Automating Form-based Processes through Annotation[*]

Sung Wook Kim[1], Hye-Young Paik[1], and Ingo Weber[1,2]

[1] School of Computer Science & Engineering, University of New South Wales
[2] Software Systems Research Group, NICTA, Sydney, Australia[†]
{skim,hpaik,ingo.weber}@cse.unsw.edu.au

Abstract. Despite all efforts to support processes through IT, processes based on paper forms are still prevalent. While they are easy to create, using paper-based forms puts a burden of tedious manual work on the end users. Automating these processes often requires heavy customisation of commercial software tools, and building new systems to replace form-based processes may not be cost-effective. In this paper, we propose a pragmatic approach to enable end users to automate form-based processes. The approach builds on several types of annotations: to help collect and distribute information for form fields; to choose appropriate process execution paths; and to support email distribution or approval for filled forms. We implemented the approach in a prototype, called *EzyForms*. On this basis we conducted a user study with 15 participants, showing that people with little technical background were able to automate the existing form-based processes efficiently.

1 Introduction

Business Process Management Systems (BPMS) are widely accepted software systems [1] that enable organisations to automate and continuously improve business processes in order to achieve better performance.

While all business processes work collectively to support business operations, BPMS solutions typically aim to support uniform and repetitive processes that will have significant impact on the performance of an organisation when automated [2, 3].

Although the benefits of BPMS have been widely recognised, there is still a large portion of business processes that are not adequately addressed by these systems. These types of processes make up the so-called long tail of processes, i.e. highly customised processes that are unique to individual organisations, or concern a small number of workers. Many organisations do not see significant ROI on implementing solutions to automate the long tail of processes [4].

[*]This work is supported by SDA Project (H2-H3), Smart Services CRC, Australia.

[†]NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

A full suite of BPMS is too technical and sophisticated for people with little or no technical background to use to build an automated solution of their own. Given the situation, it would certainly be of value to provide a coherent set of tools that may help those users equipped with the domain knowledge necessary for automation, but not the technical skills (e.g., business process modelling).

In this paper, we particularly focus on the long tail of processes that consists of form documents, as forms are one of the most prevalent artefacts in the types of processes in discussion. Forms provide a low-tech, quick and flexible way to manage processes. However, they impose a good deal of manual labour on the end users, such as searching/downloading required forms, entering the same information repeatedly, printing/faxing forms and so on.

Our approach is to enable the form users to model and deploy their *existing* form-based processes into a service-enabled framework without requiring technical knowledge or the cost of re-engineering. The key concept of our proposal is in various types of annotations on forms and effective applications of such information during the modelling and execution phases of form-based processes. Our contributions in this paper are:

- a prototype named *EzyForms*, a framework that supports the whole life-cycle of form-based process management. The framework includes the following novel aspects:
  - identification of different types of annotation for simple, implicit modelling and execution of form-based processes,
  - smart applications of the annotations to support basic data flows and execution patterns in form-based processes,
  - a WYSIWYG-fashion integration of forms in every stage of the process life-cycle, from form service creation through to modelling and execution.
- evaluation of such a framework in terms of applicability and usability through user studies.

The remainder of the paper is structured as follows. Section 2 further elaborates on the form-based processes. In Section 3, we discuss the annotations on forms and explain how the annotations are used in process modelling and execution phases. Section 4 describes a number of user supporting features for form-based process automation. The architecture and prototype implementation are described in Section 5. Section 6 shows the result of the user studies, followed by related work in Section 7. Section 8 concludes the paper and suggests some direction for future work.

2 The Form-based Processes

Figure 1 depicts the type of form-based processes in *EzyForms*. Although there are other types that do not conform to the shown model (e.g., a process involving more than one user), it is our observation that typical form-based processes exhibit the following characteristics. Our current work focuses on these initially.

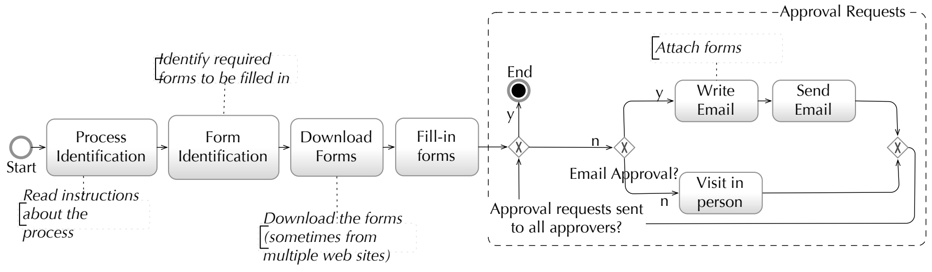

Start
Process Identification
Read instructions about the process
Identify required forms to be filled in
Form Identification
Download Forms
Download the forms (sometimes from multiple web sites)
Fill-in forms
End
y
Approval requests sent to all approvers?
Approval Requests
n
Email Approval?
y
n
Attach forms
Write Email
Send Email
Visit in person

Fig. 1. A form-based process expressed in BPMN

First, a form-based process consists of one or more paper-based forms which are eventually to be submitted to an administration unit to trigger or record an organisational process (e.g., a trip request). Second, a form-based process is initiated and executed by a single user (e.g., a trip requestor), and the user is normally responsible for finding information about the process (e.g., reading instructions on a web page, searching/downloading forms). Third, a form-based process involves obtaining zero or more approvals on completed forms, which could mean the user having to write multiple email requests and coordinating the chain of approval manually.

