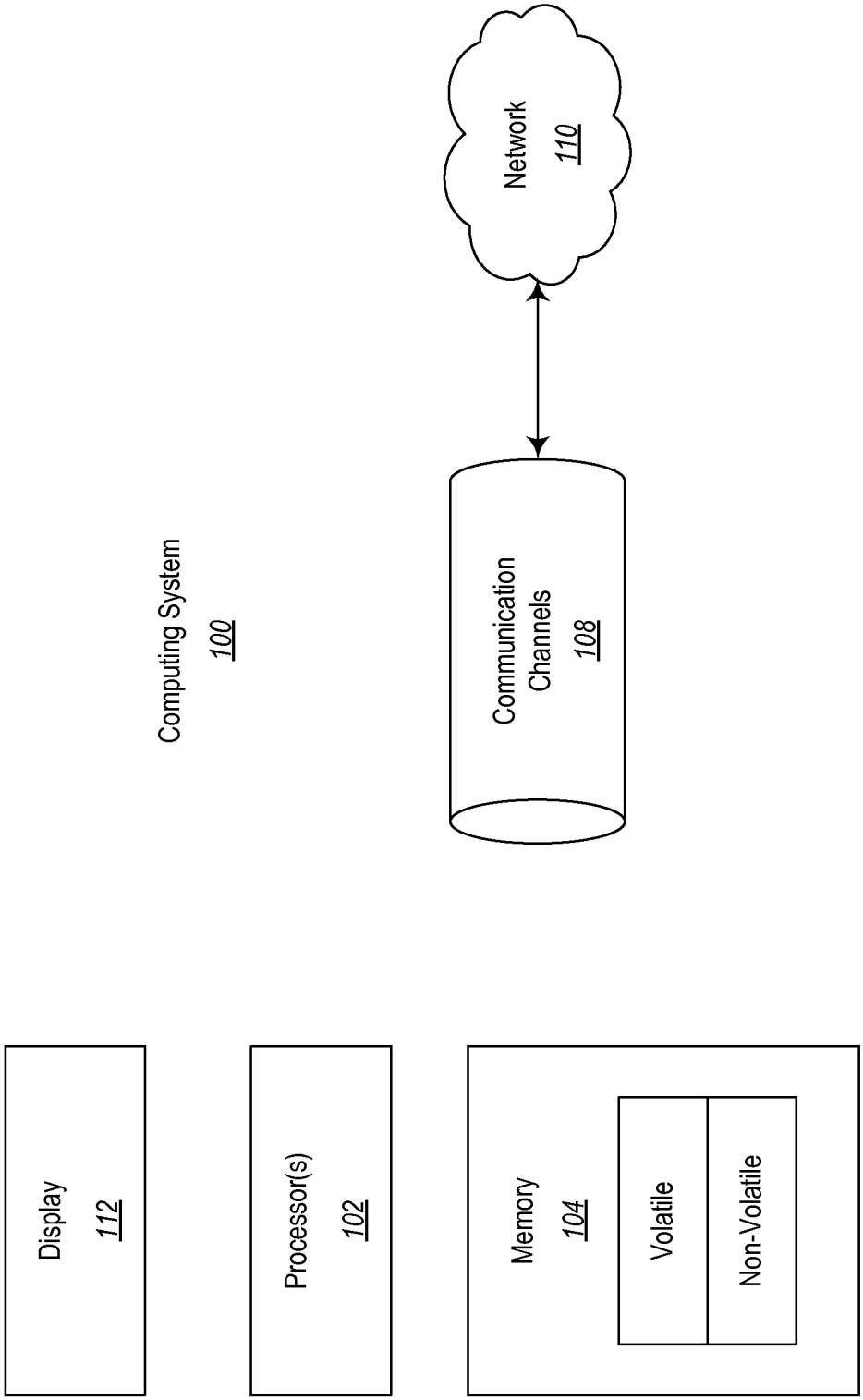

Network
110
Communication Channels
108
Computing System
100
Display
112
Processor(s)
102
Memory
104
Volatile
Non-Volatile

