(12) UK Patent Application (19) GB (11) 2526306 (13) A

(43) Date of A Publication 25.11.2015

(21) Application No: 1408930.4

(22) Date of Filing: 20.05.2014

(71) Applicant(s):
Canon Kabushiki Kaisha
(Incorporated in Japan)
3-30-2 Shimomaruko, Ohta-ku, Tokyo, Japan

(72) Inventor(s):
Romain Bellessort
Youenn Fablet
Hervé Ruellan

(74) Agent and/or Address for Service:
Santarelli
49, avenue des Champs-Elysées, Paris 75008, France (including Overseas Departments and Territori es)

(51) INT CL:
G06F 17/30 (2006.01) ***G06F 9/54*** (2006.01)

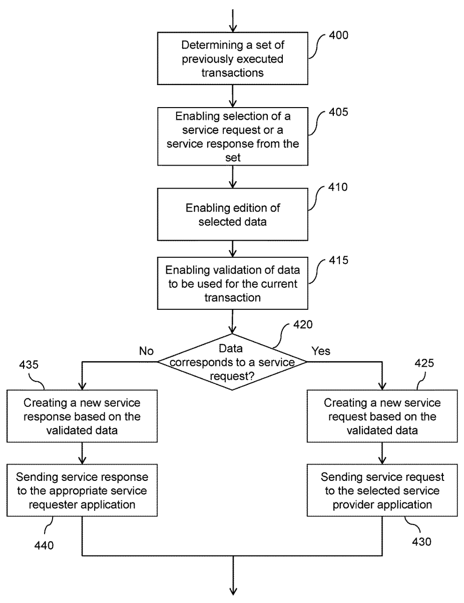
(56) Documents Cited:
US 20090234876 A1 **US 20090152341 A1**
US 20090013036 A1 **US 20080305815 A1**

(58) Field of Search:
INT CL **G06F**
Other: **Online: EPODOC, TXTE, WPI**

(54) Title of the Invention: **Method, device, and computer program for simplifying the processing of service requests in a web runtime environment**
Abstract Title: **Processing of service requests in a web runtime environment**

(57) The invention relates to processing of service requests (e.g. a share action allowing the sharing of content on a social network, a pick action for picking content such as an image, a view action for viewing files, and an edit action for editing an image, a web intent object) in a web runtime environment. After having accessed one part of one transaction, the accessed part of the transaction being previously stored and comprising at least one of one part of a service request and one part of a corresponding service response generated by a service provider application, a list comprising the accessed part is displayed. Next, one part of one transaction can be selected, in the displayed list, by a user, the selected part comprising at least one part of a service request or at least one part of a service response generated by a service provider application in response to a service request distinct from the service request being processed. Next, a service response to the service request being processed is obtained. The obtained service response is obtained as a function of the selected part of one transaction.

Fig. 4


Determining a set of previously executed transactions
400
Enabling selection of a service request or a service response from the set
405
Enabling edition of selected data
410
Enabling validation of data to be used for the current transaction
415
Data corresponds to a service request?
420
No
Yes
435
425
Creating a new service response based on the validated data
Creating a new service request based on the validated data
Sending service response to the appropriate service requester application
Sending service request to the selected service provider application
440
430

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

GB 2526306 A

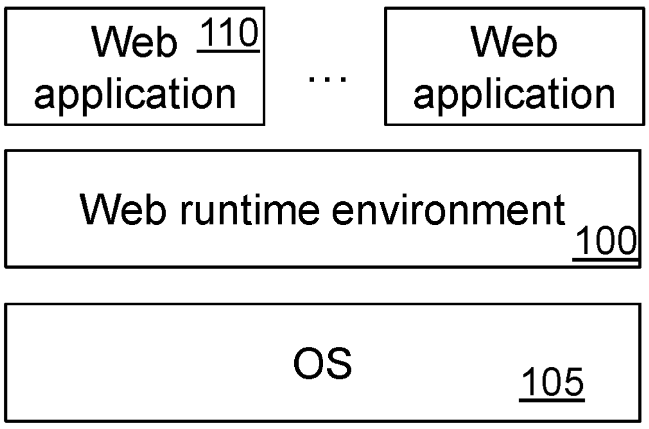


Fig. 1a

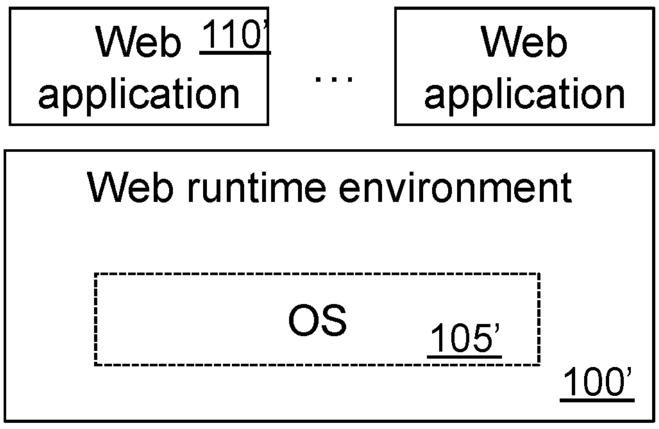


Fig. 1b

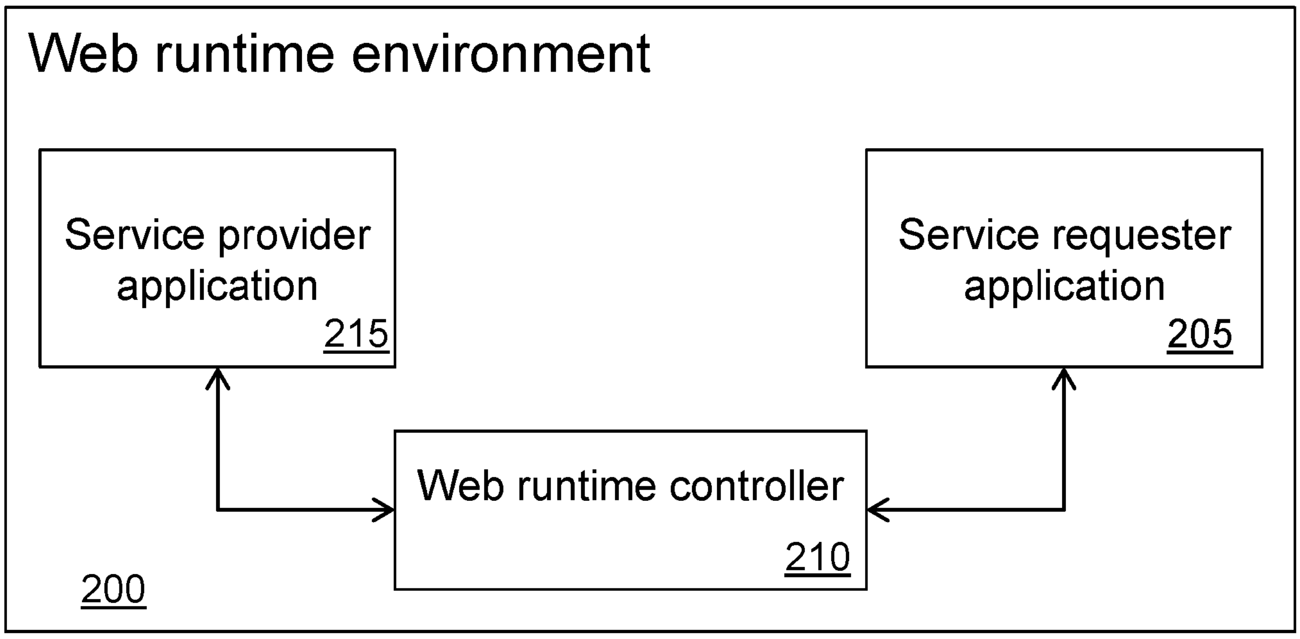

Web runtime environment
Service provider application 215
Service requester application 205
Web runtime controller 210
200

