LivePoplet: Technology That Enables Mashup of Existing Applications

● Akihiko Matsuo ● Kenji Oki ● Akio Shimono

(Manuscript received January 29, 2009)

We have developed LivePoplet, a technology that allows the addition of functionality by mashup without modification of the existing Web applications. With the concepts of SaaS and Cloud computing now popular, demand has been increasing within corporations for combining applications freely for use without having to consider whether they are provided internally or externally. By using the LivePoplet technology, flexible cooperation can be easily achieved for direct calling of a Web service from a window of an existing in-company system according to the user's needs, for example, which allows the building of a business system of a new era while making the most use of the existing system.

1. Introduction

As Internet technologies, in particular recent technologies such as Web 2.0, SaaS and Cloud computing, are becoming more widespread, the state of applications used in corporations is changing, which is described in **Figure 1**.

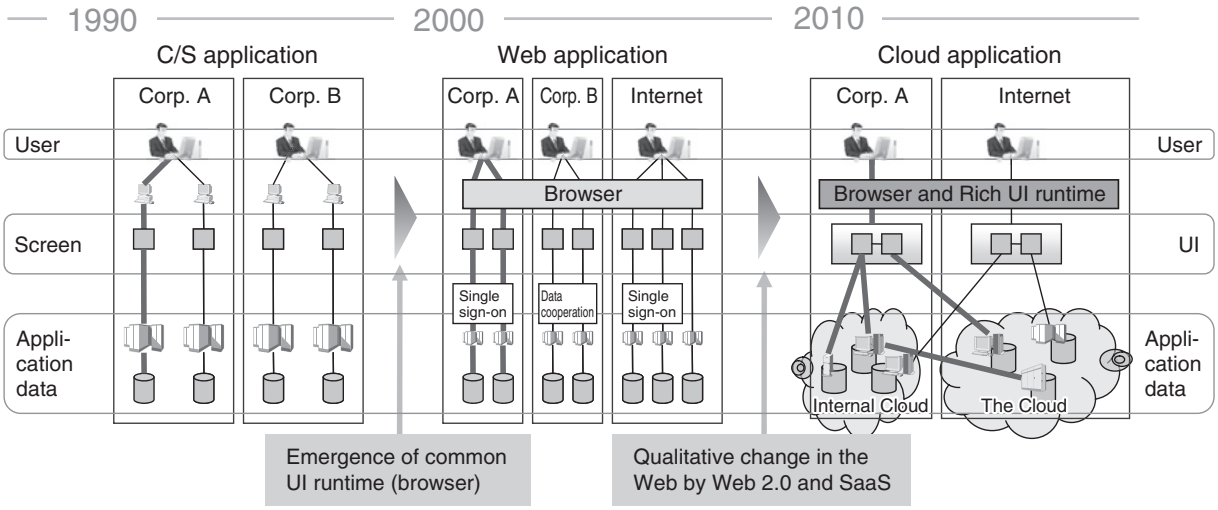
In the 1990s, before the Internet started to grow popular, client-server applications were the mainstream in corporations, and they were similar to the business applications in the era of mainframe systems in that the individual business applications were provided by different servers and the users also used individually dedicated client programs for using applications separately.

In the latter half of the 1990s, when the Internet became widely used, a form of Web application that used a Web browser as the client appeared, and the trouble of installing and updating dedicated client programs was eliminated. This made it common to construct corporate systems in companies in which introduction of an Intranet goes ahead, as Web applications. But the only change that had taken place was that browsers had come to be used in the same fashion as the client programs, and the reality was that the servers were operated in-house and users still accessed the individual sites to use applications separately.

As the concepts of SaaS and Cloud computing emerged, however, a style has become popular in which applications and services provided via the Internet cloud are used without the need for awareness of their locations, and users are expected to start using applications according to their needs without caring about who provides them.

2. State of business applications in the SaaS era and the present situations

As the era of SaaS and Cloud computing arrives, businesses other than those requiring processes specific to a given company become available as services without the need for in-house operation because SaaS applications are



C/S: Client-server

Figure 1
Changing business applications.

externally provided. This means that, while core businesses of the company will require continued operation of systems built in-house as they have so far, for peripheral businesses without special requirements, the use of SaaS applications externally provided is expected to increase, which is shown in **Figure 2**.