Figure 1

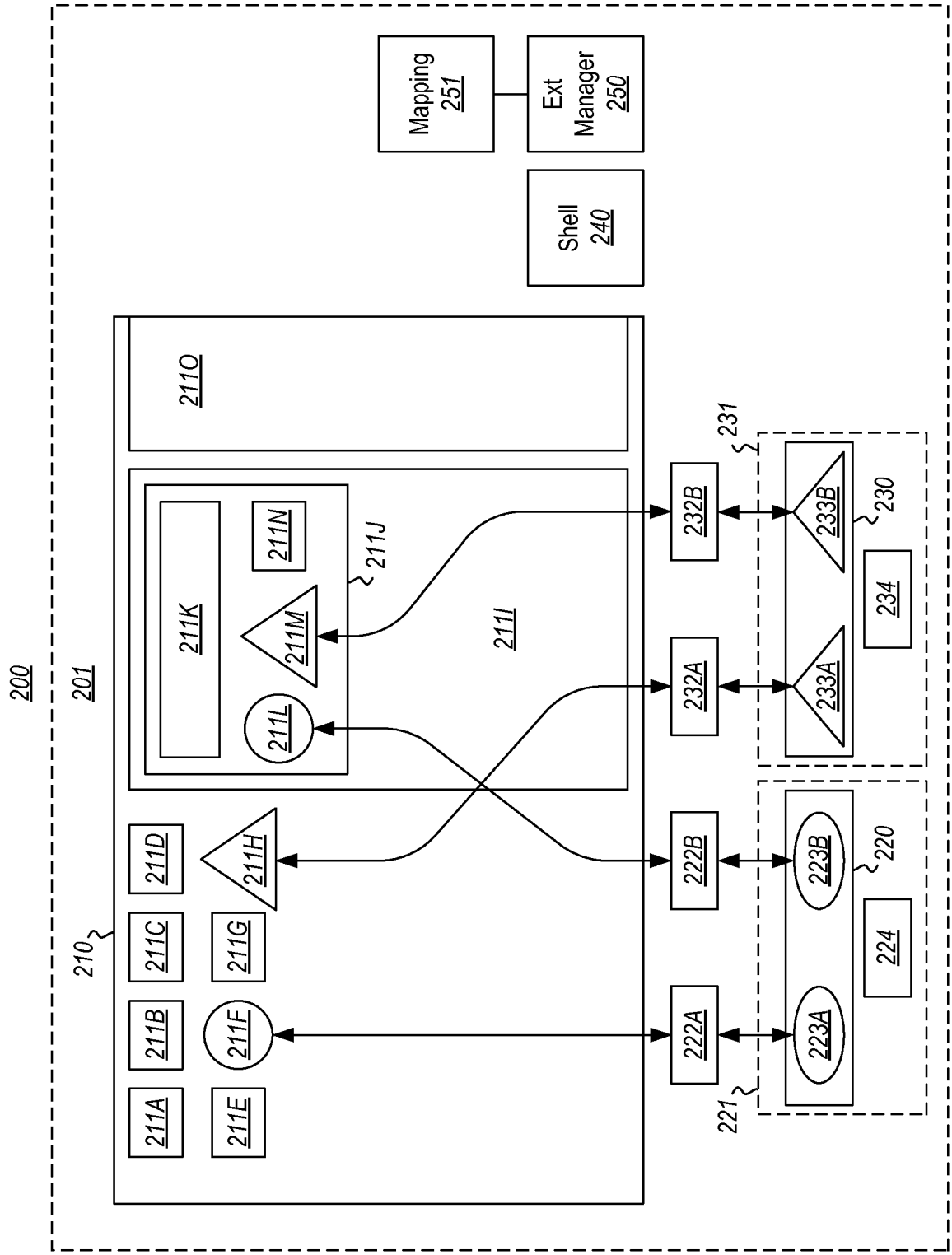

200
201
210
211A
211B
211C
211D
211E
211F
211G
211H
211I
211J
211K
211L
211M
211N
211O
Shell 240
Ext Manager 250
Mapping 251
221
222A
222B
223A
223B
224
220
231
232A
232B
233A
233B
234
230