Fig. 2

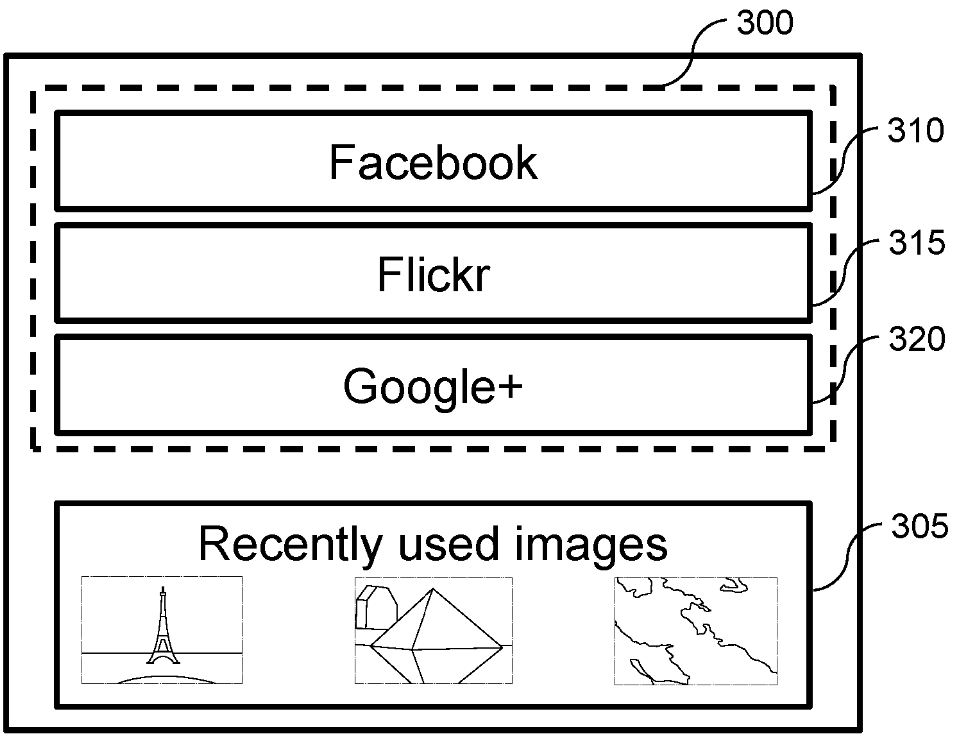


Fig. 3a

28 11 14

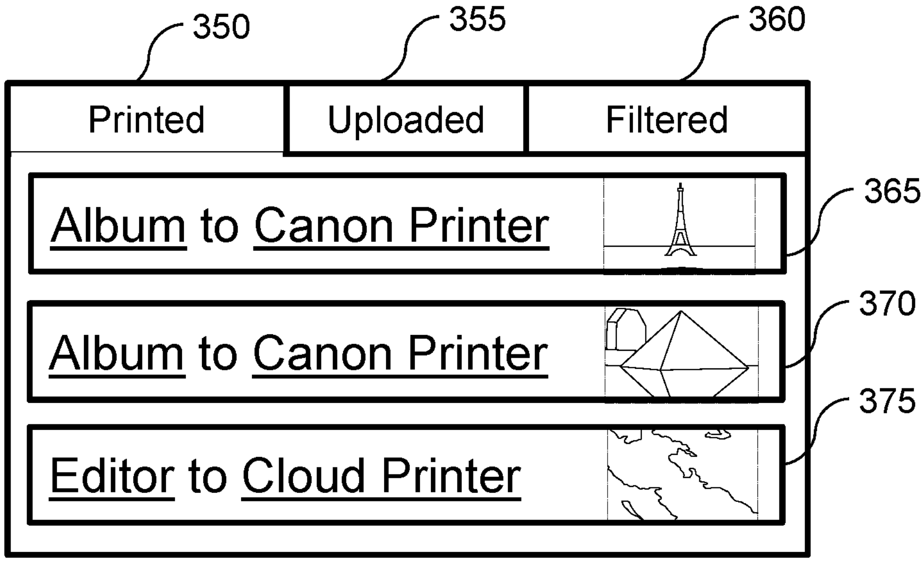


Fig. 3b

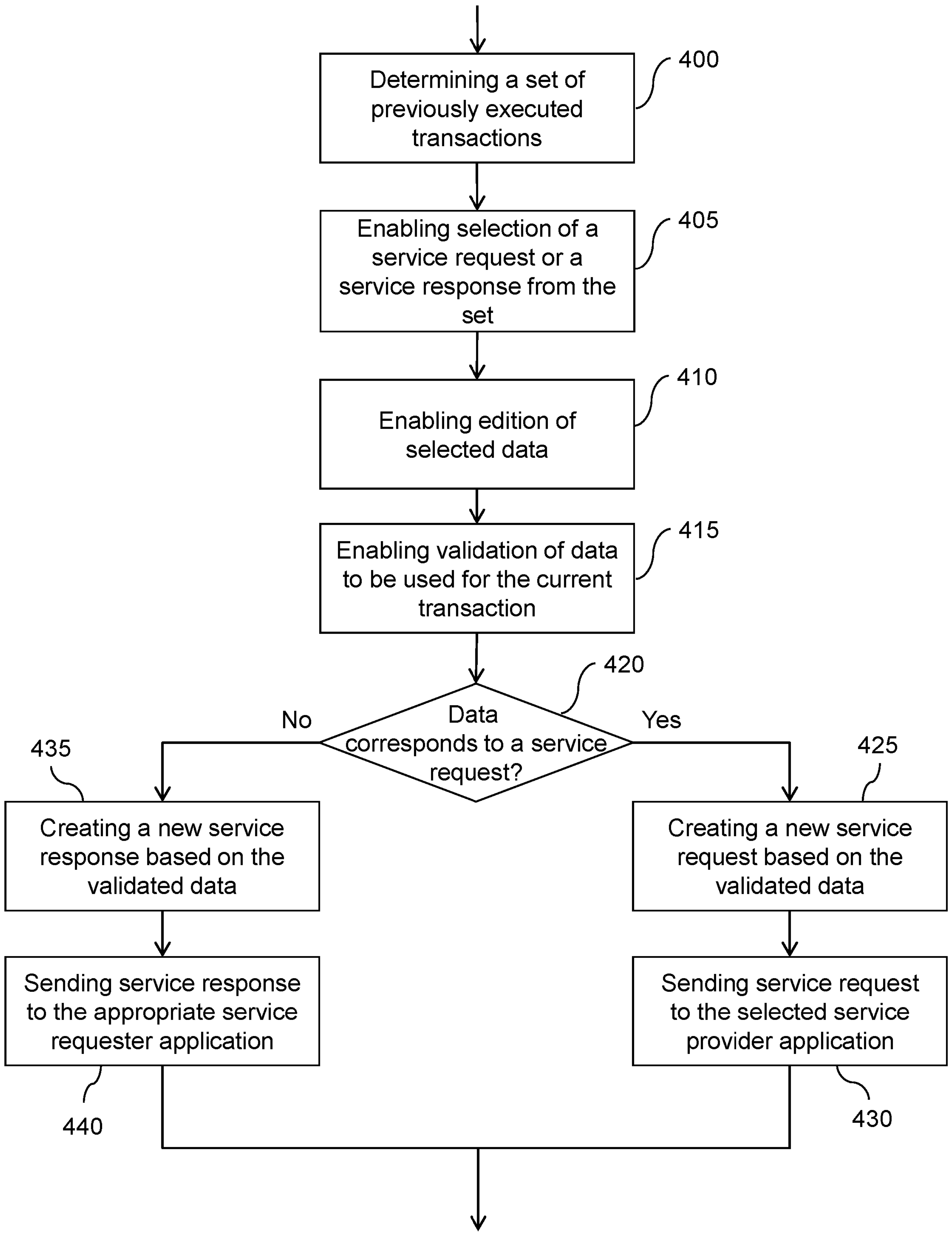
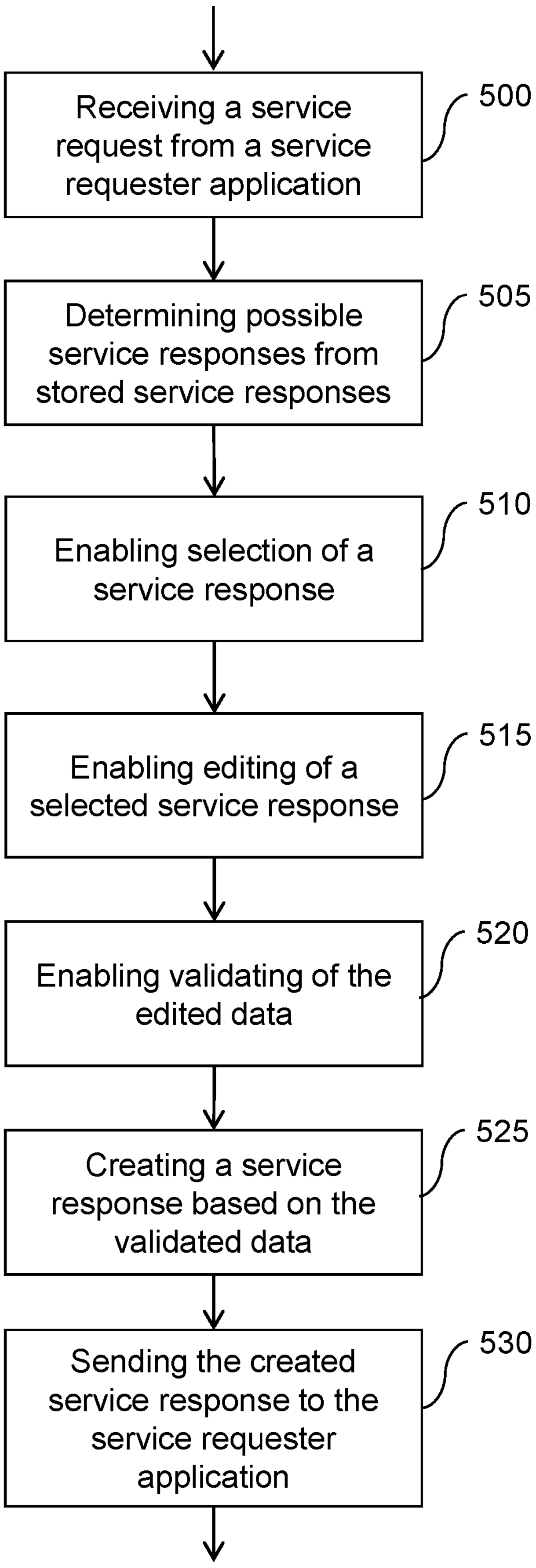
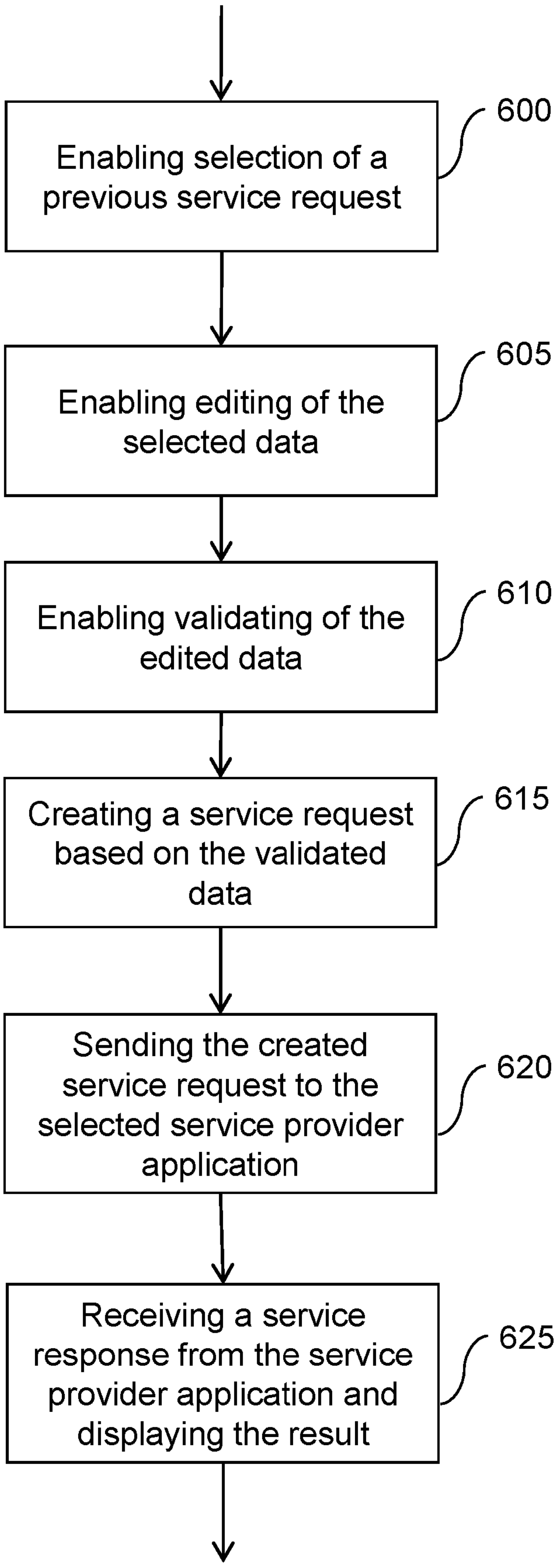