For efficient operation of businesses in such situations, combining the in-company systems with external SaaS applications to achieve free cooperation between them is desirable. However, existing in-company systems are seldom built to allow cooperation originally and the achievement of cooperation with external applications, in particular, requires activities such as implementation of communication functions, where these activities take many person-hours.

In addition, many systems usually exist in a company, including company-wide common systems and division-specific systems, and users have different needs as to which systems to cooperate with according to the divisions they belong to or their positions. Simply adding a static function to call other systems to the in-company systems is often unable to address diverse needs.

Among SaaS applications, on the other hand,

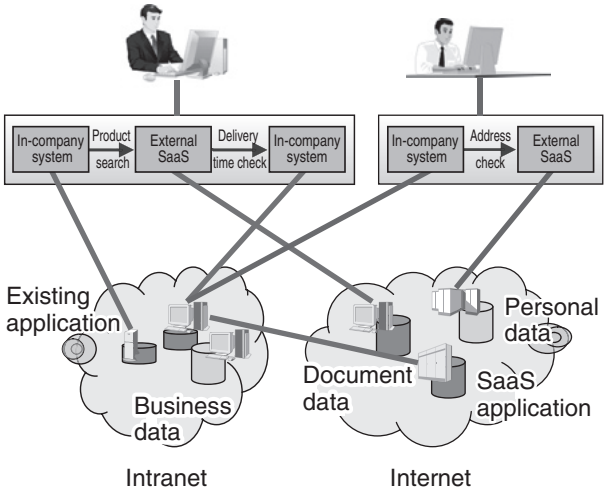


Figure 2
Style of business application use in SaaS era.

some are capable of customizing the windows and operations, adding process descriptions in special-purpose languages, and calling external Web services that facilitate changing of application operations by the user. However, calling in-company systems from external SaaS applications means accessing firewalled in-company systems from outside, which causes security concerns.

Under these circumstances, it is difficult at present to achieve flexible cooperation

between in-company systems and external SaaS applications.

3. Achievement of front-end cooperation by mashups

On the Internet, a technology for creating a new Web application by combining services from different sources appeared around 2005, which is called mashup technology. To take HousingMaps (*http://www.housingmaps.com*), which is famous as a pioneer of mashup applications, as an example to explain this technology, this application provides a function to display real estate vacancies on a map but the site itself does not have map data or real estate information and no cooperation process is taking place on the server. This site only has Web pages containing code to call a map display service provided by Google and a real estate property search service provided by craigslist, an online classified advertisement service. The application is described in JavaScript that runs on the browser, which eliminates the need for preparing any particular resource on the server, and simply creating Web pages with JavaScript code allows for the simple creation and publication of a Web application that uses services such as map services. This mechanism is shown in **Figure 3**. The feature of not needing to be the provider of content to easily implement an application that uses that content attracted the attention of developers on the Internet and many mashup applications have come to be created. Many services for mashups have become available as well.

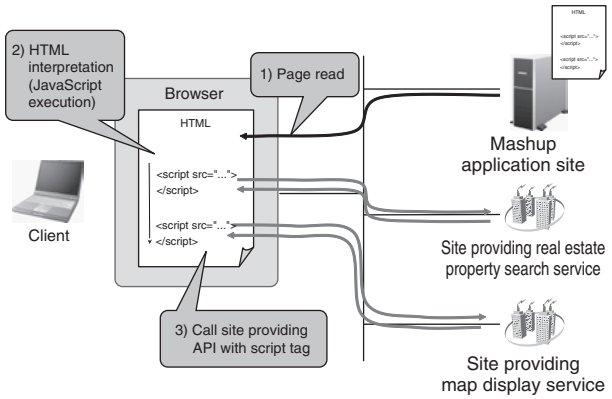


Figure 3
Mechanism of mashup application.

In corporations, there is a growing demand to be able to simply create applications with two or more services cooperating with each other by using such a mashup technique.

