FormSys: Form-processing Web Services

Ingo M. Weber
ingo.weber@cse.unsw.edu.au

Hye-young Paik
hpaik@cse.unsw.edu.au

Boualem Benatallah
boualem@cse.unsw.edu.au

Zifei Gong
zgon235@cse.unsw.edu.au

Liangliang Zheng
lzhe544@cse.unsw.edu.au

Corren Vorwerk*
correnv@cse.unsw.edu.au

School of Computer Science and Engineering, K17
University of New South Wales
Sydney, NSW, Australia, 2052

ABSTRACT

In this paper we present FormSys, a Web-based system that service-enables form documents. It offers two main services: filling in forms based on Web services' incoming SOAP messages, and invoking Web services based on filled-in forms. This can be applied to benefit individuals to reduce the number of often repetitive form fields they have to complete manually in many scenarios. It can also help organisations to remove the need for manual data entry by automatically triggering business process implementations based on incoming case data from filled-in forms. While the concept applies to forms of any type of document, our implementation uses Adobe AcroForms due to its universal applicability, availability of a usable API, and end-user appeal. In the demo, we will show the two core functions, namely `soap2pdf` and `pdf2soap`, along with use case applications of the services developed from real world scenarios. Essentially, this work demonstrates how PDFs can be used as a channel for interacting with Web services.

Categories and Subject Descriptors

H.4.0 [**Information Systems**]: Information Systems Applications; D.2.13 [**Software**]: Reusable Software

General Terms

Algorithms, Design

Keywords

Documents, Processes, Service-enabled Forms, Web Services

1. INTRODUCTION

Modern business organisations employ a diverse range of business process implementation solutions, including Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), workflows, and the like [8]. However, paper-based forms are still prevalent in many of the interactions between the organisations and their customers or business partners. For example, one needs to fill a series of forms with personal details and a varying degree of additional information to open a bank account or request a driver's license. Receiving these forms at the organisation's end may trigger a number of business process instances. More often than not, extracting the data from forms into software (called *media break*, e.g. by [4]) is a highly manual task.

Automating such manual activities involves a team of experienced programmers implementing electronic versions of the forms, e.g., HTML forms embedded in a Web-based system. This solution is ad-hoc, costly and the time-to-production is long. In fact, to make a form and its associated business process available instantly, it is still easier for administrative office workers to create a form using a word processor, save it in a widely-accepted format (e.g., PDF) and distribute it e.g., via email. Of course the processing of such forms still remains manual.

We believe it is desirable to have a solution which utilises tools that the office workers are already familiar with, but at the same time enhances the ability to create forms and easily automate the processing procedures involved. As a step towards the envisioned solution, in this paper, we describe and demonstrate FormSys, a tool that provides Web services for effective form processing. FormSys is designed to be used by people *without deep technical knowledge* and deal with any *existing form as-is*. In particular, we use a sub-standard to PDF called *AcroForms* [3]. A discussion on design choices can be found in Section 3.1.

In FormSys, office workers can easily create "form-processing Web services" from PDFs, meaning the data the forms hold can be received and sent via SOAP messages. This makes it possible, for instance, to fill the common parts of multiple forms simultaneously, or to electronically send a completed form's content to a workflow system and trigger an event. More specifically, when a form is uploaded by the user, FormSys can dynamically generate two services:

- `soap2pdf` accepts data from an application, fills an AcroForm with it, and sends it back via email or provides a URL under which the filled form is available.
- `pdf2soap` extracts the data from a filled AcroForm, assembles and sends a SOAP message to an application.

The demonstration of FormSys comprises `soap2pdf`, `pdf2soap`, and use case applications developed from real scenarios.

2. RELATED WORK

To the best of our knowledge, there is no other work that offers the functionality discussed in this paper with form documents. There are a number of commercial tools, such as Adobe LiveCycle Designer [1], Crystal Report[1], and For-

*The author is enrolled with Univ. of Applied Sciences, Karlsruhe, Germany. This work was done when visiting UNSW.

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

[1]`http://www.crystalreports.com`

mMax [2], that are aimed at designing form templates, dynamically generating forms or assisting form filling activity.

