Forms-Based Service Composition

Ingo Weber, Hye-Young Paik, and Boualem Benatallah

School of Computer Science and Engineering
University of New South Wales
Sydney, NSW, Australia, 2052
{ingo.weber,hpaik,boualem}@cse.unsw.edu.au

Abstract. In many cases, it is not cost effective to automate business processes which affect a small number of people and/or change frequently. We present a novel approach for enabling domain experts to model and deploy such processes from their respective domain as Web service compositions. The approach is based on user-editable service naming, a graphical composition language where Web services are represented as forms, a targeted restriction of control flow expressivity, automated process verification mechanisms, and code generation for executing orchestrations. A Web-based service composition prototype implements this approach, including a WS-BPEL code generator.

1 Introduction

Business process management (BPM) refers to a discipline and software suites that automate, improve, and optimize business processes to enhance productivity [12]. Despite BPM's success, the reality is that today many processes are in fact *not* automated. First, among other reasons, BPM products are not suitably equipped to deal with processes that are ad-hoc [14]. Second, there are costs and high skills involved in implementing automated processes. This affects primarily the "long tail of processes" [9], i.e. processes that are less structured, that do not affect many people uniformly, or that are not critical.

In recent reports and studies [10,11], the split between BPM technology and its value for end-users is acknowledged. The reports include recommendations on increasing the relevance of BPM for end-users, allowing process changes by non-technical personnel, and reducing the complexity of BPM technology.

Motivated by the need for user-friendly BPM technology, the goal of this work is to devise an approach to support domain experts in their long-tail process automation needs. We focus on processes that can be implemented as Web service compositions. As a user group, we target business domain experts, i.e., non-IT professionals. We believe that these often have a good understanding of the processes they participate in; and that they are able to abstract from single process instances to the bigger picture of the process model containing alternatives and exceptions. An example is hiring a new employee, where HR recruiters have a good understanding of the default process and under which circumstances they may deviate from it.

The traditional approaches to BPM for process automation have inherited from programming. We believe this causes difficulties for the targeted users. The following lists the problems and requirements that are relevant to our goal:

G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 627–635, 2011.

- Programming requires writing code as abstract artifacts, symbolic or textual [5]. It is hard for untrained users to match their tasks to the abstractions.
- The so-called *selection barrier* [6] refers to the fact that often the users do not know how to express what they want the computer to do.
- Immediate program verification is needed to give feedback to the user [5].
- The system needs to understand high-level instructions of the user and translate them to a formal representation [5].

Our goal is to create a language and system for forms-based service composition, to allow domain experts to address their idiosyncratic, long-tail process automation needs themselves.Since we want to include processes where parts are executed conditionally, we propose a scripting approach to design process models. We focus on a graphical representation based on forms for designing processes. In the approach, services can be described using names that are meaningful to the user, and independent of the services' technical names. We use these names during service discovery and in the process design.

Also, we aim at keeping the complexity of process modeling low, in order to make the approach applicable to domain experts. As such, we include features to verify the correct combination of the modeled control and data flows through automated verification techniques. A code generator can automatically translate the models to an executable language. The complexity is further limited through a targeted restriction of control flow expressivity. However, to enable fast execution, we include an automatic parallelization technique in our code generation. We note that the approach outlined above is very different to other service composition or workflow tools (e.g., JOpera, Kepler, Taverna)[1], which tend to support highly complex modelling and programming capacity, and demand higher level of assumed knowledge from their users. We conduct a preliminary case study with use cases from the financial domain: data analysis processes such as finding a correlation between news and stock price changes. The use cases are taken from our industry partner, Sirca[2].

In summary, the contributions of this paper are the following:

- A forms-based composition method, where (i) forms are linked to Web services; and (ii) compositions can be modeled in a restricted, yet powerful generic language.
- Immediate automated process verification for reducing the burden on the user to build a correct composition.
- Automatic code generation with parallelization, to generate executable orchestrations from the forms-based composition language.