To elaborate on Fig. 1 further, let us take an existing form-based process as an example – a scenario at the School of Computer Science and Engineering (CSE) at UNSW. For work-related travels, the members of the school use a travel request process[3]. There are up to five forms involved in the process. Depending on the employment status or position held in the school, people need to choose a different set of forms, which are available from and managed by different organisational units within UNSW. Here, the end user is liable for proper execution of the process, including manual tasks such as searching/downloading required forms, filling them in (often entering the same information repeatedly), and submitting them for approvals or notifications.

Using *EzyForms*, we are able to transform the *'as-is situation'* as follows: an admin officer[4] at the school uploads the five forms into the system and matching Web Services are created. The officer annotates the forms, adding details such as employment status conditions, approver's name and email. The same officer (or another officer who is familiar with the travel request process) creates a process model which consists of the five forms, data flow mapping between forms and extra annotations if needed (e.g., email templates). The model is then deployed, ready for use by a school member[5]. Now the execution of the travel request process means that all forms are in one place, any optional forms are automatically identified via the conditions, no need for repetitive entering of the same data and automated dispatch of filled forms for approvals as needed.

[3]CSE Travel Procedures, `http://www.cse.unsw.edu.au/people/fipras/travel/`
[4]We will refer to this class of users as 'form owners' in the rest of the paper.
[5]We will refer to this class of users as 'end users' in the rest of the paper.

3 Proposed Approach

Our approach aims to enable form owners to automate the form-based processes, and allows end users to execute the process in more efficient manner. This approach consists of four steps: form upload, form annotation, process modelling, and process execution (shown in Fig. 2).

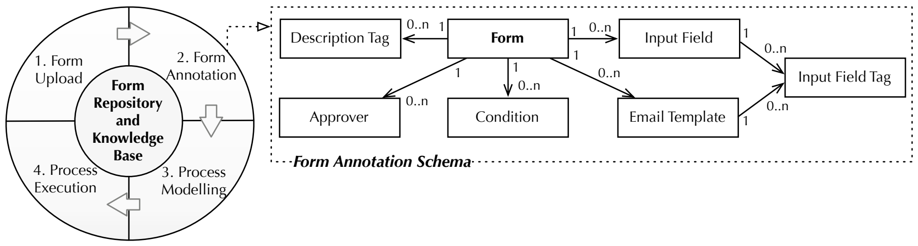


Fig. 2. A life-cycle of forms in *EzyForms* and the schema for form annotations

3.1 Form Upload

In order to fill-in the forms electronically, we convert a form into a Web Service. This is done by our previous work, FormSys [5], through which PDF forms[6] are uploaded to the central FormSys repository. All forms in the repository are available to other form owners to design and deploy new form-based processes. When a form is uploaded to the system, FormSys automatically generates two matching Web Services:

– **soap2pdf**: receives data from an application, fills a form with it, and returns a form via email or a URL where the filled-in form is available.
– **pdf2soap**: extracts the data from a filled-in form, assembles a corresponding SOAP message, which it sends to an application.

The creation of Web Service is based on WSDL/SOAP standards and the procedure is completely automated and hidden to the users. The rest of the life-cycle of forms in *EzyForms* extends these basic functions of FormSys.

Formally, we define $\mathcal{F} := \{F_1, F_2, \ldots\}$ as the set of all forms in the system, where each form has a set of fields $\mathcal{G}(F_i) := \{f_1, f_2, \ldots, f_n\}$.

3.2 Form Annotation

We recognise the fact that, to remove the need of BPM/IT professionals' involvement, the system must ascertain necessary information by itself as much as possible (e.g., which form is relevant for process X, which fields in form A

[6]To be precise, we use AcroForm, a sub-standard of PDF which contains editable and interactive PDF form elements.

are duplicated in form B). This is, of course, not always feasible or accurate. To bridge the gap, we introduce the form annotation step.

The form annotation is an act of adding metadata to the form. Adding metadata such as keywords or tags is an effective way of categorising a large pool of resource for both personal and public purposes [6, 7]. Due to its informality and flexibility in use, annotation has been widely accepted by users for assisting Web browsing or searching, and utilised in many systems like Flickr, Delicious, and Youtube [7, 8]

There are two types of annotations, each assisting different aspects of process automation: annotation for process modelling and for process execution. Also, note that the annotation is defined on two levels: the form-library level (i.e., the annotation is defined on the form generically, without regards to the processes in which the form is involved) and the process level (starting from the form-library level, the annotation can be refined within the context of a process – see Sect. 3.3).