Determining a set of previously executed transactions
400
Enabling selection of a service request or a service response from the set
405
Enabling edition of selected data
410
Enabling validation of data to be used for the current transaction
415
Data corresponds to a service request?
420
No
Yes
435
Creating a new service response based on the validated data
425
Creating a new service request based on the validated data
Sending service response to the appropriate service requester application
Sending service request to the selected service provider application
440
430

Fig. 4



Receiving a service request from a service requester application
500
Determining possible service responses from stored service responses
505
Enabling selection of a service response
510
Enabling editing of a selected service response
515
Enabling validating of the edited data
520
Creating a service response based on the validated data
525
Sending the created service response to the service requester application
530

Fig. 5



Enabling selection of a previous service request
600
Enabling editing of the selected data
605
Enabling validating of the edited data
610
Creating a service request based on the validated data
615
Sending the created service request to the selected service provider application
620
Receiving a service response from the service provider application and displaying the result
625

Fig. 6

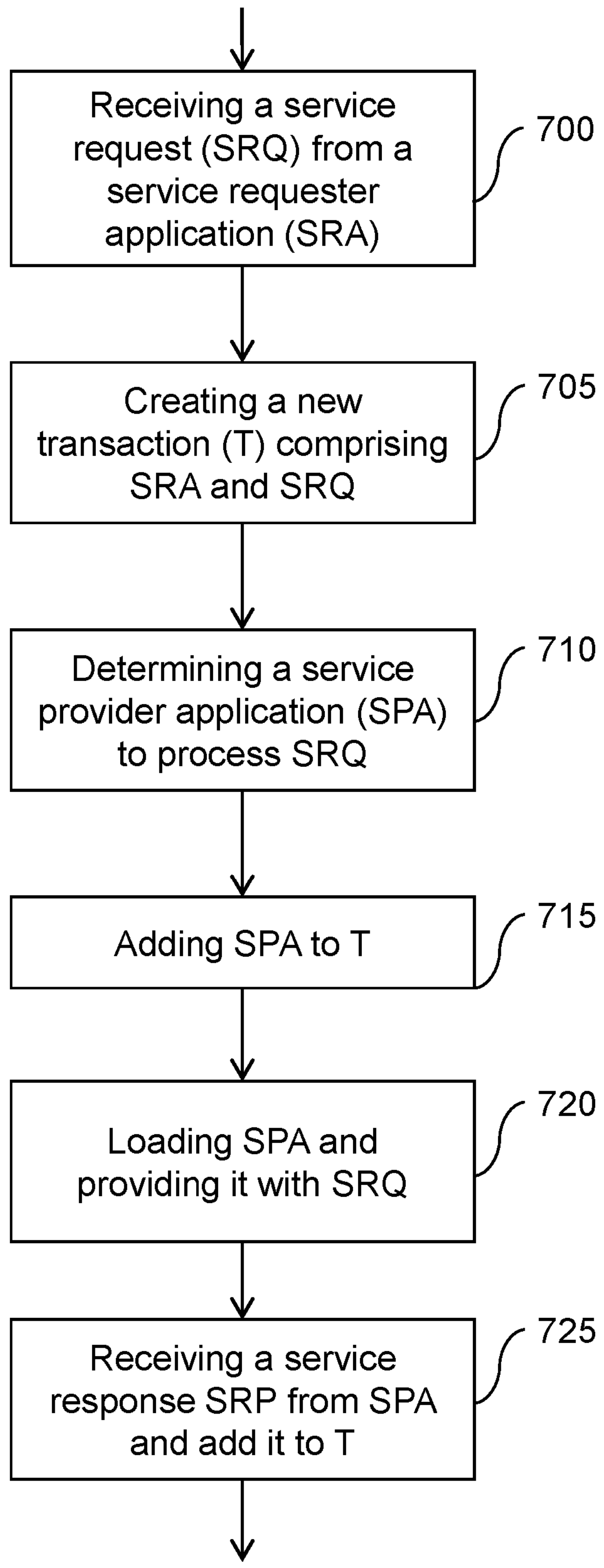

Receiving a service request (SRQ) from a service requester application (SRA)
700
Creating a new transaction (T) comprising SRA and SRQ
705
Determining a service provider application (SPA) to process SRQ
710
Adding SPA to T
715
Loading SPA and providing it with SRQ
720
Receiving a service response SRP from SPA and add it to T
725

Fig. 7

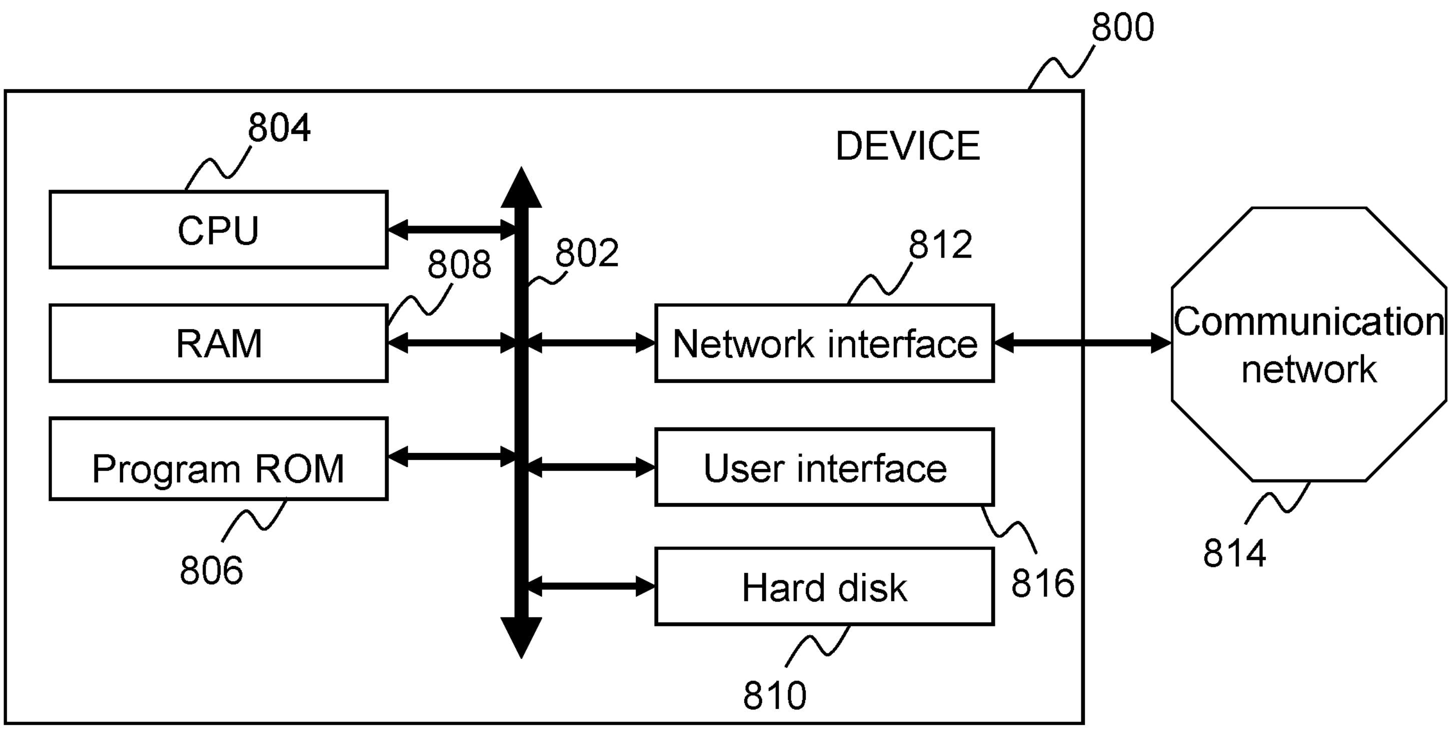

800
DEVICE
804
CPU
808
RAM
802
Program ROM
806
812
Network interface
User interface
816
Hard disk
810
Communication network
814

Fig. 8

METHOD, DEVICE, AND COMPUTER PROGRAM FOR SIMPLIFYING THE PROCESSING OF SERVICE REQUESTS IN A WEB RUNTIME ENVIRONMENT

FIELD OF THE INVENTION

The invention generally relates to the field of processing service requests in a web runtime environment. More particularly, the invention concerns a method for simplifying the processing of web intents or similar requested services by enabling a web runtime environment to store executed transactions or parts of executed transactions allowing a user to directly select a service request or a service response when executing a transaction. As a consequence, loading an application in the web runtime environment as well as interacting with the web runtime environment are not requested to create a service request or a service response in this application. Therefore, execution of transactions is faster and simpler.

BACKGROUND OF THE INVENTION

Sharing of content on the World Wide Web is now frequently made available through the use of "*share*" buttons. For instance, a person enjoying an article on a given website can generally use *Facebook*, *Twitter* or *Google+* buttons in order to share this article on the corresponding social network (Facebook, Twitter, and Google+ are trademarks).

However, since these buttons do not result from any standardization process, they are all defined differently using proprietary solutions and thus, developers have to integrate each button separately into websites. Therefore, if a reader uses a sharing service that has not been integrated by a website developer, he/she cannot share content from this website. Furthermore, a multiplicity of buttons leads to unfriendly user interfaces.

To handle such a problem, a solution consists in using a dedicated service that may be reached using a single button and that defines a communication model between a requester and a provider of that dedicated service.

For the sake of illustration, such a dedicated service can be a "*share*" service which defines a communication model for sharing data between a website publishing content ("*share*" service requester) and a service of a social network ("*share*" service provider such as *Facebook*, *Twitter* or, *Google+*). If a website declares itself as a "*share*" service requester (e.g. to enable its readers to share its articles on their favorite social network), a single "*share*" button is displayed along with shareable content. Accordingly, when a reader clicks on the "*share*" button, a list of "*share*" service providers (or advantageously a list of user's favorite "*share*" service providers) is displayed so that the reader selects one of them and the corresponding content is shared based on the communication model defined.

Such a solution makes it possible to avoid integrating different proprietary buttons into websites in order to enable the requesting of services. Moreover, only favorite service providers can be presented to a user thereby improving the user interface. Such a model is defined by elements known as web intents.

Web intents result from a standardization proposal submitted to W3C (World Wide Web Consortium) whose purpose is to enable a better cooperation between web applications by defining standard actions (web intents) such as share, pick, view or edit (a share action allows the sharing of content on a social network, a pick action is used for picking content such as an image from an online image gallery, a view action is typically an application adapted for viewing specific kinds of files, and an edit action is generally an application for editing content, for example editing an image).