The traditional cooperation scheme that has been used for internal cooperation requires the cooperating systems to directly call each other's APIs, and cooperation is allowed only to system providers. In addition, many person-hours are necessary for checking operations to ensure the reliability of the systems already running, which makes it unsuitable to call a Web service provided as a beta version or to make frequent functional additions.

With cooperation via mashups, applications that use services do not require their creators to be the system providers. Users themselves can develop in a scripting language such as JavaScript and easily add functions according to their needs while checking application behavior, which makes mashups suitable for achieving application cooperation that is desired in the Cloud computing era.

While this mashup technique is effective on the Internet, where many services for mashups are available, use of this inside corporate Intranets makes it difficult to isolate functions to allow them to be called from mashup applications because not many in-company systems are provided with cooperation functionality.

There is also a technique called scraping, which extracts information from Web application windows. Although this is applicable in some cases, many business applications are built to call a function through many window transitions for activities such as authentication processes and function selection via menus, which require a number of exchanges with the server before reaching the intended process window. If

the window layouts of the target application are changed or the displayed content varies depending on the conditions, information may not be acquired or may not be able to be extracted unless the content currently displayed is recognized.

Accordingly, cooperation of existing in-company systems by means of the mashup technology currently available may require many person-hours for system modification and scraping processes, and the development of a simpler mashup technology is in demand.

4. LivePoplet: mashup technology with existing applications

To address this issue, we have conducted a study on technology to allow cooperation via mashups with respect to existing applications, developed a technology to allow mashups without modifying the existing applications and named this LivePoplet technology, which is outlined below.

4.1 Assumed form of cooperation

Cooperation via LivePoplet assumes calling a function of an application from a window of another application.

In-company systems often have many windows and provide complicated functions, such as windows for input assistance offering choices of items. For this reason, making in-company systems into mashup services requires making many functions into services. The mashup applications that use them also require the window functions already provided by the in-company systems to be built from scratch. Consequently, a substantial amount of person-hours will be necessary for offering in-company system functions by mashup applications.

This leads to a hypothesis that enabling users to call functions of other applications while using in-company systems, rather than making in-company systems into services to call from mashup applications, should effectively improve working efficiency while maximizing use of the functions of the existing systems. For example, while using an in-company system for the settlement of travel expenses, a Web service that computes the fare when given a travel itinerary can automatically be called. Similarly, information entered in a company-wide business meeting management system can be automatically reflected into a list of business meetings per company division. In this way, tasks that traditionally required users to open multiple windows for manual copying and pasting can be automated.

4.2 Method of implementing functional additions

Making these functional additions requires modifications to be made to the window display portion of an application. To eliminate the need to modify the main portion of the application, a technique for changing the operation of the application on the browser has been adopted. That is, rather than changing the operation of the application on the server, scripts for cooperation processes are added to the window's HTML content, thereby changing the operation on the browser.

Methods of changing the operation of the application on the browser include preparing scripts for additional functions on the browser in advance and providing a relay server between the browser and application server, where scripts can be added.

The first method requires preparations for the individual browsers, and extensions and plugins must be installed in advance as well, which makes it unsuitable for in-company systems of corporations with a need to reduce the burden of management on the client.

The method of using a relay server, on the other hand, requires no particular plugin on the browser and can be implemented without depending on the browser functions. It also

has the advantage that it is easy to conduct centralized management of information about the state of usage and access rights on the relay server. Accordingly, the relay server method has been used for the present study.

4.3 User interface for cooperation functions

To call from a window of an application a function of another application, the window display content and the responses to the user's operations must be partially modified. However, if the modification affects the display layouts of the application or hinders the input required, adjustment for individual applications or changing of the user interface becomes necessary and a user interface that does not affect the windows and functions of the cooperation source application whenever possible is desirable. For this reason, the developed technology has been built to display the functions of other applications in popup windows shown overlaid on the windows of the source application.

5. Implementation of LivePoplet

The LivePoplet technology is mainly composed of a portion for relay processing between the user's browser and application server, another portion for rewriting the content of HTML pages sent from the application server and a third portion for providing applications that run in the form of LivePoplet. **Figure 4** shows an overall configuration of servers for implementing the LivePoplet technology. A description of the operations of the individual portions follows.