We also present a prototype and show how the approach can be enacted.[3]

[1] `www.jopera.org/`, `kepler-project.org/`, `www.taverna.org.uk/`, respectively

[2] `http://www.sirca.org.au`

[3] A full technical report and a demonstration video of the prototype can be found at `http://www.cse.unsw.edu.au/~FormSys/FormSys/`. The tool has also been demonstrated at the 9th Intl. Conference on Business Process Management (BPM) in August 2011, without publication.

2 Forms-Based Service Composition Approach

In this section, we explain how services are created and managed in a repository, how they are represented for domain experts, and how domain experts can then model processes graphically. We start by introducing a running example.

2.1 Use Case: News and Financial Data Analysis Process

Research and development within Sirca on the possible utilization of available datasets led to the implementation of numerous Web services [13]. The types of Web services range from query/search, data cleaning, to complex statistical analysis. Currently, each Web service is invoked by a simple user interface based on Web forms, and the services operate independently. One such example is described in Fig. 1, where each step represents a Web service.

1. Find news data: *e.g., news data on the company 'BHP'*
2. Find performance data: *e.g., hourly stock price summary for code 'BHP.AX'*
3. Merge datasets: *e.g., merge the result data sets from the first two steps*
4. Perform statistical analysis: *e.g., which news were possibly influential on the price*
5. Visualize dataset: *e.g., influential news and the prices*

Fig. 1. Financial data analysis from Sirca, for an Australian mining company 'BHP'

Repeating the above process by operating the Web forms involves around 30 mouse clicks, as well as entering the same information repeatedly at multiple steps. Once the processing is complete, the exact parameters that resulted in a given graph are lost. The set of changing parameters and the details of which service to use with which parameters differs between analysts and their tasks at hand. Therefore, while being repetitive, the processes are required to be flexibly executable or adaptable by the analyst.

Automating such processes is of interest to (i) reduce the required amount of user interaction, and (ii) retain the parameters that led to some visualization. The latter is important, because comparable graphs can be required periodically. With this motivating example in mind, we present our approach next.

2.2 Forms as Service Interface Representations

In our repository, every service is collectively represented by a WSDL document, a user-editable name, an icon, and forms as graphical representations of input and output messages. While WSDL needs to be present in the repository, it is completely hidden from the domain expert designing processes.

Graphical Representations: The service is a computational entity that performs some function, which is represented with an icon. For example, Figure 2 shows the icon for the "Find News Data" service.

Fig. 2. Find News Data service as icon

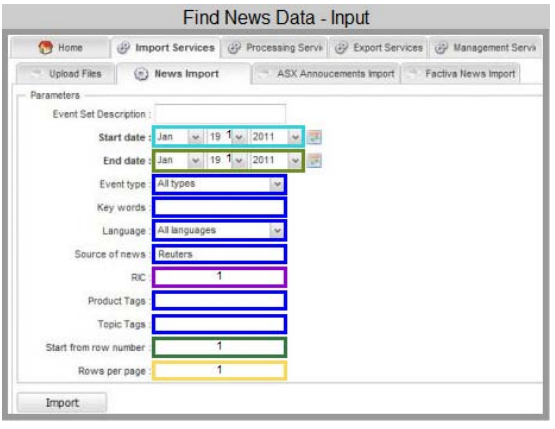

Fig. 3. Graphical representation of Input Message for Find News Data

The technical information about a service is stored as standard WSDL. In fact, a service in our approach corresponds to a WSDL operation. An invocable WSDL operation has an input and an optional output message[4]. The input and output messages for a service are represented as forms – reflecting the service's running user interface. Fig. 3 shows an example of an input message as a form. The names of form fields, corresponding to the service's input/output parameters, as well as the names of services themselves can be set to names meaningful to the users, which are used during search and composition of services. The form representation is also useful when the domain experts specify data mappings between messages.

Each data field from the message which is to used in process designs has a box associated to it, somewhere in the form. How the boxes correspond to data fields in the message has to be marked up manually for now, to enable automatic execution of designed processes. By default, the form could be rendered from the XML Schema type that belongs to the respective message. However, given our focus on domain experts, we believe that the form representation should be something the user is already familiar with. Hence, a screenshot of the UI through which the user commonly accesses this service makes a good representation.