Annotation for Modelling. This annotation is used to help form owners when modelling a new process.

- *Form description tag*: these are descriptive or representative tags that can describe the form. For example, the travel reimbursement form in our usage scenario may have form descriptions tags like `travel`, `reimbursement`, `travelcost`. We formalize the system-wide set of form description tags as $\mathcal{T} := \{t_1, t_2, \ldots\}$, and the annotation function $T : \mathcal{F} \mapsto 2^{\mathcal{T}}$ as a mapping from the forms to the description tags which apply to this form: $T(F_i) = \emptyset$ or $T(F_i) = \{t_{i_1}, \ldots, t_{i_k}\}$ for $F_i \in \mathcal{F}$ and $t_{i_1}, \ldots, t_{i_k} \in \mathcal{T}$ where k corresponds to the number of form description tags added to the form. These tags contribute to the search and discovery of forms.
- *Input field tag*: these are synonyms, alias or any descriptive tags for an input field in a form. For example, name field in the travel reimbursement form may have tags such as `name`, `staffname`, `fullname`. Formally, we write $\mathcal{I} := \{i_1, i_2, \ldots\}$ for the set of input field tags available in the system. The respective annotation function $I : \mathcal{G} \mapsto 2^{\mathcal{I}}$ is defined as a mapping from the form fields to the field tags: $I(f_j) = \emptyset$ or $I(f_j) = \{i_{j_1}, \ldots, i_{j_k}\}$ for $f_j \in \mathcal{G}$ and $i_{j_1}, \ldots, i_{j_k} \in \mathcal{I}$ where k corresponds to the number of tags added to the input field. These tags contribute to ascertain simple data flow between forms. That is, by comparing the tags associated with input fields from each form, as well as their respective text labels, we can postulate if any given two input fields share the same input data.

Annotation for Execution.

- *Condition*: This type of annotation specifies conditions under which the form should be used. We define the system-wide set of conditions as $\mathcal{C} := \{c_1, c_2, \ldots\}$, and the condition annotation function $C : \mathcal{F} \mapsto 2^{\mathcal{C}}$ as a mapping from the forms to the conditions which apply to this form: $C(F_i) = \emptyset$ or

$C(F_i) = \{c_{i_1}, \ldots, c_{i_k}\}$ for $F_i \in \mathcal{F}$ and $c_{i_1}, \ldots, c_{i_k} \in \mathcal{C}$ where k corresponds to the number of conditions associated with the form. The conditions on a form are a template for conditions in a process. Process-level conditions determine if the form should be filled by a particular end user at process execution stage. Details on all execution aspects are given below.

- *Approver*: This annotation type describes the names and email addresses of people who are responsible for approving some form (e.g., travel requests may need approval from the Head of School), and used when dispatching approval request emails. Formally, we write $\mathcal{A} := \{a_1, a_2, \ldots\}$ for the set of approvers stored in the system, and $A : \mathcal{F} \mapsto 2^{\mathcal{A}}$ is a function mapping from the forms to the approvers which apply to this form: $A(F_i) = \emptyset$ or $A(F_i) = \{a_{i_1}, \ldots, a_{i_k}\}$ for $F_i \in \mathcal{F}$ and $a_{i_1}, \ldots, a_{i_k} \in \mathcal{A}$ where k corresponds to the number of approvers associated with the form.
- *Email Template*: This annotation type specifies email templates for creating email content to be sent to approvers, where the filled-in form gets attached. The email templates available in the system are formally referred to as $\mathcal{E} := \{e_1, e_2, \ldots\}$, and the email annotation function as $E : \mathcal{F} \times A \mapsto \mathcal{E}$, a mapping from the forms and their respective approvers to the email template which should be sent to this approver for this form: $E(F_i, a_j) = e_k$ iff $a_j \in A(F_i)$, else undefined.

Note that the collected annotations on different forms by different form owners are centrally managed and shared via *EzyForms* Knowledge Base (KB). It is also noted that adding annotations is an activity separate from process modelling tasks and it is possible that annotation and modelling are done by different people.

3.3 Process Model

The process model is designed based on the following characteristics of form-based processes:

- they are purely form-to-form processes, that is, it is possible to describe the processes as multiple steps of fill-form activities
- they are a single sequential flow where conditions are used to determine optional part of the flow (i.e., which form is relevant for the current user).

In this section, we describe the the formal model for the association between form annotation and the process model.

