Managing Long-tail Processes Using FormSys*

Ingo Weber, Hye-Young Paik, Boualem Benatallah, Corren Vorwerk, Zifei Gong, Liangliang Zheng, and Sung Wook Kim

School of Computer Science and Engineering
University of New South Wales, Sydney, Australia
{ingo.weber | hpaik | boualem}@cse.unsw.edu.au

Efforts and tools aiming to automate business processes promise the highest potential gains on business processes with a well-defined structure and high degree of repetition [1]. Despite successes in this area, the reality is that today many processes are in fact *not* automated. This is because, among other reasons, Business Process Management Suites (BPMSs) are not well suited for ad-hoc and human-centric processes [2]; and automating processes demands high cost and skills. This affects primarily the "long tail of processes" [3], i.e. processes that are less structured, or that do not affect many people uniformly, or that are not considered critical to an organization: those are rarely automated. One of the consequences of this state is that still today organisations rely on templates and paper-based forms to manage the long tail processes.

In this paper, we present a novel tool for enabling end-users to model and deploy processes they encounter in their daily work. We thus focus on a subset of long-tail processes, which we refer to as *personal processes*, i.e., business processes as experienced and described by a single person. Hence we can avoid many complicated modelling constructs, and devise simple process representations which can be as easily understood as a cooking recipe or an audio playlist. By focusing our efforts on a smaller set of usage scenarios, we aim to enable end-users to design their own executable processes, to support their everyday tasks. In earlier work, we built a system named *FormSys* [4] for creating form-filling Web services. Herein we present a completely new development as an extension of the earlier tool, therefore named *FormSys Process Designer*, for modeling and executing processes over such form-filling (and related) services. This can be useful to automate processes that involve filling the same information into multiple forms. E.g., when a new employee is hired at UNSW, the following units need to be informed: HR, IT support, procurement, the faculty, facilities, etc. Filling all different forms manually is a time-consuming task.

In order to enable end-users to design executable process models, we devised a simple control flow language for sequential process steps and conditional execution (or skipping) of process steps. The processes can be represented graphically or textually, as shown in Fig. 1. For the textual representation, we tried to stay close to natural language descriptions of how a process is executed; for the graphical representation we followed the analogy to composing a playlist in an audio player, such as using "cover flow". Data field equivalence between forms used in different process steps can be specified in a *mapping* part. Once the editing

* This work has been supported by a grant from the SmartServices CRC.

is completed, the process can be translated to BPEL. By doing so, an input Web form is automatically created, from which the data is distributed to the various steps in the process. Together with other artifacts, the BPEL process and the input form comprise an Intalio deployable package. By currently restricting focusing this work on processes around form-filling services, we achieve the desired simplicity but limit the scope of application to long-tail form-based processes. More details can be found in a technical report [5] and a screencast video, available at `http://www.cse.unsw.edu.au/~FormSys/process/`.

To our knowledge, no related end-user process modeling tool features the creation of executable processes. CoScripter (`http://coscripter.researchlabs.ibm.com`) in contrast is a tool for recording and automating processes performed in a Web browser. While closely related to our textual representation, it does not support Web service invocation or conditional execution.

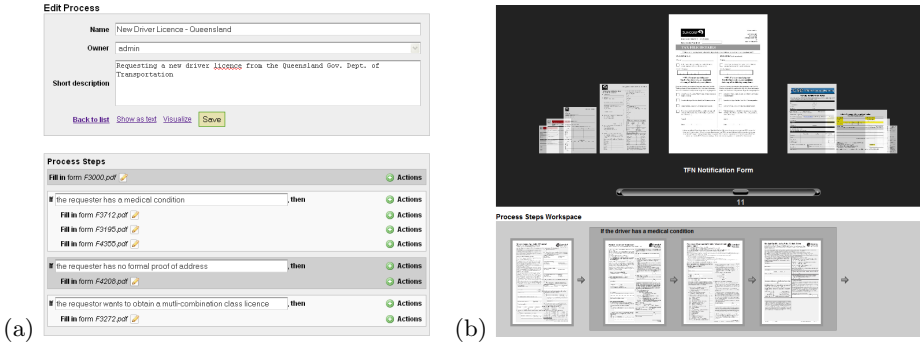

Fig. 1. Screenshots of the tool: textual (a) and graphical (b) process editing.

The demonstration includes viewing, editing, translating, and executing processes using our system. While the control flow part seems intuitive to the users, the mapping task requires further support, e.g., by semi-automatically creating the mappings, re-using mappings created by other users. From this work we want to expand the concepts, so as to support more generic scenarios involving arbitrary kinds of services and interactions between multiple people.

References

1. Leymann, F., Roller, D.: Production Workflow - Concepts and Techniques. Prentice Hall (2000)
2. Schurter, T.: BPM state of the nation 2009. bpm.com, `http://www.bpm.com/bpm-state-of-the-nation-2009.html` (2009) Accessed 25/11/2009.
3. Oracle White Paper: State of the business process management market 2008. `http://www.oracle.com/technologies/bpm/docs/state-of-bpm-market-whitepaper.pdf`, accessed 20/11/2009 (August 2008)
4. Weber, I., Paik, H., Benatallah, B., Gong, Z., Zheng, L., Vorwerk, C.: FormSys: Form-processing web services. In: WWW, Demo Track. (2010)
5. Weber, I., Paik, H.Y., Benatallah, B., et al.: Personal process management: Design and execution for end-users. Technical report, UNSW-CSE-TR-1018 (2010)