Using Service Names Meaningful to Users: Besides the technical information from WSDL and the graphical representations, the services in our repository are also given non-technical names. These names are created by users, at the time when entering a service into the repository. In the process modeling tool, the user can assign different names to services, if they prefer them for the processes at hand. For instance, while some user called a service "Find News Data", another user may refer to it as "Import News Data".

[4] For simplicity, we currently neglect fault messages and exception handling, as well as certain XML Schema constructs and certain WSDL features.

2.3 Forms-Based Control Flow Modeling

Using the rich service descriptions outlined in the previous section, modeling an executable process becomes a matter of drag-and-drop and clicking. We treat control flow and data flow as two separate, but not independent, layers. The control flow serves as an abstract process description: which services should be executed, under which conditions and in which order? The data flow adds more detail, by specifying how the input and output message fields of the various services interact.

In order to retain the focus on domain experts, the control flow modeling is restricted: services are arranged into a single sequence, and may be subject to some condition. If the condition evaluates to true in a process instance, the associated services are executed; if it evaluates to false, they are skipped in this instance. The conditions are free text, and, through code generation, are turned into questions to users starting process instances. The above restrictions have strong impact on expressiveness[5]. However, anecdotal evidence from experience with industry contacts suggests that forcing the occasional user of our system to understand the particularities of the semantics of an expressive language alienates most targeted domain experts. Therefore, we try to keep the control flow modeling as simple as possible – see Fig. 4 for a screenshot from our tool.

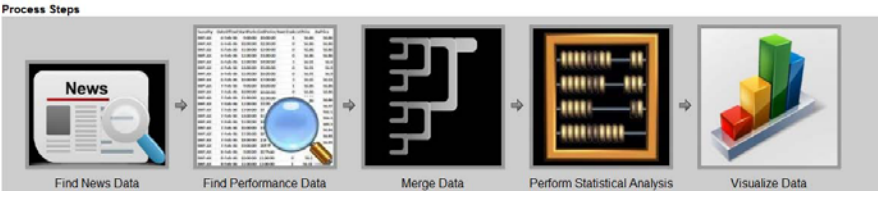

Fig. 4. Graphical Process Modeling of the Running Example

2.4 Forms-Based Data Flow Modeling

The data flow modeling works roughly as follows in our approach. Each service has an input and possibly an output message. A message consists of a set of fields, and has a form as graphical representation – c.f. Fig. 3. Data fields from one message can be mapped to data fields of another message in our approach. For instance, in our running example, the outputs of the two import services can be mapped to the merge service's input; the from/to dates in one data import service can be mapped to the date range of the other import service; etc.

Data fields of input messages fall into one of three categories, with respect to processes: user-static (i.e., always the same value for some user), process-static (i.e., always the same value for some process model), and process-instance-specific data (i.e., likely to be different for each instance of a process).

[5] In the technical report[3] , we discuss the expressiveness issue of our approach in detail with regards to workflow patterns.

In our method, the user can define certain kinds of mappings between / assignments to fields of messages of different Web services:

- specifying that an output field of one service corresponds to the input field of another (**output-input mapping**);
- specifying that two (or more) input fields of separate services will get the same value from the process-specific user input (**input-input mapping**);
- specifying a static value for an input field, including `null` (**static assignment**).

2.5 Process Verification

Implicitly, a data flow graph is created from output-input mappings (the directed edges of the graph). There can be contradictions within the data flow, or between the data flow and control flow of a process. Our approach includes an automatic verification for a set of problems that may appear. With the verification technique, the modeler can be kept from modeling, e.g., (illegal) loops in the data flow. More details, including a (semi-)formal treatment of the problems and solutions (based on well-known graph algorithms), are in the technical report[3].

2.6 Code Generation for Process Execution

The goal of modeling a process in our approach is to automate the execution of repetitive tasks. In order to determine the necessary input for the process, our solution combines all inputs for all services, and removes any field which is the target of a mapping or static assignment. The result forms a message with the consolidated input data format to start the process. For this message, we generate a Web form, where the user can enter the information and trigger an instance of the process. Analogously, the outputs of all services are consolidated to one output message of the process, for which again a Web form is created.