The web intents proposal defines a generic model for interactions between web applications. Each defined action is described in a specification that explains how parameters of the generic model should be specifically used to handle that action.

The main components of web intents comprise an intent tag, an intent object, and a *startActivity()* method.

The intent tag (<*intent*>) is defined so that a website can declare itself as a provider for a given kind of action. For instance, the tag *<intent action="http://webintents.org/share" />* may be included in an HTML source code of a webpage (that is considered to be a web application) to indicate that the webpage supports the "*share*" action.

When a user-agent (i.e. typically a web browser) detects the presence of such a tag in the HTML code of a website, it preferably checks whether the current user has already registered this website as a provider for the corresponding action. If not, the possibility of adding this website to his/her list of registered providers for this action is offered to the user.

Additional attributes of the intent tag are available. They can be used to indicate, for example, the type of data a given website can handle (e.g. a website can define itself as able to handle the edit action but only for images) or to specify the URL (Uniform Resource Locator) of the page on the website that supports the action described in the intent tag (i.e. the URL of the page that the user-agent should load when the provider is selected by a user). Another attribute enables it to be specified whether a web intent provider should be loaded in a new window or directly inside the requesting web page.

It is noted that intent tags are key components for registration of providers, even though other mechanisms may be used to register providers (as an example, web application developers may be able to publish their applications in specific repositories designed to gather applications providing a given service).

A web intent object is typically a JavaScript object representing a request made by a client web application (Java is a trademark). When created, a web intent object contains the parameters of the request, which may include a kind of action (e.g. "*share*", "*pick*", "*view*" or "*edit*"), a data type information (e.g. image or sound), and data to be provided to the application selected by the user to process the request (e.g. the title and the URL of an article to be shared or image data to be edited). A web intent object also comprises two methods known as *postResult()* and *postFailure()* that can be called only when the web intent object has been delivered to a service provider application.

After a web intent object has been created, a web page can actually propose to a user, through a user-agent, to select a provider to process the request corresponding to the web intent object. To do so, it calls a *startActivity()* method that typically comprises up to three parameters among which a web intent object previously created.

Optionally, the *startActivity()* method may comprise a call-back reference to be associated with an event known as *onSuccess*, in order to identify a method that should be called in case the provider returns a success event. Similarly, *startActivity()* method may optionally comprise a call-back reference to be associated with an event known as *onFailure*, in order to identify a method that should be called in case the provider returns a failure event.

When executing the *startActivity()* method, the user-agent displays a list of identifiers of possible providers based on intent object properties. For instance, if the web intent object corresponds to "*edit*" action, a list comprising only identifiers of providers supporting "*edit*" action would be displayed. Moreover, if the "*edit*" action concerns an image, then a list comprising only identifiers of providers supporting "*edit*" action for processing images would be preferably displayed. The identifiers of the displayed list typically correspond to providers previously registered by the user. However, other identifiers may also be displayed within the list, e.g. identifiers of providers suggested by the web page or by the user-agent.

When the user selects one of the displayed identifiers, the corresponding provider is loaded. While processed by the user-agent, provider's web page is provided access to the web intent object. In the case of an image to be edited, the image is therefore accessible to the provider.

After completion of the request, for example after an image is edited and the user clicks on a "*complete*" button, the provider typically calls a *postResult()* or *postFailure()* method of the web intent object. As mentioned above, these methods are associated with the call-back references defined for "*onSuccess*" and "*onFailure*" events. Depending on the kind of web intent, the result may be a simple string value such as a successful processing indication or an object such as an image if the web intent object corresponds to an "edit" action for processing an image (the result then being the edited image).

Web intents provide several advantages among which is to allow a web application to request a given kind of service without knowledge of a provider to be selected for performing the request. Accordingly, web developers can easily rely on other web applications and users can select their favorite providers.

Moreover, given that only the user and the user-agent know which client and which provider are considered, the user's privacy can be ensured (with a prior art solution, e.g. for a "*share*" button, the provider knows which client is requesting an action to be performed).

Similarly, security is also improved in that exchanges of data through web intent objects occur between two web pages loaded in a user-agent, i.e. exchange of data occurs only in the browser. Therefore, there is no need for a client web application to communicate with a web server distinct from its own web server.

It is noted that web intents is not the only technology enabling cooperation between web applications. For the sake of illustration, *Mozilla* has developed a very similar technology called web activities (Mozilla is a trademark). This technology is in particular used in *Mozilla's Firefox OS* operating system (Firefox is a trademark).

Web intents and web activities can be considered as a specific implementation of a generic model handled by a web runtime environment (WRE), for example *Mozilla's Firefox OS* or a web browser, enabling web applications loaded and run in the web runtime environment to cooperate.

Figure 1, comprising Figures 1a and 1b, illustrates two examples of web runtime environments.

As illustrated in Figure 1a, a web runtime environment can be a software component 100 relying on an operating system 105 to run web applications 110. A web runtime environment typically consists of one or several software modules, for example generic modules and specific modules, among which a web runtime controller, which may comprise sub-modules. Web browsers like *Mozilla Firefox*, *Google Chrome* and *Microsoft Internet Explorer*, are examples of such a web runtime environment (Google, Chrome, Microsoft, and Internet Explorer are trademarks).

Alternatively, as illustrated in Figure 1b, a web runtime environment 100' can be integrated into an operating system 105' to run web applications 110'. In such a case, the whole operating system is based on web technologies. Examples of such a web runtime environment are *Google Chrome OS* and *Mozilla Firefox OS*.

Figure 2 illustrates an example of the cooperation between two applications executed in a web runtime environment 200.

As illustrated, a service requester application (SRA), referenced 205, transmits a service request to a web runtime controller 210. Such a service request has various characteristics and comprises a type, for example "*share*", "*edit*", "*pick*" or "*save*".

In response, web runtime controller 210 selects a service provider application (SPA), referenced 215, that is able to handle at least the service request corresponding to the type of the requested service. Once selected for processing a service request, a service provider application is loaded and provided with said service request. Upon completion of the service request, the service provider application can return a result to web runtime controller 210 which in turn returns it to the service requester application 205 that originated the service request.

As can be understood from Figure 2, a web runtime controller of a web runtime environment is responsible for receiving a service request from a service requester application, transmitting it to a service provider application, receiving a result from the service provider application, and transmitting the result to the service requester application.

It is noted that in practice, a web runtime controller may correspond to various components of a web runtime environment that may also be responsible for performing other tasks handled by the web runtime environment.

A web runtime environment conforming to those described by reference to Figures 1 and 2 executes transactions between a service requester application and a service provider application. Each transaction typically consists of a service request which is transmitted to a selected service provider application and a service response which is transmitted to the service requester application which originated the service request.

It has been observed that from a user point of view, a single transaction generally involves loading a service requester application, interacting with this application so that it creates a service request, selecting a service provider application based on web runtime environment proposals for possible service provider applications, and finally interacting with a selected service provider application in order to process the service request. After processing the service request, a service response is returned to the appropriate service requester application and the transaction is complete.

Therefore, although such a way of handling a service request may be convenient as it enables cooperation between different applications, it requires a significant number of user inputs.

As a consequence, there is a need for improving and simplifying the execution of web intents, web activities, and similar requested services in a web runtime environment.

SUMMARY OF THE INVENTION

Faced with these constraints, the inventors provide a method, a device and a computer program for simplifying the execution of service requests in a web runtime environment.

It is a broad object of the invention to remedy the shortcomings of the prior art as described above.

According to a first aspect of the invention there is provided a method for a web runtime environment to process a service request, the method comprising:

- accessing at least one part of at least one transaction, the at least one accessed part of the at least one transaction being previously stored and comprising at least one of one part of a service request and one part of a corresponding service response generated by a service provider application,
- displaying a list comprising the at least one accessed part of the at least one transaction;
- enabling the selection of at least one part of at least one transaction, in the displayed list, by a user, the at least one selected part of at least one transaction comprising at least one part of a service request or at least one part of a service response generated by a service provider application in response to a service request distinct from the service request being processed according to the method; and
- obtaining a service response to the service request being processed according to the method, the obtained service response being obtained as a function of the at least one selected part of at least one transaction.

Therefore, according to the method of the invention, processing a service request is simplified: fewer applications have to be loaded to carry out a service request (it enables loading of service requester applications or service provider applications to be avoided), which results in an improved user experience and a reduced footprint for the device implementing the method of the invention (faster processing, reduced battery consumption, etc.). It is especially convenient in case of errors in relation to a service request to be processed since the service request can be replayed easily (possibly with a different service provider application).

The steps of the method are advantageously carried out by the web runtime environment.

In an embodiment, the method further comprises a step of enabling the edition of the selected part of at least one transaction and a step of enabling validation of the edited selected part of at least one transaction.

In an embodiment, the step of enabling the edition of the selected part of at least one transaction comprises a step of modifying a value of a parameter of the selected part of at least one transaction, a step of adding a parameter to the selected part of at least one transaction and of setting a value to the added parameter, a step of removing a parameter from the selected part of at least one transaction, or a step of generating a service request comprising a value of at least one parameter of the selected part of at least one transaction.

In an embodiment, the value of the modified parameter or of the added parameter is determined as a function of a selected part of at least one transaction.

In an embodiment, the selected part of at least one transaction is at least one part of a service response generated by a service provider application in response to a service request distinct from the service request being processed according to the method, the method further comprising a step of receiving the service request to process.

In an embodiment, the method further comprises a step of determining the list comprising at least the at least one part of the at least one transaction.

In an embodiment, the list comprising at least the at least one part of the at least one transaction is determined as a function of a type of the received service request.

In an embodiment, the list comprising at least the at least one part of the at least one transaction is determined as a function of a type of data to be returned in response to the received service request.

In an embodiment, the method further comprising a step of creating the obtained service response based on the selected part of at least one transaction, the step of creating the obtained service response based on the selected part of at least one transaction being preferably carried out by the web runtime environment.

In an embodiment, the method further comprises a step of sending the obtained service response to a service requester application at the origin of the received service request.

In an embodiment, the selected part of at least one transaction is a service request, the method further comprising a step of determining the list comprising at least the at least one part of the at least one transaction.

In an embodiment, the method further comprises a step of sorting the determined list comprising at least the at least one part of the at least one transaction, the sorting step being carried out as a function of a type of the at least one transaction, of the data and time of execution of the at least one transaction, of a service requester application associated with the at least one transaction, and/or of a service provider application associated with the at least one transaction.

In an embodiment, the method further comprises a step of creating a service request based on the selected part of at least one transaction.

