DoKumobility: Web services for the mobile worker

Jonathan Trevor and Gene Golovchinsky
FX Palo Alto Laboratory, Inc.
{trevor,gene}@fxpal.com

Abstract

Mobile users often require access to their documents while away from the office. While pre-loading documents in a repository can make those documents available remotely, people need to know in advance which documents they might need. Furthermore, it may be difficult to view, print, or share the document through a portable device such as cell phone. We implemented DoKumobility, a network of web services for mobile users for managing, printing, and sharing documents. In this paper, we describe the infrastructure and illustrate its use with several applications.

1. Introduction

Mobile users often need to access and use their documents while away from the office. Different technologies have addressed both the access and usage requirements of these users.

Web repositories such as BSCW[2] and DocuShare[3] expose web interfaces to file systems that users can access from anywhere using standard web browsers, provided they had uploaded the files to the web site prior to traveling.

Other web-based systems have attempted to provide services that focus on specific document services, rather than access. Internet fax services like InterFax[8] allow users to receive and send faxes using email or a web site.

There have been several limited combinations of services that access and services that act on documents (e.g., Satchel [3]). In general, however, each of these solutions remains an island. Different web repositories provide their own interfaces to their own file systems. Different online document services require different interfaces for uploading and executing. Consider the following scenario:

Fred is traveling on business, and Sally, his real estate agent, needs to get a document to him for his signature by the end of the day. Sally sends the document to him by confirmed digital fax (an enhanced fax service); the service sends an SMS message to Fred's cell phone informing him of a waiting document. Fred receives the notification when his plane lands, heads over to the airline lounge during his layover in Dallas, and uses his cell phone to route the incoming document to the club's fax machine. He reviews the document on his connecting flight. When he arrives at his hotel, he uses a networked printer at the hotel's business center to print out a high-quality copy of the document, signs it, and sends a confirmed digital fax back to Sally.

This simple example highlights several requirements of mobile access and application of information: a range of devices; a diversity of places a document can reside; and the variety of actions that need to happen on a document.

The emergence of web services and the growing provision of remote API's for using these services offers an opportunity to join different document providers and document services together.

DoKumobility is a service-oriented architecture that standardizes access to any type of document source provider and applies a pluggable set of web services to those files. The distributed and loosely-coupled nature of the system lends itself naturally to supporting mobile document-centered work. The end user experience is selecting a document from any source service and applying another service to it.

DoKumobility's system architecture (Figure 1)

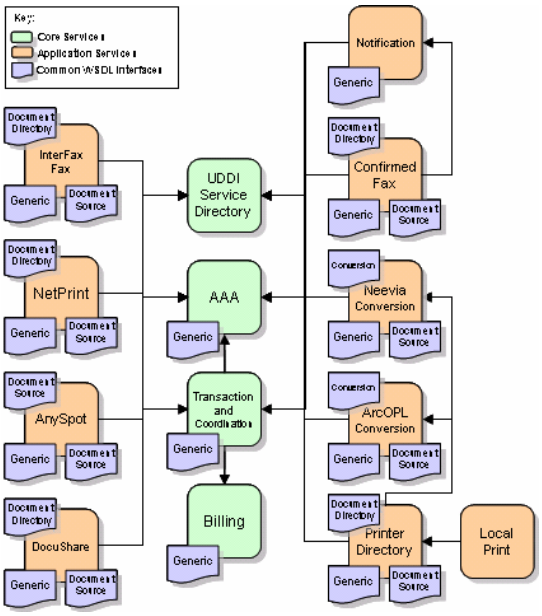

Figure 1. System architecture.

consists of several *core services* and a number of *application services*. Core services are required for the system to function; removing any one of them prevents the system from working. Application services perform useful tasks for users, whether selected explicitly by a user or through composition with other services. Finally, user interfaces – web, phone, and printer – allow the user to interact with the services wherever they go.

2. Core services

Core services are the glue that allows other (application) services to perform meaningful work on the user's behalf. They facilitate service discovery, user authentication, orchestration, and billing.

2.1. UDDI

Although not a web service *per se*, UDDI is a core component of the DoKumobility system. All services must be registered with the UDDI, and each service uses UDDI to find other services in the system prior to invoking methods on them. Different classes of services are identified through an interface type and version identifier, a string that corresponds to a particular WSDL (e.g., "`Conversion.1.0`" for conversion services). Specific instances of services are identified using a unique service ID, stored in the UDDI in a category bag along with a textual description, public key, access point, and other attributes appropriate for the service.