Process Definition. The annotations associated with the form documents in a process are translated into the process model as shown in Fig. 3. A *process model* is defined as a 6-tuple $p := (\mathcal{F}|_p, C|_p, A|_p, E|_p, I|_p, O)$, such that:

- $\mathcal{F}|_p \subseteq \mathcal{F}$ is a projection from the set of all forms to its subset used in p.
- $C|_p : \mathcal{F}|_p \mapsto 2^{\mathcal{C}}$ is a function mapping from the forms in p to the conditions which apply to this form: $C|_p(F_i) = \emptyset$ or $C|_p(F_i) = \{c_{i_1}, \ldots, c_{i_k}\}$ for $F_i \in \mathcal{F}|_p$ and $c_{i_1}, \ldots, c_{i_k} \in \mathcal{C}$.

– $A|_p : \mathcal{F}|_p \mapsto 2^{\mathcal{A}}$ is a function mapping from the forms in p to the approvers which apply to this form: $A|_p(F_i) = \emptyset$ or $A|_p(F_i) = \{a_{i_1}, \ldots, a_{i_k}\}$ for $F_i \in \mathcal{F}|_p$ and $a_{i_1}, \ldots, a_{i_k} \in \mathcal{A}$.
– $E|_p : \mathcal{F}|_p \times A \mapsto \mathcal{E}$ is a function mapping from the forms in p and their respective approvers to the email template which should be sent to this approver for this form: $E|_p(F_i, a_j) = e_k$ iff $a_j \in A(F_i)$, else undefined.
– $I|_p : \mathcal{G}(F_k) \mapsto \{\mathcal{I}, \perp\}$ is defined as a mapping from a form field to zero or one field tag: $I|_p(f_j) = \perp$ (no field tag) or $I|_p(f_j) = i_j$, where $f_j \in \mathcal{G}(F_k)$, the form belongs to p: $F_k \in \mathcal{F}|_p$, and $i_j \in \mathcal{I}$.
– O specifies the order of the forms in p, and thus is an ordered permutation of $\mathcal{F}|_p : O = (F_{i_1}, \ldots, F_{i_k})$ where k corresponds to the number of elements in $\mathcal{F}|_p$, and $i_j \neq i_l$ for any $j, l \in \{1, \ldots, k\}$.

Almost all process-specific annotations are projections of their respective forms-library level counterparts. The exception is the field tags: where on the form library level sets of field tags can be annotated, on the process-specific level at most one field tag can be assigned to each field. Note that we do not require the annotations on the process level to be subsets of the library level.

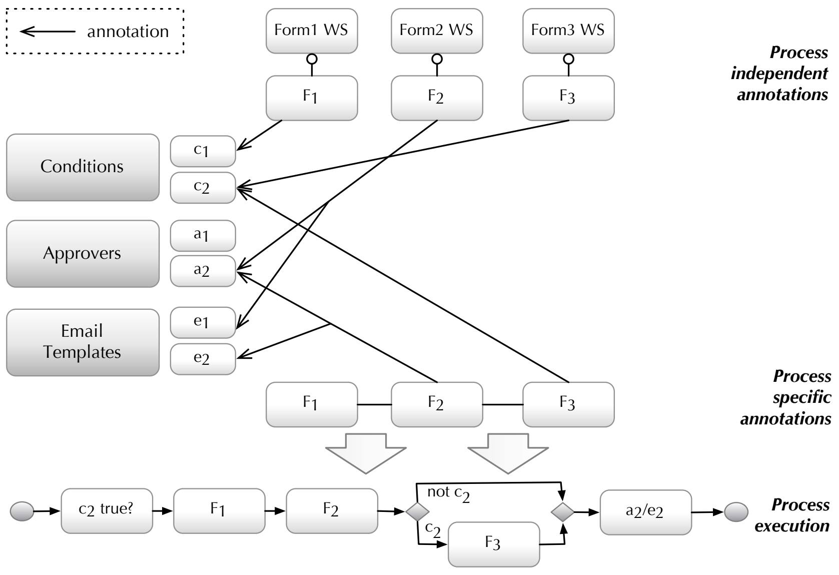


Fig. 3. Example relationships between forms, process, annotation, and execution (fields omitted for clarity).

3.4 Process Execution

We now explain how a process model in our approach is executed. When an end user starts an instance of some process model $p = (\mathcal{F}|_p, C|_p, A|_p, E|_p, I|_p, O)$, the approach first asks the user to determine the truth of all conditions used in the process (if any), as a set union: $\bigcup_{F_i \in \mathcal{F}|_p} C|_p(F_i)$. This means that conditions which are shared between forms are only evaluated once. The user selects whether the condition applies to her case, i.e., is true for this instance, or not.

Next, the forms without conditions, or whose conditions all were marked to be true, are executed. That is, a form F_i is only shown to the user if $C|_p(F_i) = \emptyset$ or c_j is true for all $c_j \in C|_p(F_i)$. The execution takes place in the order specified by O. Form execution here means that each form is shown to the user, who can enter values for the fields. The user can go back and forth between the forms.