In an embodiment, the method further comprises a step of sending the created service request to a service provider application adapted to process the created service request, the step of sending the created service request to a service provider application adapted to process the created service request being advantageously carried out by the web runtime environment.

In an embodiment, the method further comprises a step of selecting the service provider application adapted to process the created service request.

In an embodiment, the method further comprises a step of receiving a service response from the selected service provider application in response to the step of sending the created service request.

In an embodiment, the method further comprises a step of displaying the received service response.

A second aspect of the invention provides a device for processing of a service request in a web runtime environment, the device comprising at least one microprocessor configured for carrying out the steps of:

- accessing at least one part of at least one transaction, the at least one accessed part of the at least one transaction being previously stored and comprising at least one of one part of a service request and one part of a corresponding service response generated by a service provider application,

- displaying a list comprising the at least one accessed part of the at least one transaction;

- enabling the selection of at least one part of at least one transaction, in the displayed list, by a user, the at least one selected part of at least one transaction comprising at least one part of a service request or at least one part of a service response generated by a service provider application in response to a service request distinct from the service request being processed; and

- obtaining a service response to the service request being processed, the obtained service response being obtained as a function of the at least one selected part of at least one transaction.

Therefore, according to the device of the invention, processing a service request is simplified: fewer applications have to be loaded to carry out a service request (it enables loading of service requester applications or service provider applications to be avoided), which results in an improved user experience and a reduced footprint for the device (faster processing, reduced battery consumption, etc.). It is especially convenient in case of errors in relation to a service request to be processed since the service request can be replayed easily (possibly with a different service provider application).

The steps carried out by the microprocessor of the device are advantageously carried out by the web runtime environment.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of enabling the edition of the selected part of at least one transaction and a step of enabling validation of the edited selected part of at least one transaction.

In an embodiment, the at least one microprocessor is configured so that the step of enabling the edition of the selected part of at least one transaction comprises a step of modifying a value of a parameter of the selected part of at least one transaction, a step of adding a parameter to the selected part of at least one transaction and of setting a value to the added parameter, a step of removing a parameter from the selected part of at least one transaction, or a step of generating a service request comprising a value of at least one parameter of the selected part of at least one transaction.

In an embodiment, the value of the modified parameter or of the added parameter is determined as a function of a selected part of at least one transaction.

In an embodiment, the selected part of at least one transaction is at least one part of a service response generated by a service provider application in response to a service request distinct from the service request being processed, the at least one microprocessor being further configured for carrying out a step of receiving the service request to process.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of determining the list comprising at least the at least one part of the at least one transaction.

In an embodiment, the list comprising at least the at least one part of the at least one transaction is determined as a function of a type of the received service request.

In an embodiment, the list comprising at least the at least one part of the at least one transaction is determined as a function of a type of data to be returned in response to the received service request.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of creating the obtained service response based on the selected part of at least one transaction, the step of creating the obtained service response based on the selected part of at least one transaction being advantageously carried out by the web runtime environment.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of sending the obtained service response to a service requester application at the origin of the received service request.

In an embodiment, the selected part of at least one transaction is a service request, the at least one microprocessor being further configured for carrying out a step of determining the list comprising at least the at least one part of the at least one transaction.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of sorting the determined list comprising at least the at least one part of the at least one transaction, the sorting step being carried out as a function of a type of the at least one transaction, of the data and time of execution of the at least one transaction, of a service requester application associated with the at least one transaction, and/or of a service provider application associated with the at least one transaction.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of creating a service request based on the selected part of at least one transaction.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of sending the created service request to a service provider application adapted to process the created service request, the step of sending the created service request to a service provider application adapted to process the created service request being advantageously carried out by the web runtime environment.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of selecting the service provider application adapted to process the created service request.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of receiving a service response from the selected service provider application in response to the step of sending the created service request.

In an embodiment, the at least one microprocessor is further configured for carrying out a step of displaying the received service response.

Since the present invention can be implemented in software, the present invention can be embodied as computer readable code for provision to a programmable apparatus on any suitable carrier medium. A tangible carrier medium may comprise a storage medium such as a floppy disk, a CD-ROM, a hard disk drive, a magnetic tape device or a solid state memory device and the like. A transient carrier medium may include a signal such as an electrical signal, an electronic signal, an optical signal, an acoustic signal, a magnetic signal or an electromagnetic signal, e.g. a microwave or RF signal.

BRIEF DESCRIPTION OF THE DRAWINGS

Further advantages of the present invention will become apparent to those skilled in the art upon examination of the drawings and detailed description. It is intended that any additional advantages be incorporated herein.

Embodiments of the invention will now be described, by way of example only, and with reference to the following drawings in which:

Figure 1, comprising Figures 1a and 1b, illustrates two examples of web runtime environments;

Figure 2 illustrates an example of the cooperation between two applications executed in a web runtime environment;

Figure 3, comprising Figures 3a and 3b, illustrates examples of using data of previously executed transactions to process a new service request;

Figure 4 illustrates an example of steps executed by a web runtime environment for simplifying the execution of service requests by using stored transactions or parts of transactions;

Figure 5 illustrates an example of steps of an algorithm for using previously stored service responses to create a new one;

Figure 6 illustrates steps of an example of an algorithm for using previously stored service requests to create a new one;

Figure 7 illustrates steps of an example of an algorithm enabling a web runtime environment to store executed transactions; and

Figure 8 represents a block diagram of a computer device in which steps of one or more embodiments may be implemented.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

A particular embodiment of the invention aims at simplifying the execution of service requests, or more generally of transactions, in a web runtime environment, by storing transactions or parts of transactions and by using stored data (transactions or parts of transactions) to simplify execution of new service requests. This is typically done by the web runtime environment.

According to a first embodiment, when a service request is to be processed, the web runtime environment searches for service responses that were previously returned and stored for similar service requests (similar service requests are typically service requests corresponding to the same kind of services). Then, instead of only enabling the user to select a service provider application, the web runtime environment also enables the user to select a previous service response or at least some data that were part of a previous service response (e.g. an image). If the user selects a previous service response, the web runtime environment creates a new service response based on user-selected data and sends it to the service requester application. Therefore, the overall process is simplified for the user as there is no need to select a service provider application to process a service request.

Figure 3a illustrates an example of using results of previously executed service requests to process a new service request, simplifying the way a user obtains a result for a service request. According to the illustrated example, it is considered that a service requester application requests an image by creating a "*Pick*" service request. Such an application can be, for instance, a photo editing application or a photo album application.

According to a standard solution, the web runtime environment displays a list of possible service provider applications when such a service request is received and the user selects one of them. If such a list is still displayed when carrying out the invention according to the first embodiment, as illustrated with reference 300, a list of recently used images, denoted 305, is also displayed. The images of the list 305 represent results of previously executed service requests.

For the sake of illustration, the list 300 comprises three examples of possible service provider applications, denoted 310, 315, and 320, corresponding to the Facebook, Flickr, and Google+ applications, respectively (Facebook, Flickr, and Google+ are trademarks). As illustrated, in addition to these service provider applications, the web runtime environment displays the list 305 of the images which have been previously returned by the "*Pick*" service provider application in response to a "*Pick*" service request for an image and stored by the web runtime environment in a list of previously executed transactions. In this example, three images are displayed in a single row (even though they are associated with different service responses).

Naturally, other representation designs can be implemented for displaying service responses (for instance one row per service response). Likewise, if recently used images can be displayed, other selection schemes can be used for selecting a subset of previous service responses (e.g. most frequently used images).

If a user selects one of the images from the list 305, the web runtime environment relies on a corresponding stored service response to create a new service response which comprises the same image. This service response is then sent to the service requester application, which obtains a service response even though no service provider application has been loaded to process the service request. Compared to the prior art, the processing of the service request is therefore faster, has a smaller footprint on the device and requires fewer user inputs.

An example of implementation of such an embodiment is described with reference to Figure 5.

According to a second embodiment, a user can easily replay a previously executed service request without having to load the corresponding service requester application. To that end, the web runtime environment enables a user to select a previously executed service request (or part of a previous service request) and to replay it (or execute a new service request based on a previous service request). For the sake of illustration, the user can access previously executed requests thanks to a dedicated component of the web runtime environment user interface that enables a list of previously executed transactions (or part of those transactions) to be displayed.

After a user has selected a previously executed service request and data to be processed, the web runtime environment creates a service request. Depending on a user input, the web runtime environment requests the user to select a service provider application prior to sending the created service request or skips this step of selecting a service provider application and sends the created service request to the service provider application used to process the original service request. Therefore, since there is no need to load a service provider application to create a service request, the overall process is simplified for the user.

Figure 3b illustrates an example of using previously executed service requests to process a service request, simplifying the way a user obtains a result for a service request. According to the illustrated example, it is considered that a user wants to re-print a document he/she has already printed using a "*Print*" service provider application.

It is to be recalled that for handling such an action according to a standard solution, a user reloads the service requester application that created the "*Print*" service request, requests the printing of the appropriate document from this service requester application, selects a "*Print*" service provider application from a list of possible service provider applications displayed by the web runtime environment, and finally starts the printing process from the selected "*Print*" service provider application.

According to this embodiment of the invention, the web runtime environment provides a specific user interface component (e.g. a button) enabling a user to display a list of previously executed transactions.

It is noted that those transactions have not necessarily been executed in the same web runtime environment. Indeed, a user may use different devices, each running a different web runtime environment that can be synchronized with others (i.e. the stored transactions are synchronized between the different web runtime environments of a user).

Figure 3b illustrates such a list of previously executed transactions. According to the illustrated example, each type of service is associated with a particular tab: services of the "*Print*" type correspond to tab 350, services of the "*Upload*" type correspond to tab 355, and services of the "*Filter*" type correspond to tab 360. It is noted that an advantage of such a representation is that it enables a user to easily select the transactions based on their types of services.

As illustrated, the "*Print*" service type tab is currently selected and thus, the previously executed "*Print*" transactions are displayed. For the sake of illustration, three transactions are visible: the transactions 365 and 370 correspond to "*Print*" requests from the "*Album*" application to the "*Canon Printer*" application and the transaction 375 corresponds to a "*Print*" request from the "*Editor*" application to the "*Cloud Printer*" application.

Upon selection of one of the displayed transactions by a user, the web runtime environment creates a new service request based on the selected transaction.

Depending on the implementation, this service request is sent to the service provider application selected to process the original service request or the web runtime environment enables the user to select a service provider application for this newly created service request.

Advantageously, the web runtime environment provides both options to users, for instance by providing two distinct buttons (one for reusing the same service provider application and one for using a different one).