2.2. AAA

Users authenticate themselves to the Authentication, Authorization and Accounting (AAA) service. We allow multiple credentials for the same account, so people can use an authentication scheme specific to a class of device. For example, a user may use the phone's built-in subscriber ID as the login, and a pin as a password. This service also implements a secure credential cache for storing credentials for third-party services, used to enable transparent access and single sign on to legacy services with other security schemes.

When a service wishes to authenticate its invoker, it encrypts a secret with the invoker's public key (stored in the UDDI) and expects the invoker to decrypt the secret. This provides a measure of security for services such as the billing service.

2.3. Coordination Service

We provide a light-weight distributed transaction mechanism to coordinate composite operations. A transaction specifies services that perform specific operations to complete a given task. The Coordination Service creates transactions at another service's request and collects notifications from participating services regarding the progress of a transaction. Transactions can have sub-transactions: the service that creates a transaction does not need to know the implantation details of other services. Furthermore, sub-transactions isolate service failure and allow the calling service to perform alternative operations. For example, if an SMS notification sub-transaction fails, the notification service can create a new sub-transaction that uses e-mail; the overall transaction can then complete successfully. The Coordination Service interacts with the Billing service to debit the customer's account as operations are performed. If a transaction fails, the debit operations are reversed, and each service is notified to reverse the operation.

2.4. Billing

The Billing Service manages customers' accounts and supports real-time micro-payments for document services. It has two account types, prepaid and credit: prepaid customers can execute services as long as the account balance stays above zero; credit customers can execute services up to the credit limit.

Service billing rates depend on selected Billing Service Plans. For example, printing services can be charged per page, while translation services can be per word and/or per document setup. A discount can be applied when a transaction involves printing and translation services at the same time.

For security reasons, the billing service can be invoked only by the coordination service.

2.5. Interoperable WSDL-based interfaces

As we developed services, it became clear that many had common methods and properties, including the service ID, public keys, a description, a service type, etc.

To avoid unnecessary code duplication, and to aid service discovery, interoperability and composition (see section 5.4) we developed a set of common WSDL "interface" descriptions that services providing that functionality should implement (see Figure 1). The four interfaces in DoKumobility are necessarily document biased, although additional ones could be defined:

- **Generic Service.** Each web service implements the Generic Service interface, which defines methods and structures for accessing service descriptions and for registering the service with UDDI.
- **Document Source.** The Document Source interface specifies methods for obtaining document metadata and for fetching and storing document contents. Contents are retrieved and stored using HTTP GET and POST requests.
- **Document Directory**. The Document Directory interface defines the API for a hierarchical

directory structure that is used in conjunction with a document source. Note that a document source doesn't need a directory structure if all it does is store and retrieve documents that are not accessed by a user. (The conversion service is one example.)

- **Conversion Service**. The Conversion Service WSDL defines an API for converting documents from one MIME type to another. Each service that implements this interface also registers with the UDDI the pairs of MIME types that it can convert. It is the caller's responsibility to find the right service instance for the required conversion.

3. Application Services

There are three types of application service: *self contained*, with no dependences on anything outside of the service; *local wrappers*, which coordinate access and invocation of a local application; and *external wrappers*, which manage the marshalling and execution of calls from DoKumobilty to an existing 3rd party service API.

3.1. Self-Contained Application Services

The fax machine is ubiquitous in the business world. For all its popularity, however, it suffers from several important problems. Faxing is tied to devices, not to people. If a person is traveling, determining the right fax number may be quite difficult. Fax machines lack adequate feedback regarding the success of the transmission. Faxing is not secure. One person may inadvertently (or intentionally) pick up someone else's fax, and the intended recipient will not know that a document had arrived. Because the sender lacks adequate feedback on the outcome of the process, it is common for recipients to deny having received the document.

Despite these shortcomings, sending documents to people is invaluable in many business processes. The **Digital Confirmed Fax Service** provides functionality that is analogous to fax transmission, but sends documents to a particular person rather than to a device, notifies the sender that a document has been sent and when it has been read, and requires the recipient to authenticate himself prior to receiving the document. The approach is similar to sending e-mail attachments, but, in contrast to e-mail, notifies the sender when the recipient has accessed the document and allows them to apply any service in the system from any device. Finally, unlike e-mail, the Confirmed Digital Fax service guarantees notification.