We also found numerous software products which utilise OCR or barcode reading[2] to capture data from a document. These systems mostly focus on building a database of scanned forms for indexing or search purposes. There are other works that focus on automatically extracting a structure out of flat PDF documents. For example, in [5] where an algorithm is proposed to automatically detect different parts of a PDF document (e.g., Header/Footer) and produces an XML. Although the purposes of the conversion may vary, generally speaking, these automatic structure extraction techniques can be applied in our system, for example, to eliminate the field/group name mapping step which is done manually at the moment.

We should note that our system's focus is not on designing, creating or automatically scanning forms, but on providing a small, effective and self-contained service which can be used to easily consolidate manual, form-based user interactions in a business process into some degree of automation by providing form data processing as services.

3. SYSTEM OVERVIEW

After discussing design decisions, we will present the architecture and implementation of the tool.

3.1 Design Choices

While the concept of passing information from forms to application program interfaces (APIs) and vice versa is independent of specific API or document formats, we had to decide on specific technologies for the implementation. We here discuss these choices.

Form document formats. The de-facto Web standard for exchanging read-only fixed-layout documents today is Adobe PDF[3], which we decided to use. There is a substandard to PDF called *AcroForm* which features editable fields [3]. These fields can be of various types (text, checkbox, etc), and they reside on a visual layer above the regular document. The fields are named, and their size and position are determined by the form's designer. As alternatives, we considered GoogleDocs[4] and other common office software packages such as OpenOffice[5] or Microsoft Office[6]. We found that none of their APIs are as rich and as portable as what PDF offers.

Data exchange. Another design decision was to build on a structured communication standard. We here decided to use Web service technology. As alternatives we also considered REST and proprietary Remote Procedure Calls (RPC). However, REST by itself is not well structured, and thus less suitable for our scenarios. RPC technology imposes requirements on using the same programming paradigm (e.g., Java or .NET) in the communication partner as used for implementing the service provider. Therefore we decided on Web services, SOAP over HTTP, as well as regular email with MIME attachments.

[2]`http://www.scantopdf.co.uk/`

[3]A Google search for "filetype:pdf" finds 562,000,000 documents, vs. around 84,057,000 "doc" and "docx" (2/2/2010).

[4]`http://docs.google.com` (15/10/09)

[5]`http://www.openoffice.org` (15/10/09)

[6]`http://office.microsoft.com` (15/10/09)

3.2 Architecture and Tool Overview

The architecture of our system, cf. Fig. 1, has the following components: a front-end for forms administrators to upload and administer form templates; a format converter container, where converters from arbitrary formats to PDF can be included as plug-ins; a central `pdf2soap` component handling uploaded filled forms; an OCR pre-processor for using `pdf2soap` functionality on filled forms no in AcroForm format; a set of self-contained `soap2pdf` services for filling forms; a database for form templates and field mappings; and a core component controlling the whole system including `pdf2soap` functionality and `soap2pdf` service creation. These components and their usage are described below.

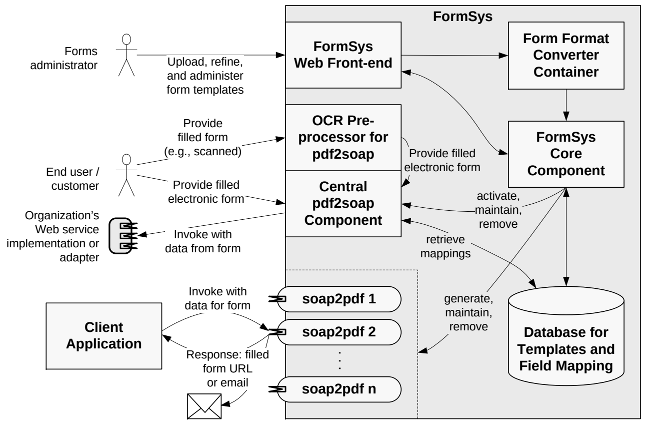
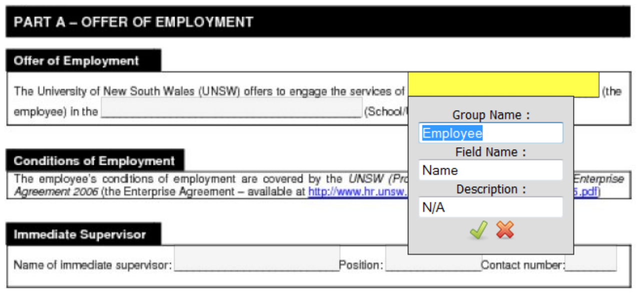