5.1 Relay server

The relay server comes between the user's browser and the application-providing server to provide the function of relaying communications with the server. One method of relay processing is to use the proxy configuration of the browser. With this method, however, the relay server is not capable of re-encryption if communications using SSL are relayed, which makes it unsuitable for corporate systems because they often use SSL. For this reason, the CGI proxy method rather than the proxy server method is used for this

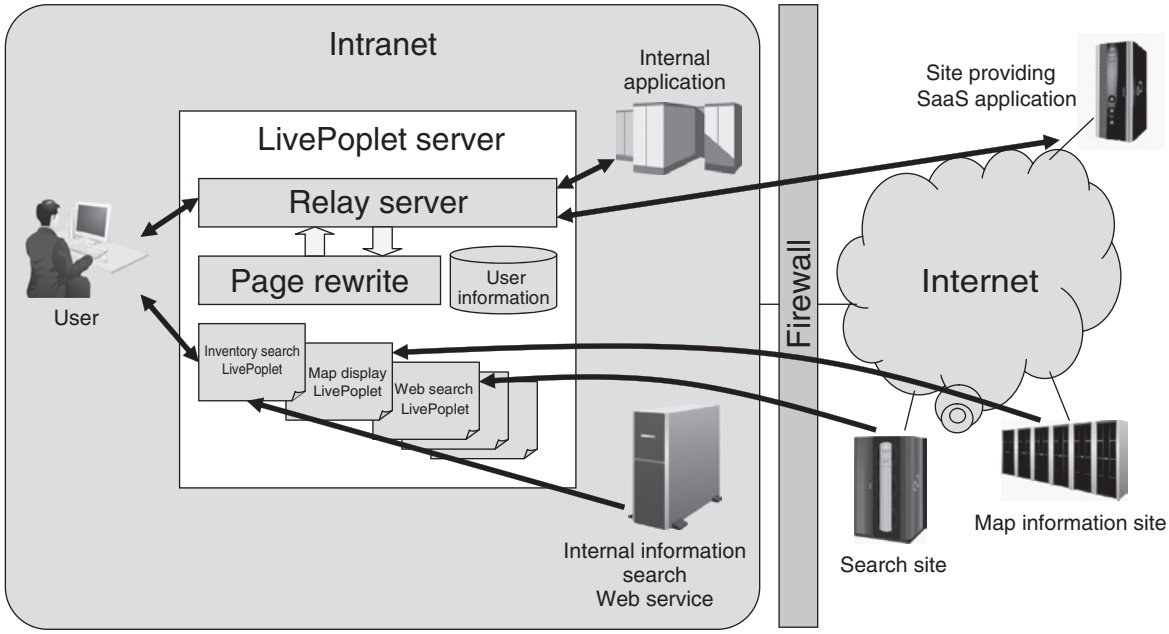


Figure 4
LivePoplet server configuration.

technology.

In the CGI proxy method, the URI of the original access destination is encoded in some fashion in the URI to specify the relay server as the access destination of the browser and have the relay server reconstruct the URI of the original access destination. For the present technology, a proprietary encoding method is used to satisfy the security requirements necessary for relaying for corporate systems. The following lists the typical encoding methods and an example of the encoding of the present method. In this example, relayserver.com is used as the relay server and *http://www.app.com/index.html* as the access destination.

1) First method: Passing the URI of the access destination as an argument
   http://relayserver.com/?url=http%3A%2F%2Fwww.app.com%2Findex.html
2) Second method: Adding the FQDN (full qualified domain name) of the access destination as it is
   http://www.app.com.relayserver.com/index.html
3) Present method: Adding the FQDN of the access destination after integrating into one component
   http://http-www-app-com.relayserver.com/index.html

One characteristic of the present method is that the FQDN of the original URI is integrated into one string joined by hyphens and added before the FQDN of the relay server to obtain an FQDN for encoding (any hyphen included in the original FQDN is replaced with two successive hyphens), after which the path name is appended as it is. Compared with typical methods, the present method has an advantage of achieving the following two features at the same time.