The fax-like nature of the transaction is reinforced by the integration of this service with Fuji Xerox's f450 printer/copier. A mobile worker can scan a paper document, and have the Confirmed Digital Fax service deliver an electronic copy of it to the recipient; the recipient can view it on the screen, print it to paper, or route it to some other service.

The **Printer Directory Service** addresses one of the key considerations that motivated the design of this system. We wanted to create a system through which a mobile user could print any document on any printer in the world. Printing a document involves the orchestration of three services: the Printer Directory Service, a Document Conversion service, and a Local Print service. Printing poses a particular orchestration challenge because Local Print services typically run behind firewalls. We get around this limitation by having the Local Print service poll the Printer Directory service to request print jobs.

Finally, the **notification** service routes messages from services in the system (such as confirmed fax delivery messages) to users. Messages are delivered as SMS text messages or as email depending on the user's profile stored in the AAA service.

3.2. Local Application Service Wrappers

The **ArcOPL** and **Neevia** conversion services transform documents to printable formats. Both services are wrappers for existing conversion software. The printer directory service selects the appropriate conversion service based on the MIME type of the document to be converted and on the MIME type of the Print Description Language (PDL) that should be produced. These types are stored in each service's UDDI metadata. In the event more than one service can perform the conversion the first matching web service is used.

3.3. External Service Wrappers

External Service Wrappers integrate pre-existing or legacy services into the architecture. In general, these services already have their own sets of accounts, users and credentials and DoKumobility supports two approaches for integrating them.

First, each service is responsible for managing its own RSA public-private key pair (the public portion is available in the UDDI). User's can allow the AAA service to store their credentials for the external service, encrypting them with the services public key. When the user attempts to access the legacy service through the service wrapper, the wrapper fetches the cached credential from the AAA service and decrypts it using its private key. In this manner a user can perform *single-sign on* and credentials can be updated securely through a single user interface.

DocuShare[4] is an online web repository that allows users to access upload and share files from any web browser. The DocuShare wrapper makes use of the single-sign on mechanism for exposing the contents of a DocuShare user's files in the DoKumobility system

AnySpot is a research prototype that uses a combination of web services to transparently enable access to any file or folder within an organization on existing file systems (such as a user's desktop files, or a network's Linux server).

The second approach for wrapper services is to perform their own mapping of DoKumobility accounts to a single account known only to the wrapper itself for the external service.

NetPrint[9] allows users to upload documents to an online repository, and then to print them to 7-Eleven stores in Japan by typing in the six digit code for the document and depositing some money. The wrapper accepts documents from any other service in the system, submits it to a single NetPrint account, and sends a notification with the document code to the user's cell phone or e-mail (using the notification service).

InterFax[8] is a web service through which electronic documents can be sent via analog fax to any fax machine in the world. Our wrapper service submits documents to InterFax, and debits the user's account for the cost of faxing the document.

4. User interfaces

Mobile workers may use their laptops, PDAs, or cell phones to access electronic information, as the situation permits. One of the design goals for DoKumobility was to provide access to documents and services through a variety of devices to support this opportunistic work practice. Here, we describe three interfaces that make up the DoKumobility suite. Through these interfaces, users are able to retrieve their documents, and print, fax, share, or view them.

4.1. Desktop Web Browser Interface

When a web browser is available, a mobile worker can use it to access documents and services. The web interface (Figure 2) lists the document sources on the left, shows the contents of the selected source in the middle frame, and the details of a selected document in the bottom pane. When a document is selected, services that are able to act on that document display buttons (e.g., "Print", "Fax", etc.) that trigger the corresponding actions.

Pressing a button shows a dialog box (e.g., Figure 3) through which the user interacts with the service.

4.2. Printer

We extended the embedded interface of Fuji Xerox's DocuCentre f450 printer/copier to connect to our web services, and to perform printing and scanning functions on documents. By selecting appropriate job templates from the printer console, users can: scan a paper document and route it to themselves or to another person; scan a document and confirm or analog fax it; print any document, including incoming confirmed digital faxes.