The process-level field tags (zero or one per field) are hereby interpreted as data mapping: all fields tagged with the same tag are set to the same value. For a fixed but arbitrary $i \in \mathcal{I}$, this set of fields is $\{f \in \mathcal{G}(F) \mid F \in \mathcal{F}|_p, I|_p(f) = i\}$. This value equality is applied whenever the value for a field in this set is changed.

After filling all desired values into the forms, the user can trigger the next step in the process, where all filled forms can be downloaded in their original format. Finally, all approval emails are sent out for each form F_i without annotated conditions ($C|_p(F_i) = \emptyset$) or where all conditions $C|_p(F_i)$ are true.

4 User Supporting Features

As a way to help the form owners and end users with their tasks in-hand, we built a number of user supporting features throughout the system.

Recommendation for Adding Tags. For form owners, manually creating tags for a form (e.g., for input fields) can be time-consuming and tedious. To assist, our system includes recommendation algorithms for suggesting tags. This way, a tag can be chosen from a system-suggested list, or newly created by the user if none from the list is suitable. The new tag is added to the knowledge base for sharing and reuse.

We employ different recommendation strategies for different annotation types. For example, the algorithm for recommending form description tags will consider elements such as title text of the form, or free text appearing after the title (which is normally a description/instruction about the form) to derive potentially relevant terms. The algorithm for recommending input field tags calculates a weighted score for available tags based using computed value such as the similarity between the tags and extracted text (usually input field label) around the chosen input field. The details of the tag recommendation algorithms and evaluation results are explained in [9].

Candidate Generation for Mappings. For form owners, the mapping task in form-based process modelling involves identifying input fields that share the same value across forms so that the value is only request once from the user. During the modelling phase, the form owners will be asked to identify the mappings by examining forms. To assist, we automatically generate a suggested list for mapping candidates in the chosen forms.

Each input field has zero or more annotation tags associated with it. We take the tags in an input field with those appearing in other input fields. We then calculate the number of common tags between any two input fields. Based on the common tags, we generate a ranked list of mapping candidates. The larger the number of common tags, the more likely the input fields are to share the same input data.

Saving and Loading of Process Execution Data. For the end users, it may be useful to be able to save the form data they typed in during the execution of a process, and be able to reuse the same data next time the same process is required. To assist, we allow the end users to save the entire form data entered into a process into a local system file, after the execution. The file can be loaded into *EzyForms* when the same process needs to run.

5 *EzyForms* Implementation

A prototype has been implemented to evaluate the feasibility of our approach and its screencast is available at `http://www.cse.unsw.edu.au/~skim/ezyforms`.

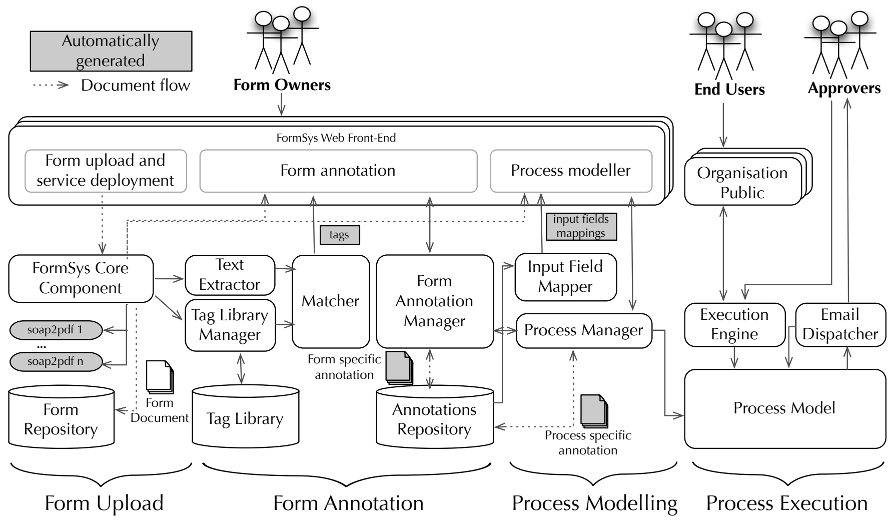


Fig. 4. *EzyForms* Architecture

The overall architecture is shown in Fig. 4, which is partitioned into *form upload, form annotation, process modelling*, and *process execution* components.

Form Upload. The forms uploaded into the system are stored into the repository by the *FormSys Core Component*. It is also responsible for `soap2pdf` service generation [5].

Form Annotation. `Matcher` component is responsible for managing tag library and tag recommendations. In `Form Annotation Manager`, forms are indexed using Apache Lucene[7]. The indices are created on the form's title text, file name, text content as well as the annotation tags.

Process Modelling. Based on the annotated input fields, `Input Field Mapper` generates mapping candidates for any selected input fields amongst forms during the modelling process. `Process Manager` stores and manages all processes created by the form owners.