FormSys
Forms administrator
Upload, refine, and administer form templates
FormSys Web Front-end
Form Format Converter Container
OCR Pre-processor for pdf2soap
Provide filled form (e.g., scanned)
Provide filled electronic form
FormSys Core Component
End user / customer
Provide filled electronic form
Central pdf2soap Component
activate, maintain, remove
Organization's Web service implementation or adapter
Invoke with data from form
retrieve mappings
Invoke with data for form
soap2pdf 1
soap2pdf 2
soap2pdf n
generate, maintain, remove
Client Application
Response: filled form URL or email
Database for Templates and Field Mapping

Figure 1: FormSys Overview

Through the main Web interface of FormSys, forms administrators can control the handling of form templates in the core component: they can upload forms, specify the service name and namespace, edit the field naming, group the fields, and activate, maintain, or discontinue `soap2pdf` and `pdf2soap` functionality. If the form is initially not available in the required format (AcroForm), it needs to be converted to PDF and the fields need to be specified. For the prior step, available converters to PDF can be included in the system through the converter container. For partly automating the latter step, the discussed work [5] may be employed. Depending on a provided form template, it may be necessary to add fields, change the position and type, and change the names of fields or groups in the form – this ranges from specifying all fields for new forms to only renaming to make field names more descriptive. The forms administrator can edit fields and grouping, cf. Figure 2, using an intuitive user interface: an image of the uploaded form itself is presented, and existing fields can be selected by clicking on the respective areas of the image. The form template and field naming are stored in the database; they influence the data structures of the services created from the form.

An active `soap2pdf` service is a full-fledged Web service offering a WSDL interface with a single operation: `fillForm`. The input to this operation is the following: the input data for the form document; an expiry date for specifying how long a filled form should be kept available; an email address to which the filled form is mailed back; and a boolean switch for determining whether the filled form will be editable (i.e., remain as AcroForm) or rendered to a flat PDF. If no email address is given, or an email address and an expiry date


PART A – OFFER OF EMPLOYMENT

Offer of Employment

The University of New South Wales (UNSW) offers to engage the services of (the employee) in the (School/

Group Name : Employee

Field Name : Name

Description : N/A

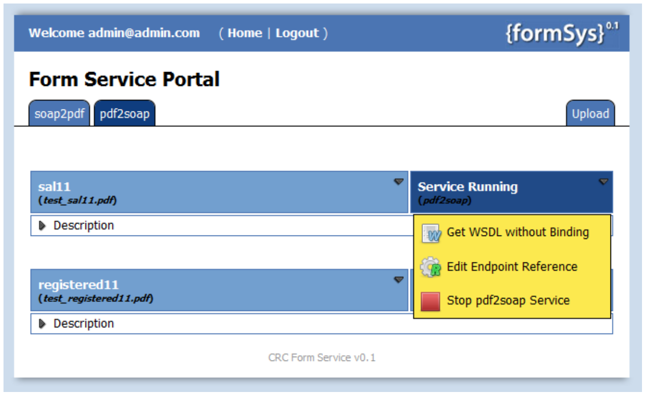
Conditions of Employment

The employee's conditions of employment are covered by the *UNSW (Pro* *Enterprise Agreement 2006* (the Enterprise Agreement – available at http://www.hr.unsw. 5.pdf)

Immediate Supervisor

Name of immediate supervisor: Position: Contact number:

Figure 2: Field name and group editing


Welcome admin@admin.com (Home | Logout) {formSys}0.1

Form Service Portal

soap2pdf pdf2soap Upload

sal11 (test_sal11.pdf)

Description

registered11 (test_registered11.pdf)

Description

Service Running (pdf2soap)

Get WSDL without Binding

Edit Endpoint Reference

Stop pdf2soap Service

CRC Form Service v0.1

Figure 3: Options for a running pdf2soapservice

are specified, the filled form will be made available under a URL, which is sent back in the synchronous SOAP response to an invocation. This response also contains a status report and a list of faults, if any. soap2pdf services can be flexibly used in (Web) applications and orchestrations, as discussed in the use case section.