The integration of the device's scanning and printing capabilities into the DoKumobility framework allows users to treat paper and electronic documents interchangeably, switching from one to the other as appropriate to the circumstances. This not only streamlines interaction with the system, but also allows the DoKumobility user to work with others who may not have access to the DoKumobility platform through normal faxing.

Figure 2. DoKumobility on the web

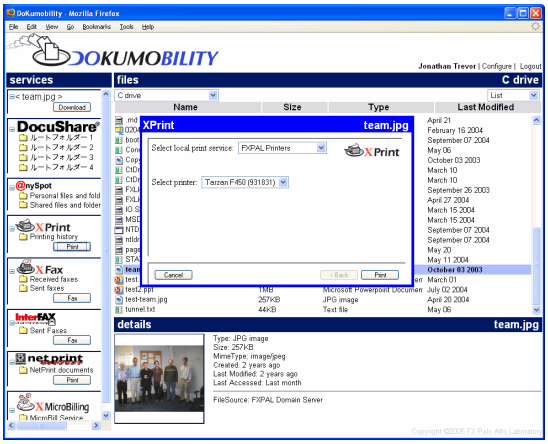

Figure 3. Printing a file

4.3. Phone

When traveling, a cell-phone based interface to DoKumobility is a viable and essential substitute for the more capable infrastructure available in the office and desktop-oriented UIs. The phone interface provides access to the same documents and services through a web-based interface which is more suitable for such a small form-factor.

The phone interface enables users to select a document from any document source and then apply the same services available in the web interface to the selected document (Figure 4). Thus, using the cell

Figure 4. Cell phone user interface

phone UI, a user can select a document and fax it, print it, or share it with another person.

5. Lessons Learned

As we developed and completed the services in Figure 1, we made several architectural decisions related to web services integration. In this section, we describe some of the important findings, and also talk about some outstanding issues and future work.

5.1. Optimizing document processing

The nature of DoKumobility promoted "documents" as one of the main types of data to flow between services – typically in the range of a few kilobytes to several megabytes. Because some services process document metadata rather than content, we created a mechanism that enabled web services in our network to the document content from the documents metadata, only fetching the actual content if necessary.

Document metadata include a set of pre-defined common document fields (name, MIME type, size and so on), a name-value bag that could be used by any service to store additional information about that document in addition to the common fields (such as the number of pages in a paginated document), and the identity of the document source service that maintains the document content.

Web services receiving such a document description can fetch the actual contents locally (to disk) by looking up the service's access point through UDDI. In our current design this transfer is performed using a simple HTTP request, although a DIME based method would also be appropriate.

5.2. Decentralized orchestration vs. BPEL

We chose a decentralized orchestration approach in which services call other services as needed to satisfy users' requests. The first service to start a process defines the participating services; this transaction is tracked by the Coordination Service.

This approach has several advantages. First, it is quick to develop and prototype. Secondly, the amount of network overhead in decentralized orchestration is inherently less that a centralized model (25% fewer calls in our print/conversion/local print service orchestration versus a BPEL engine). Unfortunately, this approach requires significant duplication of code (as each application must implement similar functions to invoke appropriate services). More significantly, changes to the orchestration require service code to be recompiled.

A centralized orchestration alternative, such as a WSBPEL[1] engine that executes process specifications expressed in XML, may solve the extensibility problem. Further, centralizing the execution simplifies error handling, and exposes a consistent API through which applications can monitor job progress. This solution scales with the addition of multiple BPEL engines.

5.3. Firewalls and security

Firewalls complicate web services architectures because they can block incoming web services calls. Yet they are an inescapable (and vital) part of most networks, and must be accommodated in any web-based platform design. We solved the problem in our current services by requiring the Local Print Service to poll the Printer Directory Service to obtain pending print jobs. While not ideal, this solution typically requires no modifications to existing firewall policies, and it allowed us to demonstrate the system in networks over which we had no administrative control.

Unfortunately, polling across the network is not ideal because it either increases traffic to critical resources if the polling frequency is high, or degrades throughput if the frequency is low. To solve this problem we would recommend applying reverse firewall tunnels to transparently invoke web services when they cannot be reached directly.