Process Execution. When an end user instantiates a process, `Execution Engine` first, interacts with the end user to evaluate any conditions in the process, determines forms to be filled-in, presents the forms for input and executes the form services. Finally, it interacts with `Email Dispatcher` to send the forms according to the approval chain generated.

Approval Chains. From the approver annotations, we auto-generate an approval chain model that supports a multi-level hierarchy (e.g., supervisor, then Head of School, then the Dean) as well as "parallel-splits" (e.g., signed by two supervisors)[8]. From the approval chain model, *EzyForms* dispatches approval request emails to appropriate approvers and collate the responses. The current implementation reads the content of the email to determine if the request has been approved, that is, if the reply text of the email contains *approved or yes*, the request is considered approved and the next email is dispatched to the next approver in the chain. If *declined*, the process is terminated and the notification email is sent to the applicant.

6 User Study

We conducted a user study with fifteen participants. The goals of the study were to evaluate whether: i) our form annotation approach can be applied to people with little technical background (especially in BPM) to automate form-based processes, ii) user supporting features were helpful, and iii) end users find the automated execution of a form-based process convenient.

6.1 User Study Method Overview

We asked the participants to self-rate their knowledge on IT skills including business process modelling and automation. Later, we used the answers to categorise the evaluation results into 'experienced users' and 'novice users' groups.

All participants were given approximately 15 minutes of overview/induction session on the user study and the tool. The task scenario was based on a university student activity grant application process and the tasks were identical for all participants. We first introduced the participants the manual 'as-is' situation of

[7]http://lucene.apache.org/

[8]Note that it is possible to plug-in an algorithm which takes different interpretation of the given approver annotations

the process. The participants were given two paper forms[9] and asked to observe the forms and the process instructions. Then, using our tool, the participants played two different roles:

form owner: in this role, the participants performed form annotation and process modelling tasks. For the form annotation task, they first identified the form fields that they thought would frequently appear across other forms, and annotated those fields with descriptive tag names. The conductor of the study made observations on the tagging behaviour and usage of tag recommendation feature. For the process modelling task, the participants used the two forms to create an activity grant application process. The conductor of the study made observations on the use of data mapping options and mapping candidate generation feature. At the end of each task, the following questions were asked:

– How easy was it to understand and perform each task? - (Understanding how it works)
– How easy was it to understand the purpose of each task? - (Understanding how it is used)
– How easy was it to complete the task with a given tool? - (Tool usability)
– How helpful did you find the user supporting features for completing the task? - (User supporting features for form owner tasks)

form end user: in this role, the participants were asked to execute the process created using *EzyForms* and evaluate the experience against the 'as-is' situation. In order to emulate the execution of manual form-based process, we have prepared cards, each describing the steps in the form-based process (as described in Sect. 2), and asked the participants to put them in the order that they would perform if they were carrying out the process manually. Then, they completed the same process (but automated) using our tool. The participants were asked to evaluate how much improvement they saw compared to the manual process.

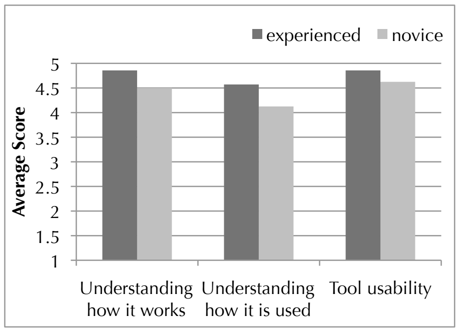
6.2 Results and Discussion

Out of the total fifteen, we categorised 7 participants into 'experienced users' and 8 participants into 'novice users'.

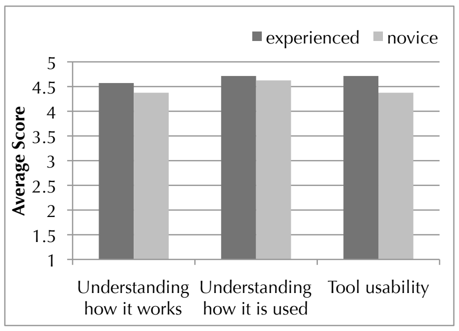
On the feasibility of the annotation approach. All fifteen participants were able to use the tool and complete the given set of tasks designed for the form owner's role – without any extra help, regardless of their respective category. The questionnaire results on the tasks as the form owner (Fig. 5) show that the understandability of tasks they have to perform, and the usability of the tool are scored high (well over 4 in a 5-point scale) across the user groups. Also, we can observe that the results show little differences between the two groups across

[9]http://www.arc.unsw.edu.au/get-involved/clubs-and-societies/club-forms-and-policy

the tasks. Hence, overall, we believe our proposed approach to automating the form-based process (and its implementation in *EzyForms*) is applicable to both groups and not bound to any process modelling experience.