In a situation like the one described by reference to Figure 3b, it may be envisioned that when a previously executed transaction is selected, a list of possible service provider applications is displayed, the service provider application associated with the selected transaction being preselected. If the user does not modify the selection for a predetermined period of time, for example two seconds, the preselected service provider application is actually used. Conversely, if the user modifies the selection and confirms the new selection, the newly selected service provider application is used.

Again, compared to prior art, the processing of the service request is faster, has a smaller footprint on the device and requires fewer user inputs.

An implementation example of such an embodiment is described with reference to Figure 6.

Figure 4 illustrates an example of steps executed by a web runtime environment for simplifying the execution of service requests by using stored transactions or parts of transactions.

A first step (step 400) aims at determining a set of previously stored transactions and/or parts of transactions (e.g. service requests and/or results of executed service requests) corresponding to previously executed transactions. As mentioned above, since a user can use different devices, each running a different web runtime environment that can be synchronized with others (i.e. the stored transactions and/or parts of transactions are synchronized between the different web runtime environments of the user), those transactions may have been executed in different web runtime environments. An example of steps for storing transactions is described in reference to Figure 7.

According to a particular embodiment, stored data for a given transaction comprise a reference to a service requester application, a reference to a service request, a reference to a service provider application and a service response.

The determination step 400 typically consists in filtering a list of stored transactions based on a service type. Figures 5 and 6 illustrate how a service type may be obtained by a web runtime environment.

Next, at step 405, the web runtime environment enables a user to select at least one service request or at least one service response from the determined set of previously stored transactions and/or parts of transactions. The selected data are to be used in the context of the current transaction to create a service request or a service response.

Next, after having carried out this selection step, the web runtime environment enables the user to edit the selected data (step 410), giving the ability to modify some of their parameters and/or to add parameters not comprised in the selected data. For the sake of illustration, a user may select a service request at step 405 but not select the corresponding service provider application that was previously used to process the selected service request. In such a case, step 410 may enable the user to add a reference to the service provider application to be used for processing the service request to be created.

Steps 405 and 410 are described in more detail by reference to Figures 5 and 6, more precisely by reference to steps 510 and 515 and to steps 600 and 605, respectively.

It is noted here that even though step 405 involves selecting a service request or a service response, the user does not necessarily have to select a whole service request or a whole service response. For example, if a service response comprises an image, the user may simply select that image. However, from the web runtime environment point of view, the selection of the image corresponds to the selection of the corresponding service response.

More generally, step 405 is achieved by displaying transactions or parts of transactions to a user so that the user may select at least one part of a displayed transaction (or part of transaction). Being associated to a transaction, that selected part is necessarily associated to a given service request or a given service response, which can therefore be used as a basis to create a new service request or a new service response depending on the context and/or a user input. In particular, if a service request or a service response comprises a plurality of objects (e.g. a set of images), a user may be able to select a single object to create a corresponding service request or service response comprising only that single object.

It is also noted that a user can choose data from other transactions when editing the selected data. For example, a user can select a "*Print*" service request associated with a given printing service application but then replace the data to print comprised in the service request by other data coming from another service request or from a service response. Such operations may advantageously be made through a graphical user interface, for example by using a drag-and-drop function.

According to the given example, step 410 is followed by a validation step (step 415) which allows a user to validate the data to be used for the current transaction (typically the data necessary to create a service request – possibly along with a reference to a service provider application – or a service response).

Next, at step 420, it is determined whether or not the validated data correspond to a service request.

If the validated data correspond to a service request, a new service request is created by the web runtime environment based on the validated data (step 425). The newly created service request is then sent to the selected service provider application (step 430). It is noted that if no service provider application was provided by the user in the validated data, a service provider application selection is performed after step 425. Such a selection is made in a standard way, for example by enabling the user to select a service provider application from among a list of possible service provider applications displayed by the web runtime environment. The process then ends.

On the contrary, if the validated data do not correspond to a service request (step 420), this means, in the illustrated example, that the validated data correspond to a service response. In such a case, a new service response is created by the web runtime environment based on the validated data (step 435). The newly created service response is then sent to the appropriate service requester application (step 440) and the process then ends.

It is noted here that the editing and validating steps 410 and 415 can be made optional in certain implementations. As a matter of fact, a web runtime environment may ignore those steps in order to make the process even faster for a user to complete a transaction. Determining whether or not steps 410 and 415 have to be executed can be defined by a specific setting parameter accessible to the users. Alternatively, a user interface can be used to let a user decide whether or not the editing and validating steps have to be executed for each execution of the process (for instance by having two different buttons or options in a menu).

Figure 5 illustrates an example of steps of an algorithm for using previously stored service response to create a new one.

According to the given example, the process starts at step 500 where a service request is received by the web runtime environment from a service requester application. Next, at step 505, a set of possible service responses is determined from previously stored service responses. The latter correspond to previously executed service responses and typically belong to previously stored transactions and/or parts of transactions.

Basically, only service responses corresponding to service requests of the same type as that of the received service request are proposed to the user to create a service response to the received service request. However, since some of the service responses corresponding to service requests of that type may be invalid service responses to the received service request, these service responses are advantageously filtered. For example, a service requester application creating a "*Pick*" service request for an image expects an image to be returned. Therefore, in such a case, it is generally advantageous to filter to obtain only service responses associated with "*Pick*" service requests for images and to discard service responses associated with "*Pick*" service requests for other types of data (e.g. video, sound or textual data). As a consequence, in addition to the service type, some parameters comprised in the received service request may also be relied on to determine a set of possible service responses, such as the data type.

Nevertheless, it may happen in particular cases that service responses of different types than that of the currently processed service request have to be used. For example, if a service request of a first type A and a service request of a second type B rely on service responses that share the same parameters, then a service response of the service request of type B may be used in response to a service request of type A.

Similarly, if the main parameters of a service response of a service request of type B are the same as those of a service response of a service request of type A (even though some parameters may be different), a service response of a service request of type B may be used in response to a service request of type A. As an example, if a service request of type B returns an image and a service request of type A returns an image along with a file name, the web runtime environment may enable a user to select a service response of a service request of type B for a service request of type A. It is noted that in such a case, the user may be requested to input a value for the missing parameter (i.e. the file name).

Next, at step 510, the web runtime environment enables the user to select a service response. As previously described, the user does not necessarily have to select the whole service response. For example, if a service response comprises an image among several items of data, the web runtime environment may simply display that image to represent that service response. Then, from the user point of view, only an image is to be selected. However, from the web runtime environment point of view, the selection of the image corresponds to the selection of the corresponding service response that may comprise the name of the image and metadata comprising a type, a size, a coding format and so on.

It is noted that the selection of a service response can be facilitated by an appropriate representation of the determined set of service responses. For example, a named image can be suggested to the user by displaying its name or by displaying the image itself. From the user point of view, displaying the image is generally more helpful than the name of the image. To provide more details, a thumbnail of the images along with their names can be displayed by the web runtime environment. Similarly, if a video or a sound file is comprised in a service response, enabling a user to play this resource is likely to help the user in selecting a service response.

After a service response has been selected, the web runtime environment enables the user to edit the selected service response at step 515. As mentioned above, a service response typically comprises several parameters, for example an image along with a file name and/or metadata. Therefore, step 515 allows the user to modify the file name or the metadata. Moreover, if some parameters are optional, the user can remove data from the service response and/or add data to the service response. Similarly, if the service response comprises a set of images, the user can select only a subset of images during step 515.

Next, after having edited and modified the selected service response, if needed, the user validates the selected data at step 520.

These validated data are then used by the web runtime environment to create a new service response (step 525) which is sent to the service requester application that has created the currently processed service request (step 530), that is to say the service request received at step 500, and the process then ends.

It is noted that the service response created at step 525 can be based on more than one service response. For example, if the currently processed service request is a "*Pick*" service request for a set of images, the user may be able to select images from different service responses to build a new set of images. The selection of images from several service responses may be done at editing step 515. Alternatively, instead of selecting a single service response at step 510, a user may be able to select different service responses. In such a case, the firstly selected service response may be used as a basis for the new service response to create and the other selected service responses may be integrated into the service response to be created. In the context of a service response comprising a set of images, images from several service responses can be added to the set of the firstly selected service response. Accordingly, a single response comprising a set of images obtained from different service responses can be obtained.

As mentioned above, the steps of editing a selected service response and of validating edited data can be optional. If these steps are not carried out, the selection of a service response by the user (step 510) is preferably directly followed by the creation of a new service response (step 525).

Figure 6 illustrates steps of an example of an algorithm for using previously stored service requests to create a new one.

A first step aims at enabling the selection of a previously executed and stored service request (step 600). This can be achieved by displaying a list of previously executed transactions. This step is typically carried out by the web runtime environment upon request from a user through a specific graphical component (for example a button enabling access to previously executed transactions).

Previously executed transactions can be displayed as a list which can be sorted, for example, as a function of the type of the service request associated with each transaction. For the sake of illustration, different types of service requests can be displayed in different tabs, as illustrated in Figure 3b. Moreover, the service requests may be sorted as a function of the date and time of execution and/or as a function of the associated service requester application and/or the associated service provider application. The user may also be able to bookmark some service requests.

Another criterion that can be used to sort transactions is the value of one or more parameters comprised in the service requests or in the service responses. For example, all the transactions corresponding to a "*Pick*" service request for images can be displayed consecutively in the list of transactions.

Naturally, more than one criterion can be used to sort displayed transactions (e.g. both a service type criterion and a service requester application criterion) and a user may be able to modify the order of the list by selecting sorting criteria.

If many transactions are stored, a subset may be initially displayed (e.g. the five most recent transactions if execution time is selected as the main criterion).

It is noted that even though step 600 aims at selecting a service request, most of the data available and associated with a given transaction (i.e. typically a service requester application, a service request, a service provider application and a service response) are advantageously displayed (not only the service requests).

Indeed, providing more information to the user may help him/her in the selection process.

For example, displaying a service requester application and a service provider application may provide a context that is helpful for a user as a reminder of a previous request he/she wants to replay or re-use as the basis for another service request.

Regarding the service requests and the service responses, only some of their parameters are preferably displayed. For example, in the case of a "*Print*" service request, the name of the printed resource and/or a rendering of this resource (e.g. a thumbnail for an image) may be displayed. Similarly, in the case of a "*Pick*" service response, the name of the picked resource and/or a rendering of this resource may be displayed. If the service response simply comprises an indication as to whether or not the service request has been correctly processed, this indication may be displayed by using a specific background color for the corresponding transaction (e.g. green for a successful request and red for a failure). If a transaction failed, a user may want to easily find this transaction and replay it, possibly with a different service provider application. Correlatively, knowing that a service provider application has processed successfully all of the submitted service requests may prove that this service provider application is reliable which is a useful indication for a user to select it. Finally, displaying the service provider application may enable a user to indicate that the same service provider application is to be used for processing the new service request (which still has to be created at that stage).