In such a scheme, the tunnel is constructed using two components, the invoker that runs inside the firewall, and a proxy that runs outside. The invoker opens a few connections to the proxy (outside the firewall) and blocks, waiting for a response. When the external proxy receives an HTTP request (SOAP or otherwise) for the service, it returns the request on a corresponding open connection to the invoker, which then executes the original HTTP request against the internal web server. The invoker returns the response data through a new connection to the external proxy, which, in turn, passes the data back to the original caller.

5.4. Composition

A goal for any "pluggable" component approach is to add new services to the collection with minimal

changes to the system. In our architecture we address this concern through WSDL interfaces. If a service supports a certain interface (e.g., a document source), application interfaces know how to render that interface and other services know how to access the documents that service provides.

Our experience showed that this approach works when all services implement common interfaces and associated semantics. There are two inherent consequences of this approach, however. First, existing services (e.g., AnySpot) that provide "documents" with proprietary or legacy APIs need to be wrapped by services that implement our interfaces and data types and translate calls to the original service. Also, service-specific capabilities not represented in these standard interfaces become "lost." If the common WSDL interfaces do not represent the capability, it cannot be represented in the user interface.

This disconnect means that services like AnySpot still need to expose their own specific interfaces (both graphical and programmatic) for providing additional configuration or functionality. It also means that introducing new functionality into the platform requires new interfaces and standard data-types, which in turn leads to more software development, particularly for application UIs.

These shortcomings seem to indicate the need for composibility and orchestration not only at the process and application level – but *also* at the user interface itself.

6. Related work

A consequence of DoKumobility's document-centric service oriented architecture is that many pieces of functionality overlap, and indeed leverage, existing work. On-line services like PrintMe[7] or eFax[10] provide alternatives for the printing and faxing capabilities in DoKumobility, and could be incorporated into the architecture to further increase the availability of printers to the system. However, these types of document-oriented internet services are limited to the documents either uploaded before-hand, or available at the user's device for uploading. In contrast, DoKumobility facilitates the coupling of document sources with these types of services.

Conversely, systems that act as document sources (e.g., BSCW[2], GotoMyPC[5]) provide network access to documents, but have only a very fixed number of supplementary services (if any) or can be accessed only through desktop devices, limiting their usefulness to mobile workers. Wrapping such systems with DoKumobility's common WSDL interfaces can give access to these different sources in a unified manner, for any device.

The actual document source and directory interfaces provide similar functionality to the WebDAV[6] protocol, but provide it at a web service level (enabling orchestration) rather than as a lower level network protocol.

Finally, our system differs from Satchel[3] in that it is an open architecture with an unlimited number of services, it has access to all files on users' networks without the need to cache, and it includes user interfaces for a variety of devices, including printers.

7. Conclusions

DoKumobility represents an attempt to construct a document-based service oriented architecture using current web service standards and technologies from the ground up. Our experiences reinforce one of the key touted advantages of web services – the ability to create new and complex systems from reusable web components. It took less than a week to introduce an existing SOAP-based service into the system. It took nearly 3 months (and still continuing) to integrate a standalone application.

Much remains to be done, however.

User interface construction across different platforms and devices using such re-usable components can also benefit from a web service based method of construction. Moving most of the logic out of applications and into services should make it easier to add new devices. We are in the process of replacing our initial architecture with a BPEL-based approach. One side effect is a more homogeneous representation of computation, which should make the applications more light-weight. Finally, we are developing our own tunneling gateway to enable remote services such as AnySpot or DocuShare to be incorporated more easily.

8. References

[1] OASIS Web Services Business Process Execution Language Technical Committee, "WSBPEL Version 2.0 Working Draft", http://www.oasis-open.org/

[2] Bentley, R., Horstmann, T., Sikkel, K., and Trevor, J. (1995): "Supporting collaborative information sharing with the World-Wide Web: The BSCW Shared Workspace system", 4th International WWW conference, Boston, MA, USA. Dec. 11-14, 1995.

[3] Lamming, M., Eldridge, M., Flynn, M., Jones, C., Pendlebury, D. (2000): "Satchel: providing access to any document, any time, anywhere". ACM TOCHI 7(3)

[4] http://www.xerox.com/ds

[5] https://www.gotomypc.com/

[6] http:// www.webdav.org/

[7] http://www.printme.com/

[8] http://www.interfax.net/

[9] https://www.printing.ne.jp/

[10] http://www.efax.com/