pdf2soap works as follows: when a filled form is uploaded by an end user through the pdf2soap front-end component, the data is extracted and packaged in a SOAP message according to the mapping in the database, and sent to the associated consuming service. For forms administrators the set-up is less straight-forward: challenges are how to find the service, and how to map the data from the form to the schema of the message to be sent. We circumvent the difficulties by enforcing the following procedure. For an uploaded form template, the forms administrator can generate a pdf2soap WSDL interface specification without endpoint information. This interface needs to be implemented by the consuming organization. Once that is done and the service is deployed, the forms administrator provides FormSys with the endpoint under which the service is reachable. With this information a pdf2soap service can be started. The data extracted from filled forms is then forwarded to the given endpoint. Fig. 3 shows the forms administrator's interface in the pdf2soap view, where the part for controlling the pdf2soap service is a drop-down menu shown for "sal11.pdf". The database is hereby the only point where the pdf2soap runtime depends on the form design time. Thus, the central pdf2soap component can be physically separated from the core component, given the database is accessible.

Due to the prescribed procedure, the mapping from the form fields to the schema is known to our system. The procedure applies naturally to situations where the organization has no service implementation available at this point, as it frees them from designing the interface manually. However, if a Web service is already implemented, it is unlikely that the generated WSDL will match it. In this situation, an adapter has to be developed, e.g., based on [6] or on commercial products such as Oracle Fusion Middleware (OFM)[7] or SAP NetWeaver Process Integration (SAP NetWeaver PI)[8].

If a filled form is not an AcroForm, pdf2soap has a preprocessing component for Optical Character Recognition (OCR). Thus, the data from fields can be extracted even out of image formats, such that paper copies can be scanned and processed by our system. We plan to use existing OCR technology[2] to extract the data from the fields, however, this is not implemented as yet.

3.3 Implementation

Although the designed architecture is generic, our current implementation is specific to supporting AcroForms. However, we believe the system is mature enough to demonstrate the concept we propose. The system is entirely written in Java and makes use of the following libraries:

- Apache CXF[9] for all aspects related to Web service.
- ImageMagick[10] for creating images from PDFs.
- iText[11] for accessing and manipulating AcroForms.
- jQuery[12], for interactive graphical Web user interfaces.

The soap2pdf code generation starts from the mapping of user-given field names to each of the AcroForm fields – if unchanged the mapping is simply the identity function. The map is used in Java code generation from templates. The generated code has a class for each group containing the respective fields and their mapping to AcroForm fields. These classes are combined in an input bean with the according other parameters. A Web service implementation is then generated using CXF, which creates the filled AcroForm when invoked with the input bean, and accordingly handles the response and emailing. This code is compiled, packaged, and deployed using standard Java APIs. For pdf2soap the field mapping is handled centrally by the upload site.

4. USE CASES

In order to demonstrate the applicability and value of FormSys, we identified and implemented two use cases, all taken from real-world. Use case 1 is implemented as a Web application, and makes use of both pdf2soap and soap2pdf. Use case 2 is implemented as a Web service orchestration in BPEL[7], and uses only soap2pdf. All PDFs used in the use cases are publicly available on the respective Web sites.

4.1 Use case 1: Suncorp investment forms

Personalised Form Download. Often, banks require a paper-based form to be filled in by their customers to use additional banking services. As an example, the investment fund management forms of Suncorp, an Australian bank, can be found at `http://www.suncorp.com.au/suncorp/personal/Investing/forms.aspx`. From this page, Suncorp customers download blank PDF forms (not AcroForms), print, fill in,

[7]`http://www.oracle.com/us/products/middleware/`
[8]`http://www.sdn.sap.com/irj/sdn/nw-pi71` (13/10/09)
[9]`http://cxf.apache.org`
[10]`http://www.imagemagick.org`
[11]`http://www.lowagie.com/iText`
[12]`http://jquery.com`

and return the forms via postal mail or fax to consume the respective service.

However, for existing customers, banks already have many details such as name, title, residential address. With our solution, the data that is already present in the bank's data bases can be pre-filled into personalized forms. This is particularly useful for the data the customer is unlikely to use everyday, but required in the form nonetheless, such as investment fund numbers or insurance police identifiers.

To showcase this, we developed a Web application resembling the above-mentioned Suncorp website. This application is available online (cf. Section 5). Logged-in users can generate personalized versions of Suncorp forms. In the background, the Web application retrieves user data from a database, assembles a SOAP message, and invokes the FormSys Web service for the respective form with this message. The service then returns a URL under which the form can be downloaded. The fields in the generated form is still editable by the user.