- Enabling the same origin policy restriction of browsers

In consideration of security, browsers adhere to the same origin policy, which prevents direct calling of services or such like belonging to other domains. If this policy is not followed, just showing a malicious page may lead to the risk of being freely accessed to another application to read out information. In the first method, all URIs are seen from the browser as the same domain (*cgiproxy.com*) and using this method to access an external site and an in-company system at the same time can be said to be dangerous. With the present method, a different domain remains a different domain even after encoding and the problem does not occur.

- Support for SSL communications

External SaaS applications often use SSL to prevent information leaks and SSL certificates must be prepared for FQDNs. With a method in which an FQDN changes according to the relay destination, preparing an SSL certificate in advance is not possible. Use of a wildcard certificate allows an arbitrary name for one layer, which cannot be applied to an arbitrary layer. For this reason, the second method is unable to support SSL, which is essential for business applications. With the present method, the name changes according to the access destination only for one layer and use of a wildcard certificate allows SSL communication to an arbitrary URI.

With a DNS configuration that directs all **.relayserver.com* accesses to *relayserver.com*, the relay server, in response to any access to such a URI, reconstructs the *http-www-app-com* component and joins it with the path name to reconstruct the original URI and makes the access to return to the user's browser. If the original URI is via SSL, SSL communication takes place between the application providing site and the relay server, which is decoded once on the relay server. Because SSL communication using the certificate of the relay server is enabled between the user's browser and the relay server, all communication paths for relaying can be encrypted.

5.2 Page rewrite

In response to any access from the browser, the relay server decodes the access destination URI into the URI of the original access destination. It then accesses this URI. If the

content of the reply is in HTML, a script injection process takes place and the URI of any link in the HTML is rewritten to an encoded URI to access to the relay server when the link is clicked in the browser.

In the script injection process, JavaScript code is added by inserting a script tag into a specific place on the page and, in theory, the content and operation of the page displayed can be freely modified by using this code. However, modification such as addition of a button in a window of the original application causes a conflict with the original display and functions of the application, which may require adjustment for the individual windows or leave the user in confusion.

To deal with this problem, the script to be injected first has been built to change only the operation activated by right-clicking of the mouse. The window itself is not changed, and a popup menu for calling LivePoplet is displayed upon right-clicking. **Figure 5** shows an example of a popup menu display.

The content of the popup menu displayed may be changed for individual users and individual URIs in the window displayed, which has achieved cooperation to meet the user's needs.

5.3 Application in the form of LivePoplet

A different application called from the popup menu is displayed as LivePoplet in a popup window independent of the window layout of the original application. **Figure 6** shows how LivePoplet is called. This popup is not a dialog or window but is shown overlaid on the original display of the application; it is in an area within the browser window that uses an iframe element of HTML and can be moved about the browser window.

Processes such as drawing and moving a window frame of LivePoplet are performed by the script injected and what is displayed in LivePoplet is an ordinary HTML document. Scripts such as JavaScript can be used as well, which makes it possible to display a window of another site or Web application as it is and display mashup applications using mashup services or scraping.

For the cooperation between the original application and LivePoplet, an interface is prepared that, when calling LivePoplet, passes a string selected in the browser as an argument of the URI for calling. This function allows hand-over of information from the original application to LivePoplet and a LivePoplet can be called indicating, for example, the result of a search for

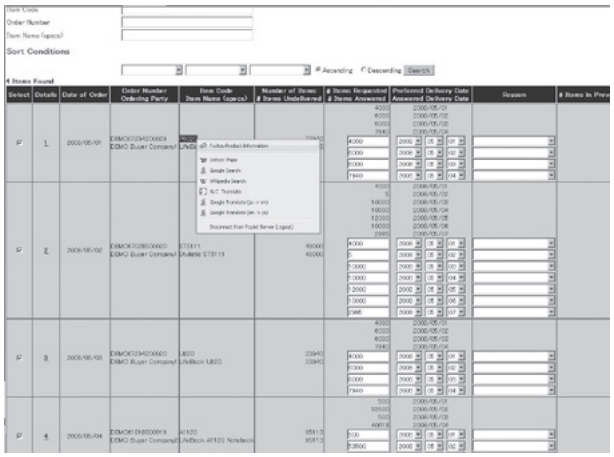


Figure 5
Example of popup menu display.

Figure 6
Example of LivePoplet display.