Figure 2

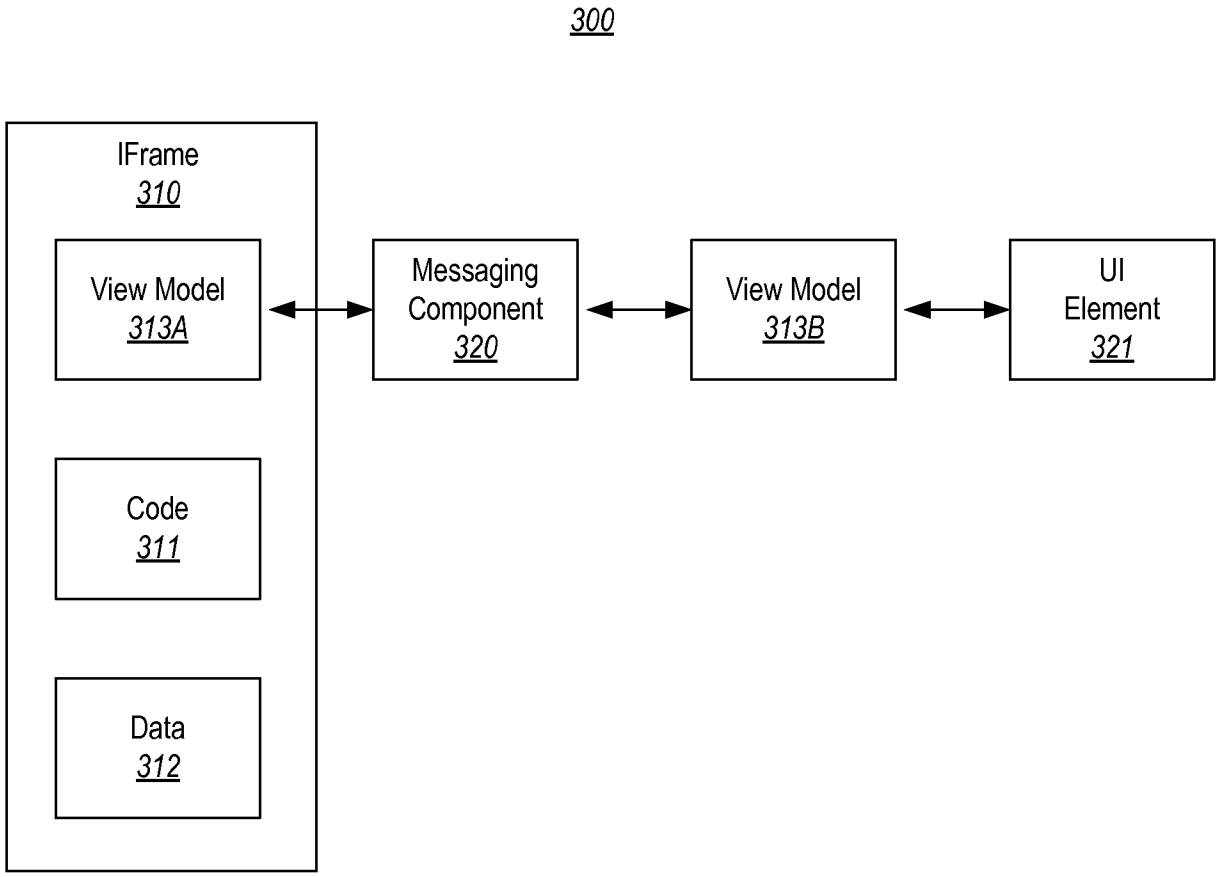

300
IFrame
310
View Model
313A
Messaging
Component
320
View Model
313B
UI
Element
321
Code
311
Data
312

Figure 3

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/057324

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/44
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	W3c: "HTML 5 - A vocabulary and associated APIs for HTML and XHTML", , 6 August 2013 (2013-08-06), pages 1-121, XP055154168, Retrieved from the Internet: URL:http://www.w3.org/TR/2013/CR-html5-20130806/embedded-content-0.html#the-iframe-element [retrieved on 2014-11-20] page 22, last paragraph - page 29, last paragraph -----	1-10

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 November 2014	01/12/2014

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Archontopoulos, E