(a) on Form Annotation Task



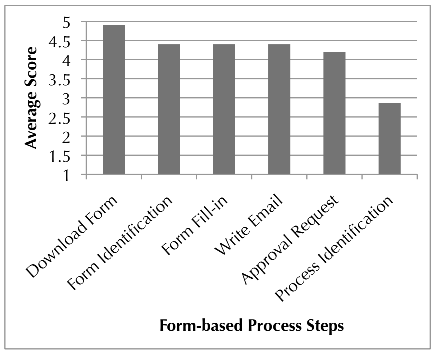
(b) on Process Modelling Task (incl. Data Mapping)

Fig. 5. Scores from the questionnaires on form owner's tasks - rated on a 5-point Likert scale (1 being worst, 5 being best).

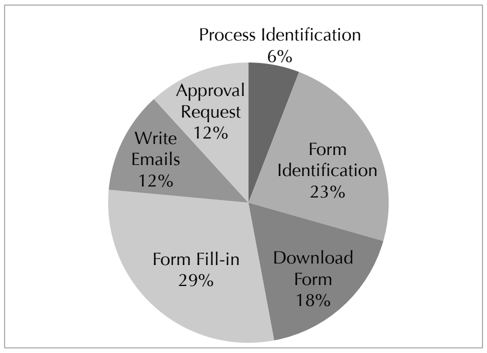
On user supporting features (form owner perspective). We also collected ratings for the user supporting features intended for form owners, namely the input field tag recommendation (during form annotation tasks) and mapping candidate generations (during process modelling tasks).

The participants scored 4.8/5 and 5/5 for tag recommendation, and mapping candidate generations features respectively. Also, on average, the input fields which did not have any tag recommendations had 1.17 tags, compared to 2.09 tags on those with tag recommendations. Out of 2.09 tags, 2.04 were selected from the recommended tag list. This observation shows that the participants were likely to add annotations (tags) more actively when there were recommendations made from the tool (about 80% increase in the number of tags) This also indicates that better recommendations may lead to better usability at the process modelling phase.

On process execution (end user perspective). All participants were able to complete the tasks given for the second role, form end user. Figure 6(a) shows the scores on the questionnaires which asked to rate the amount of improvement they saw at each steps of the form-based process, compared to the manual ones. All participants commented that *EzyForms* allowed them to conduct the process in more efficient manner. This is largely due to the fact that most manual work was either completely removed or significantly reduced (e.g., identifying which forms to fill-in, downloading forms to fill-in). Finally, each participant selected three favourite features from our approach without specific order. It shows that the most popular point was that they no longer had to fill-in same information repeatedly, closely followed by the point that they did not have to identify required forms by themselves (the conditions annotation in our tool automates that aspect) (Fig. 6(b)).



(a) Process execution steps (end user experience) - rated on a 5-point Likert scale



(b) Favourite features

Fig. 6. Form annotation user study evaluation

7 Related Work

Automation Through Annotation. Another field of area where annotation is actively being researched is semantic Web services. Semantic Web services aim at automating the discovery, selection, composition and management of Web services. The METEOR-S framework [10, 11] is a semantic annotation framework for Web services which uses schema matching techniques based on linguistic and structural similarity. [12] presents a practical method for semantically annotating collections of XML Schemas and Web service interfaces with incremental methodology for building ontology. Many research work in this category of annotation assumes an existing (formal) ontology which may not be feasible in many long-tail process scenarios. In [13], the authors proposed an approach to embrace user-centric aspect to enhance Web service discovery by providing collaborative tagging-based environment. We consider the work in semantic annotation of Web services as complementary to ours. One possible future direction in our work is to employ ontology to improve the recommendation and data mapping accuracy.

Forms Related Work. [14] developed a formal model that associates meaningful segments of documents with process activities which allows tight control over document related processes as well as customisation of document monitoring and content based interaction between WFMS and other applications. In [15], the authors propose active documents [16, 17] as a method for automating business processes based on form documents. Their research derives new concepts of active documents by examining meanings implied in paper form documents and explicitly express them in a knowledge representation language in which machines can understand and process. While their work enhances document processing with regard to the process execution, they require form designer tool to generate new forms which is stored as XML document, and the workflow engine to processes them. [18] proposes a generic Web service composition framework which uses forms as a metaphor to represent service's input and output. The system is also an extension of FormSys [5]. The extension focuses enabling visual service composition framework which generates executables in terms of BPEL.

Commercial Tools. We take two most well-known commercial tools in this area which are Microsoft(MS) SharePoint[10] and Adobe LiveCycle[11].

MS SharePoint is primarily a collaboration tool for enterprises. It implements form workflows using forms created from MS InfoPath. However, when dealing with existing paper-based forms such as MS Office documents, its capability is limited to storage, content management and routing of the forms. A workflow is supported only via explicitly designed process models, which is different from our implicit approach. MS Sharepoint also lacks in a functionality to reuse input data across different forms in the same process.