Form Triggering a Business Process. When a form is filled in and the customer submits it to his bank, a set of business process at the bank are triggered by the customer's request. If these processes are implemented in software, the data from paper-based forms has to be provided to the software, and more often than not this is a manual process.

In contrast, if the data is available in AcroForm fields, then our `pdf2soap` solution can be used. After the filled-in form has been uploaded to FormSys, and the respective `pdf2soap` service is active, FormSys extracts the data from the AcroForm, assembles a SOAP message, and invokes the associated endpoint with this message.

Usually this endpoint would refer to an actual implementation of the business process that is triggered upon arrival of a respective form. For the demonstration purposes, we developed a simple Web service as part of the Web application: it accepts `pdf2soap` messages and writes the contained data into a file. When accessing the output page, the information from the file is displayed in a table.

4.2 Use case 2: Queensland Government

Driver Licence Request. When renewing or requesting a new driver licence in Queensland, an individual has to fill in up to six different forms. Most of these forms request some standard personal data, such as name, title, address, date of birth and the like. Also, an individual has to understand which forms are needed when.

We analysed the requirements and downloaded the respective forms from the Queensland Government, Transport Department website, starting from `http://www.transport.qld.gov.au/Home/Licensing/Driver_licence/`. On this basis we implemented an executable process which accepts the data needed in more than one form and forwards it to the respective FormSys `soap2pdf` services. The process has been deployed to an Intalio|Server[13], where users have access to a simple workflow: when starting process instances by providing input data, the required subset of the six forms is filled and returned to the user.

5. DEMO SCENARIO

The demonstration shows the personalized form generation from Use case 1 (cf. previous section) as a motivation, and then explains the underlying system, FormSys. We then demonstrate how to upload an AcroForm, edit field names, start a `soap2pdf` service, and view the dynamically generated WSDL file. We also explain and demonstrate `pdf2soap` in general, and how it is used in Use case 1, followed by Use case 2 where we show the executable process model and the different ways to interact with the user.

The different parts of the demonstrations can be viewed individually as screencasts from a Web site: `http://www.cse.unsw.edu.au/~FormSys/FormSys.html`. This Web site also links to the running implementation, which we encourage to test and to comment on.

6. CONCLUSION AND FUTURE WORK

In this paper, we showcase FormSys, a web-based system which Web service-enables form documents. The main functionality of FormSys is twofold: `soap2pdf` provides filled-in forms based on data in SOAP messages, while `pdf2soap` extracts data from filled-in forms and invokes a given Web service endpoint with it. We also discussed three real-world use cases, which make use of `soap2pdf` and `pdf2soap`. The running system is accessible online (cf. Section 5).

We conclude from this work that PDFs can be used as a channel for interacting with Web services: individuals can provide filled forms as input to Web services, and Web services can output filled form documents. The philosophy was to enable non-technical users to interact with our system.

In future work, we consider mechanisms to support some tasks for forms administrators, like field grouping and naming, through partial automation. One major open point is how to consume `soap2pdf` services. In order to allow end users to implement their personal processes themselves, we plan on extending a mashup tool with FormSys functionality. Another open point is dealing with other form types than PDF: while the architecture contains a flexible mechanism to plug in software to handle different document types, the concepts have yet to be implemented.

Acknowledgments

This work has been supported by a grant from the Smart Services CRC under the Service Delivery Framework project.

7. REFERENCES

[1] Adobe LiveCycle Designer ES2. `www.adobe.com/products/livecycle/designer/`.

[2] Acro Software. FormMax 3.5. `www.acrosoftware.com/`.

[3] Adobe Systems Incorporated. Acrobat Forms API Reference. Technical Note No. 5181, 2003.

[4] J. Becker, L. Algermissen, and B. Niehaves. A Procedure Model for Process Oriented e-Government Projects. *Business Process Management Journal*, 12(1):61 – 75, 2006.

[5] H. Déjean and J.-L. Meunier. A system for converting PDF documents into structured XML format. In *Document Analysis Systems VII, LNCS 3872*, 2006.

[6] H. Motahari, B. Benatallah, A. Martens, F. Curbera, and F. Casati. Semi-Automated Adaptation of Service Interactions. In *WWW'07*, 2007.

[7] OASIS. *Web Services Business Process Execution Language Version 2.0*, Apr. 2007.

[8] D. Woods. *Enterprise Services Architecture.* O'Reilly, 2003.

[13] `http://www.intalio.com`