After selecting a previously executed and stored service request (step 600), the web runtime environment enables a user to edit the selected data (step 605). This editing step gives to a user the possibility of modifying some parameters of the selected service request as well as to add some parameters which were not present in the service request. For example, if the service request is a "*Print*" service request for a set of images, the user may be able to add more images to the set of images, these images being possibly obtained from other previously executed service requests.

If optional parameters are defined for the type of the considered service request, these optional parameters may be added and/or removed by the user.

Advantageously, the editing step further comprises a step of defining a service provider application for processing the service request to be created. If this is not the case and if no service provider application is selected by the user, a specific step of selecting a service provider application is to be executed by the web runtime environment which can provide a list of possible service provider applications to the user so that he/she selects one of them.

The editing step may also give to a user the possibility of creating a new service request with previously used parameters. For the sake of illustration, a printing service request may be created based on a previously executed editing service request, by using the same parameter values (e.g. the same image references).

Next, the user validates the selected data (step 610) and the web runtime environment creates a new service request based on the selected and validated data (step 615). The new service request is sent to the selected service provider application (step 620).

At this point, the created service request is being processed by the service provider application on a standard basis (generally requiring user inputs) and a service response is returned by the service provider application to the web runtime environment. When received, the service response is displayed to the user (step 625) and the process ends.

It is noted that contrary to prior art execution of a transaction, the service request is created by the web runtime environment (and not by a service requester application). As a consequence, the received service response cannot be sent to a service requester application to be displayed to the user and therefore this task is carried out by the web runtime environment.

In the case of a "*Print*" service request, displaying the service response may simply consist in displaying an indication of success or failure of the processing of the service request. In addition, the name of the printed document or a preview of this document may also be displayed so that the user can easily determine to which data the indication relates. Such an indication may be displayed in a specific window (e.g. a pop-up window) or through an alternative mechanism (e.g. in a notification area of the web runtime environment user interface).

It is observed here that the algorithm illustrated in Figure 6 is particularly adapted to service requests for which the responses mainly comprise an item of information to indicate whether or not the corresponding service request has been successfully processed (e.g. a "*Print*" service request). In other words, the algorithm illustrated in Figure 6 is particularly adapted to service requests for which the responses do not comprise results used by service requester applications to carry out other tasks (such as a service response comprising an image for a service requester application that relies on this image). This results from the fact that no service requester application is used to create the service request hence, no service requester application can be used to handle the data comprised in the service response.

It is noted here that the steps of editing a selected service request and of validating the edited data may be optional. In such a case, the selection of a service request by a user (step 600) is directly followed by the creation of a new service request (step 615).

It is also noted that even if previously executed transactions are generally displayed upon a request from a user, other circumstances may lead to that displaying step. In particular, if processing a service request by a service provider application fails, a failure result is generally returned to the service requester application through the web runtime environment. Therefore, such a failure can trigger a function of the web runtime environment. Such a function can be directed to automatically displaying the current transaction (possibly along with other transactions) so that the user can easily re-execute a similar service request, possibly with a different service provider application.

If the failure is detected by the web runtime environment before the service request is processed by the service provider application, for instance because the loading of said service provider application failed, the same process may be applied (i.e. display of the at least failed transaction to the user so that the service request can be re-executed, preferably with a different service provider application).

An algorithm similar to the one described by reference to Figure 6 can be used to enable selection of a service response by a user, that service response then being edited, and validated so as to create and send a new service response to a service requester application. However, since a service requester application is generally supposed to create a service request prior to receiving a service response, it is unlikely that the latter will be properly handled by the service requester application. Nevertheless, if the web runtime environment has the ability to determine that a given service requester application can handle a service response even if it has not created a service request, such an algorithm can be useful. This may be possible by enabling service requester applications to indicate that they can handle such service responses. Such an indication can be contained in the service requester application source code, for example in its HTML code or in an application manifest.

It is noted that if a user wants to re-execute a service request which has been stored by the web runtime environment with exactly the same parameters (including the service provider application), exactly as done previously, it is not necessary to edit the service request parameters. Therefore, it may be advantageous for the web runtime environment to enable the replaying of the service request with a single input from user.

To that end, the web runtime environment can display a specific graphical component, for instance a button displayed along with each displayed transaction. Alternatively, if the user interacts with the representation of the transaction instead of interacting with that button, the editing and validating steps (steps 605 and 610) are carried out.

In addition, a specific parameter may be added to the service request to inform the service provider application that no user input is needed to process the service request. For example, if the service provider application usually requires a confirmation from the user to launch the processing of the service request through a "*Start processing*" button, such a specific parameter enables the user to avoid that confirmation step. This is particularly useful in case the web runtime environment has stored the parameters input by the user when the original service request was processed by the service provider application (e.g. in case of a "*Print*" service request, such parameters may comprise the number of copies or the color settings). Accordingly, in such a case, all the needed parameters can be set appropriately and automatically to replay the exact same service request with the same service provider application.

It is also noted that the list of previously executed transactions can be adapted to the current context. In particular, when the web runtime environment displays a service provider application for a given type of service request, the service requests of that type may be made easily accessible to a user, for instance through a dedicated button displayed only in such circumstances (the web runtime environment runs a service provider application for a type of service request which corresponds to the type of service requests of some stored service requests). In addition, if a service request of a given type is selected while a service provider application of the same type is being executed by the web runtime environment, this service provider application may be selected as the default service provider application.

Figure 7 illustrates steps of an example of an algorithm enabling a web runtime environment to store executed transactions and/or parts of them.

As illustrated, a first step (step 700) is directed to receiving a service request, denoted SRQ, from a service requester application, denoted SRA, in a web runtime environment. Next, at step 705, the web runtime environment creates a new transaction object, denoted T, representing a current transaction. The created transaction is stored in the web runtime environment. The received service request (SRQ) and a reference to the service requester application (SRA) from which it has been received are added to the created and stored transaction T.

In a following step, the service provider application, denoted SPA, used to process the received service request (SRQ) is determined (step 710). Such a determination step is typically achieved by displaying a list of possible service provider applications so that a user may select one of them. Alternatively, in certain circumstances, the service provider application to be used can already be comprised in the received service request (SRQ).

Once the service provider application (SPA) has been determined, a reference to that service provider can be added to the created and stored transaction T (step 715). That service provider application (SPA) is loaded and provided with the received service request (SRQ) to be processed (step 720). Next, at step 725, the service response, denoted SRP, corresponding to the received service request (SRQ), is received by the web runtime environment from the service provider application (SPA). That service response (SRP) is added to the created and stored transaction T and returned to the service requester application (SRA).

It is noted that an indication as to whether or not the service response (SRP) has been successfully delivered to the service requester application (SRA) can also be stored in the created and stored transaction T. Indeed, even though this delivery is unlikely to fail, it may happen in some cases (e.g. if service requester application and service provider application are displayed in two different tabs and if the tab of the service requester application has been closed by a user while the received service request was being processed by the service provider application). Storing such information in the created and stored transaction T may be helpful for a user to identify efficiently a transaction that failed among other stored transactions.

It is also noted that created and stored transactions do not need to comprise all the data described with reference to Figure 6. Typically, as soon as a service request is received from a service requester application, a transaction can be created and stored within the web runtime environment. However, if a user cancels the received service request or if an error occurs while selecting the service provider application to be used, there is no reference to a service provider application and no service response to store along with the transaction. Nevertheless, storing such a transaction can be useful for simplifying re-execution of the received service request.

Naturally, when the execution of a received service request fails, only available information is displayed (e.g. only a reference of the service requester application and the service request, possibly along with an indication that an error occurred). In addition, such a transaction may be displayed in a specific way so as to highlight the fact that it was not successfully completed.

In order to reduce the amount of data to be stored within a web runtime environment, it can be decided not to store some transactions or parts of transactions.

As a matter of fact, not all the data that can be stored in a transaction are of equal interest to a user. For instance, a "*Pick*" service request for an image is unlikely to be re-executed by a user since such a service request is typically only useful in the context of a service requester application that requests an image for a specific reason (e.g. editing that image or creating a photo album). Re-executing such a service request without a service requester application means that the returned service response would not be used by any service requester application (i.e. it would not be used to do anything). As an alternative, in such a case, the service requester application corresponding to the service request can be loaded, but given that this service requester application does not expect to receive a service response (since it has not created a service request), an error may occur as described above. Nevertheless, even in such a case, storing parts of the service request can be useful. For example, storing the type of a service request (e.g. "*Pick*") and the type of requested data (e.g. "*image*") is helpful for selecting possible service responses for a given service request (as described by reference to step 505 in Figure 5).

Similarly, storing a service response that simply comprises an indication as to whether or not the service request has been successfully processed may seem to be unnecessary since it is unlikely to be reused by a user to create another service response. However, storing such an indication can prove to be helpful as it enables the web runtime environment to adapt the display of corresponding transactions (e.g. using a red background for transactions that failed and a green background for transactions that succeeded). In such cases, a Boolean value indicating whether or not a service request was successfully processed can be stored instead of a whole service response.

It is noted that, as described above, a web runtime environment may decide to store some parameters input by a user in a service provider application in order to process a received service request. By doing so, the web runtime environment may enable a user to replay a previously executed service request with a single input (given that it would be able to set parameter values required by the service provider application).

Furthermore, a web runtime environment may decide to store only specific types of service requests (e.g. only "*Pick*" or "*Print*" service transactions). Advantageously, the web runtime environment enables the users to define the types of service requests for which transactions are stored and/or the types of service requests for which transactions are not stored.

It is also noted that service requester applications and service provider applications may be enabled to indicate which parts of service requests and of service responses may be stored in a web runtime environment. Such indication may be contained in the source code of the service requester applications or the service provider application (e.g. in their HTML code or in an application manifest).

For the sake of illustration, a service response may comprise a temporary URL or a URL intended to be used only once (e.g. a URL intended to obtain some credentials). In such a case, the URL may not be stored in the web runtime environment or only for a short period of time. As another example, if a camera hosts a service provider application displaying images, the URLs associated with those images are unlikely to be stable given that they depend on the current network parameters. Moreover, images on a camera may frequently be deleted. Therefore, such URLs may be indicated as not stable and should not be preserved for a long period of time in a web runtime environment. However, in the context of a local network, URLs may be preserved as long as the hosting device is accessible via the network.