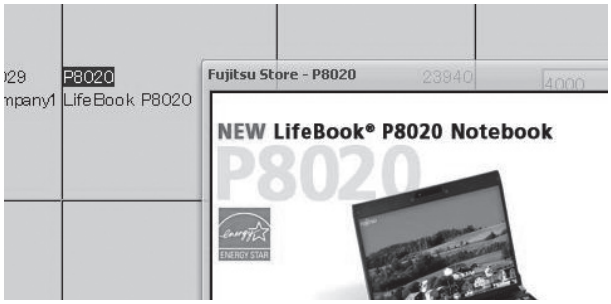


Figure 7
Example of search string selected for calling.

a selected item name as shown in **Figure 7**.

Automatic bi-directional cooperation calls for attention. Implementing automatic cooperation by a script results in the original application and LivePoplet being handled as the same HTML document in the browser. If the original application contains any malicious script, the information on LivePoplet may be read out. For this reason, iframe elements are generally used to have them handled as different HTML documents even within one window so that the original application and LivePoplet cannot directly refer to each other's information. If the original application and LivePoplet are both trusted, display without using iframes to allow automatic exchange of information is possible.

6. Implementation of new business applications by the LivePoplet technology

Use of the LivePoplet technology allows the addition of various cooperation functions without modifying existing Web applications. The following sections describe examples of cooperation and functional additions implemented by applying this technology.

6.1 Calling in-company systems from external SaaS applications

There is frequent demand for calling functions of in-company systems such as when referring to information in an in-company system while using external SaaS applications. Calling an in-company system within an Intranet from an external application requires access from outside through a firewall, which is difficult to implement. With the present technology, however, placing the LivePoplet server within the Intranet allows a user to call an in-company system from an external SaaS application.

6.2 Cooperating with beta version services

Many mashup services are provided as beta versions and are not guaranteed to reliably operate or are subject to frequent specification changes. Having such services cooperate with applications for which stable operation is required may compromise the quality or stability of the applications themselves. With the present technology, however, any problem with cooperation functions that may be generated does not affect the operation of the applications themselves and handling mashup services prone to frequent specification changes is easy.

6.3 Change of application UI functions

Use of the script injection function allows modification to the window display content of existing applications or replacement with more sophisticated displays. For example, functions can be added to displays containing simple HTML forms and table elements to allow use of information in the zip code field for automatic setting of the default value of the address field or to provide the tables with the column-by-column sort and width adjustment functions. Functions such as batch saving of the content of the input form on the display for reuse can be added as well.

7. Conclusion

As the concepts of SaaS and Cloud computing are gaining popularity, business applications are undergoing changes, and it is becoming more important to be able to freely combine various applications and services to construct new

applications that better serve the needs of users. The present technology has made it possible to call new services from existing in-company systems and to construct business systems of a new era while taking advantage of the existing systems.

The present technology is in the prototyping phase at present and will be experimentally provided in the near future. In addition, research on technologies to implement business applications of a new era is scheduled including cooperation between applications to allow bi-directional communications and single sign-on functions by LivePoplet server.

Akihiko Matsuo
Fujitsu Laboratories Ltd.
Mr. Matsuo received a B.S. degree in Physics from Tokyo University of Science, Tokyo, Japan in 1987. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1987 and has been engaged in research and development of software engineering. He is a member of the Information Processing Society of Japan (IPSJ).

Kenji Oki
Fujitsu Laboratories Ltd.
Mr. Oki received a B.S. degree in Information Science and Engineering and an M.S. degree in Computer Science from Tokyo Institute of Technology, Tokyo, Japan in 2004 and 2006, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 2006, where he has been engaged in research and development of software engineering, particularly Web technologies and software verification.

Akio Shimono
Fujitsu Laboratories Ltd.
Mr. Shimono received B.S. and M.S. degrees in Civil Engineering from the University of Tokyo, Japan in 1996 and 2000, respectively. He joined Fujitsu Ltd., Tokyo in 2000 and was engaged in production of B-to-C Web systems. He transferred to Fujitsu Laboratories Ltd., Kawasaki, Japan in 2007 to conduct research on Web-oriented architecture. He is a member of the Information Processing Society of Japan (IPSJ).