Adobe LiveCycle supports PDF, SWF, and HTML forms. It can assemble multiple PDF forms into a form package for end-users' convenience, and employs data extraction and pre-population. However, in order to automate the process with existing forms, a non-trivial amount of effort is required to develop a component that can transform data from one form to the other.

Commercial enterprise suite solutions impose management overheads such as number of programs requiring installation, integration with legacy system and training. *EzyForms* on the other hand provides a light-weight and cost-effective solution to a specific problem that is prevalent in today's business environment.

8 Conclusion and Future Work

In this paper, we have presented a pragmatic approach for automating form-based processes. In our approach, the form-based process model and execution are deliberately kept plain so that deriving their definitions for automation is attainable through tags and other simple form of annotations by the form owners. So far in this work, we have identified the types of annotations that can be used to support the end-to-end life-cycle of form-based processes. We have developed a fully working prototype named *EzyForms* as proof of concept. Our preliminary evaluation revealed that form annotation approach for automating form-based processes are applicable to people with little or no technical background, that the form annotation and process modelling task was intuitive enough to understand and complete. Part of the credit goes to the tooling support and user supporting features such as tag and mapping recommendation which the participants found useful and made active use of. The evaluation also showed that automated form-based processes significantly improved the overall user experience especially for form fill-in, form identification and form download tasks.

[10]`http://sharepoint.microsoft.com/`

[11]`http://www.adobe.com/products/livecycle`

In future work, we plan to explore the process data management issue so that *EzyForms* can deal with process specific data (e.g., id number of the project that will fund the travel) and sharing of such data between users. We will improve the tooling features including more complex input fields mapping (e.g., concatenating two strings or manipulating dates) and attaching files to the forms.

References

1. Weske, M.: Business process management concepts, languages, architectures. Springer, New York (2007)
2. Pesic, M.: Constraint-based workflow management systems: shifting control to users. PhD thesis, Eindhoven University of Technology (2008)
3. Hill, J.B., Sinur, J., Flint, D., Melenovsky, M.J.: Gartner's Position on Business Process Management, 2006. http://www.gartner.com/id=489533 (2006)
4. Larson, P.: BPM Suites and the Long Tail of Process Automation (2005) `www.philiplarson.com/docs/bpm-longtail.pdf`.
5. Weber, I., Paik, H.Y., Benatallah, B., Gong, Z., Zheng, L., Vorwerk, C.: FormSys: Form-processing Web Services. In: WWW '10 - demo, Raleigh (2010) 1313–1316
6. Ames, M., Naaman, M.: Why We Tag: Motivations for Annotation in Mobile and Online Media. In: CHI '07, San Jose (2007) 971–980
7. Cattuto, C., Loreto, V., Pietronero, L.: Semiotic dynamics and collaborative tagging. Proceedings of the National Academy of Sciences **104** (2007) 1461–1464
8. Golder, S.A.: Usage patterns of collaborative tagging systems. Journal of Information Science **32** (2006) 198–208
9. Kim, S.W.: Form annotation framework for long tail process automation. In: Workshop on User-Focused Service Engineering, Consumption and Aggregation, (to be published in 2012) (2011)
10. Patil, A.A., Oundhakar, S.A., Sheth, A.P., Verma, K.: METEOR-S web service annotation framework. In: WWW '04, New York (2004) 553–562
11. Rajasekaran, P., Miller, J., Verma, K., Sheth, A.: Enhancing web services description and discovery to facilitate composition. In: Workshop on Semantic Web services and Web Process Composition. (2004) 34–47
12. Kungas, P., Dumas, M.: Cost-Effective semantic annotation of XML schemas and web service interfaces. In: ICSOC'09, Bangalore (2009) 372–379
13. Chukmol, U., Benharkat, A.N., Amghar, Y.: Enhancing web service discovery by using collaborative tagging system. In: Proc. 4th Int'l Conf on Next Generation Web Services Practices, Washington, DC, USA (2008) 54–59
14. Bae, H., Kim, Y.: A document-process association model for workflow management. Computers in Industry **47** (2002) 139 – 154
15. Nam, C.K., Jang, G.S., Bae, J.H.J.: An xml-based active document for intelligent web applications. Expert Systems with Applications **25** (2003) 165 – 176
16. Ahonen, H.: Intelligent assembly of structured documents. Technical Report Report C-1996-40, Department of Computer Science, University of Helsinki, http://www.cs.helsinki.fi/TR/C-1996/40 (1996)
17. Dourish, P., Edwards, W., Lamarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D.B., Thornton, J.: Extending document management systems with user-specific active properties. ACM Tran. on Info. Systems **18** (1999) 140–170
18. Weber, I., Paik, H.Y., Benatallah, B.: Forms-based service composition. In: ICSOC'11, Paphos (2011) 627–635