According to a particular embodiment, the means used to display a list of stored transactions and enabling a user to select a service request or a service response in that list can also be used by the user for selecting one or several stored transactions to be removed from that list (and preferably deleted from the web runtime environment). This allows a user, in particular, to reduce the number of displayed transactions which is likely to make selection easier.

According to a particular embodiment, transactions and/or some parts of transactions are stored in a remote device, typically a web server, so that each device running a web browser environment does not have to store all the transactions and/or all the data corresponding to each transaction. For the sake of illustration, a service response comprising a large document may be stored on a web server and obtained only if actually used by a new transaction.

Alternatively, certain service provider applications can also store service response data obtained as a result of the execution of service requests. Such service provider applications may include identifiers for identifying given service requests inside the corresponding service responses. The web runtime environment may then store this identifier instead of storing all the data comprised in the service request. Later on, if corresponding service request is to be replayed, the web runtime environment sends that identifier to that service provider application. This solution enables minimization of the amount of data stored in a web runtime environment but requires adequate support from service provider applications.

Figure 8 represents a block diagram of a computer device 800 in which steps of one or more embodiments may be implemented.

Preferably, the device 800 comprises a communication bus 802, a central processing unit (CPU) 804 capable of executing instructions from program ROM 806 on powering up of the device, and instructions relating to a software application from main memory 808 after the powering up. The main memory 808 is for example of Random Access Memory (RAM) type which functions as a working area of CPU 804 via the communication bus 802, and the memory capacity thereof can be expanded by an optional RAM connected to an expansion port (not illustrated). Instructions relating to the software application may be loaded to the main memory 808 from a hard-disk (HD) 810 or the program ROM 806 for example. Such software application, when executed by the CPU 804, causes the steps described with reference to Figures 4 to 7 to be performed in the computer device.

Reference numeral 812 is a network interface that allows the connection of the device 800 to the communication network 814. The software application when executed by the CPU 804 is adapted to react to requests received through the network interface and to provide data and requests via the network to other devices.

Reference numeral 816 represents user interfaces to display information to, and/or receive inputs from, a user.

It should be pointed out here that, as a variant, the device 800 for processing data can consist of one or more dedicated integrated circuits (ASICs) that are capable of implementing the method as described with reference to Figures 4 to 7.

Naturally, in order to satisfy local and specific requirements, a person skilled in the art may apply to the solution described above many modifications and alterations all of which, however, are included within the scope of protection of the invention as defined by the following claims.

CLAIMS

1. A method for a web runtime environment to process a service request, the method comprising:

- accessing at least one part of at least one transaction, the at least one accessed part of the at least one transaction being previously stored and comprising at least one of one part of a service request and one part of a corresponding service response generated by a service provider application,

- displaying a list comprising the at least one accessed part of the at least one transaction;

- enabling the selection of at least one part of at least one transaction, in the displayed list, by a user, the at least one selected part of at least one transaction comprising at least one part of a service request or at least one part of a service response generated by a service provider application in response to a service request distinct from the service request being processed according to the method; and

- obtaining a service response to the service request being processed according to the method, the obtained service response being obtained as a function of the at least one selected part of at least one transaction.

2. The method of claim 1 further comprising a step of enabling the edition of the selected part of at least one transaction and a step of enabling validation of the edited selected part of at least one transaction.

3. The method of claim 2 wherein the step of enabling the edition of the selected part of at least one transaction comprises a step of modifying a value of a parameter of the selected part of at least one transaction, a step of adding a parameter to the selected part of at least one transaction and of setting a value to the added parameter, a step of removing a parameter from the selected part of at least one transaction, or a step of generating a service request comprising a value of at least one parameter of the selected part of at least one transaction.

4. The method of claim 3 wherein the value of the modified parameter or of the added parameter is determined as a function of a selected part of at least one transaction.

5. The method of any one of claims 1 to 4 wherein the selected part of at least one transaction is at least one part of a service response generated by a service provider application in response to a service request distinct from the service request being processed according to the method, the method further comprising a step of receiving the service request to process.

6. The method of claim 5 further comprising a step of determining the list comprising at least the at least one part of the at least one transaction.

7. The method of claim 5 or claim 6 wherein the list comprising at least the at least one part of the at least one transaction is determined as a function of a type of the received service request.

8. The method of any one of claims 5 to 7 wherein the list comprising at least the at least one part of the at least one transaction is determined as a function of a type of data to be returned in response to the received service request.

9. The method of any one of claims 5 to 8 further comprising a step of creating the obtained service response based on the selected part of at least one transaction.

10. The method of any one of claims 5 to 9 further comprising a step of sending the obtained service response to a service requester application at the origin of the received service request.

11. The method of any one of claims 1 to 4 wherein the selected part of at least one transaction is a service request, the method further comprising a step of determining the list comprising at least the at least one part of the at least one transaction.

12. The method of claim 11 further comprising a step of sorting the determined list comprising at least the at least one part of the at least one transaction, the sorting step being carried out as a function of a type of the at least one transaction, of the data and time of execution of the at least one transaction, of a service requester application associated with the at least one transaction, and/or of a service provider application associated with the at least one transaction.

13. The method of claim 11 or claim 12 further comprising a step of creating a service request based on the selected part of at least one transaction.

14. The method of claim 13 further comprising a step of sending the created service request to a service provider application adapted to process the created service request.

15. The method of claim 14 further comprising a step of selecting the service provider application adapted to process the created service request.

16. The method of claim 15 further comprising a step of receiving a service response from the selected service provider application in response to the step of sending the created service request.

17. The method of claim 16 further comprising a step of displaying the received service response.

18. A computer program product for a programmable apparatus, the computer program product comprising instructions for carrying out each step of the method according to any one of claims 1 to 17 when the program is loaded and executed by a programmable apparatus.

19. A computer-readable storage medium storing instructions of a computer program for implementing the method according to any one of claims 1 to 17.

20. A device for processing of a service request in a web runtime environment, the device comprising at least one microprocessor configured for carrying out the steps of:

- accessing at least one part of at least one transaction, the at least one accessed part of the at least one transaction being previously stored and comprising at least one of one part of a service request and one part of a corresponding service response generated by a service provider application,

- displaying a list comprising the at least one accessed part of the at least one transaction;

- enabling the selection of at least one part of at least one transaction, in the displayed list, by a user, the at least one selected part of at least one transaction comprising at least one part of a service request or at least one part of a service response generated by a service provider application in response to a service request distinct from the service request being processed; and

- obtaining a service response to the service request being processed, the obtained service response being obtained as a function of the at least one selected part of at least one transaction.

21. The device of claim 20 wherein the at least one microprocessor is further configured for carrying out a step of enabling the edition of the selected part of at least one transaction and a step of enabling validation of the edited selected part of at least one transaction.

22. The device of claim 21 wherein the at least one microprocessor is configured so that the step of enabling the edition of the selected part of at least one transaction comprises a step of modifying a value of a parameter of the selected part of at least one transaction, a step of adding a parameter to the selected part of at least one transaction and of setting a value to the added parameter, a step of removing a parameter from the selected part of at least one transaction, or a step of generating a service request comprising a value of at least one parameter of the selected part of at least one transaction.

23. The device of claim 22 wherein the value of the modified parameter or of the added parameter is determined as a function of a selected part of at least one transaction.

24. The device of any one of claims 20 to 23 wherein the selected part of at least one transaction is at least one part of a service response generated by a service provider application in response to a service request distinct from the service request being processed, the at least one microprocessor being further configured for carrying out a step of receiving the service request to process.

25. The device of claim 24 wherein the at least one microprocessor is further configured for carrying out a step of determining the list comprising at least the at least one part of the at least one transaction.

26. The device of claim 24 or claim 25 wherein the list comprising at least the at least one part of the at least one transaction is determined as a function of a type of the received service request.

27. The device of any one of claims 24 to 26 wherein the list comprising at least the at least one part of the at least one transaction is determined as a function of a type of data to be returned in response to the received service request.

28. The device of any one of claims 24 to 27 wherein the at least one microprocessor is further configured for carrying out a step of creating the obtained service response based on the selected part of at least one transaction.

29. The device of any one of claims 24 to 28 wherein the at least one microprocessor is further configured for carrying out a step of sending the obtained service response to a service requester application at the origin of the received service request.

30. The device of any one of claims 20 to 23 wherein the selected part of at least one transaction is a service request, the at least one microprocessor being further configured for carrying out a step of determining the list comprising at least the at least one part of the at least one transaction.

31. The device of claim 30 wherein the at least one microprocessor is further configured for carrying out a step of sorting the determined list comprising at least the at least one part of the at least one transaction, the sorting step being carried out as a function of a type of the at least one transaction, of the data and time of execution of the at least one transaction, of a service requester application associated with the at least one transaction, and/or of a service provider application associated with the at least one transaction.

32. The device of claim 30 or claim 31 wherein the at least one microprocessor is further configured for carrying out a step of creating a service request based on the selected part of at least one transaction.

33. The device of claim 32 wherein the at least one microprocessor is further configured for carrying out a step of sending the created service request to a service provider application adapted to process the created service request.

34. The device of claim 33 wherein the at least one microprocessor is further configured for carrying out a step of selecting the service provider application adapted to process the created service request.

35. The device of claim 34 wherein the at least one microprocessor is further configured for carrying out a step of receiving a service response from the selected service provider application in response to the step of sending the created service request.

36. The device of claim 35 wherein the at least one microprocessor is further configured for carrying out a step of displaying the received service response.

37. A method for a web runtime environment to process a service request substantially as hereinbefore described with reference to, and as shown in Figures 4 to 7.

38. A device for processing of a service request in a web runtime environment substantially as hereinbefore described with reference to, and as shown in Figure 8.

Intellectual
Property
Office

Application No:	GB1408930.4	**Examiner:**	Kalim Yasseen
Claims searched:	1-38	**Date of search:**	19 November 2014

Patents Act 1977: Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance
X	1-38	US2009/0013036 A1 (KIM) see whole document especially paragraphs 9, 10
X	1-38	US2009/0152341 A1 (KASPERKIEWICZ) see whole document especially paragraphs 40-42, 57, 59
X	1-38	US2009/0234876 A1 (SCHIGEL) see whole document especially paragraphs 26, 52, 56, 98, 116, 123
X	1-38	US2008/0305815 A1 (SMARTTOUCH) see whole document especially paragraphs 32, 86

Categories:

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC^{X} :

Worldwide search of patent documents classified in the following areas of the IPC

G06F

The following online and other databases have been used in the preparation of this search report

Online: EPODOC, TXTE, WPI

International Classification:

Subclass	Subgroup	Valid From
G06F	0017/30	01/01/2006
G06F	0009/54	01/01/2006