When desired by the user, our approach can parallelize steps in the process based on the data flow, by ignoring additional constraints from the control flow. This is achieved by mapping the problem onto graph algorithms, as before. The resulting, non-redundant process is then translated to WS-BPEL directly. Details can be found in the technical report[3].

3 Related Work

Here we present related work in brief; an exhaustive discussion of related work is included in the technical report[3].

Mashups have similar goals to our system. Topics such as data harvesting and visualization, composition of existing data and UI, and custom views or UIs for existing services are common in mashups [18]. While this may facilitate certain processes, the predominant composition paradigms in mashups are event-based synchronization [19] and data flow between components, not control flow over

process activities [4]. While there is some overlap between mashup approaches and ours, we see them as largely complementary.

End-User Process Modelling: Todor Stoitsev [15] investigates using Task Management (Outlook plugin) for "process modelling by example": the system tracks how people split up larger tasks into subtasks, and delegate some subtasks to others; this can be used as input to a workflow design tool. [8] describes a technique for constructing process models with formal execution semantics from informal models (e.g., Powerpoint drawings). The technique stops at producing BPMN, but possibly could be extended to generate executable models. However, the missing aspects to enable that (e.g., service selection, data flow) are not explored. [3] describes BPEL4UI / MarcoFlow: a language and tool for enabling BPEL designers to incorporate distributed UI composition in BPEL processes. Mashup-like UI components are synchronized between each other (for a single user), with UI components at other users, and the process. Microsoft InfoPath[6] essentially is a code-free software engineering tool. However, it is still for users familiar with programming, e.g., who know how databases work or what Web service are. *PICTURE* is a domain-specific modeling method and notation for public administration [1]. The key differences to our work are: PICTURE targets capturing the processes, and does not support creating executable processes; and PICTURE is domain-specific, whereas our tool is generic.

End-User Programming: A field of research that has a similar goal to ours, although in a different domain, is *end-user programming (EUP)*. EUP is the umbrella term for approaches that "make limited forms of programming sufficiently understandable and pleasant that end users will be willing and able to program" [2]. An approach in EUP that we consider closely related work is *CoScripter* [7], which primarily focuses on personal processes in the scope of browsing and using Web applications. The user can record, play and publish/share such browser processes on a public Wiki. The processes are stored in a simple end-user understandable language, using natural language keywords such as "go to <URL>" and "click on <link>". In CoScripter, all steps in a script need to be standard operations in a browser. While closely related to our work in terms of the understandability of process steps and the user focus, it does not support Web service invocation or conditional execution.

Own work: The work presented herein is also related to our own work. The tool in its current form has been presented (without publication)[3]. The tool and approach are significant extensions of earlier work for processes made up of PDF forms [17]. The PDF form-filling services are, in turn, the result of a separate work [16], with which FormSys Process Designer shares some database tables. The idea to relate Web services and their messages to forms stems from the earlier work, but representing arbitrary WSDL Web services through forms is a novel contribution of this paper.

[6] `http://office.microsoft.com/en-us/infopath/`

4 Conclusion, Discussion and Future Work

We have presented a forms-based service composition approach which allows domain experts with little technical knowledge to encode idiosyncratic, repetitive business processes themselves from design to execution.

Our preliminary evaluation revealed the following as desirable: user-editable input/output forms of the process; conditional data mappings; and process simulation in the design environment. For our use cases, the number of data mappings stayed reasonably small and therefore using colors to represent them was sufficient. However, if too many colors are required, they can eventually become confusing. These findings guide part of our immediate future work; in particular, a richer language for data mapping is under investigation, to enable more complex mappings and conditions over data fields. The tool will be further tested by our industry partner, and user studies will be conducted.

Acknowledgements. This work has been supported by a grant from the Smart Services CRC[7]. We thank Maurice Peat, Fethi Rabhi, Kader Lattab, and Angel Lagares Lemos for their valuable feedback.

References

1. Becker, J., Algermissen, L., Pfeiffer, D., Räckers, M.: Bausteinbasierte Modellierung von Prozesslandschaften mit der PICTURE-Methode am Beispiel der Universitätsverwaltung Münster. Wirtschaftsinformatik 49, 267–279 (2007)
2. Cypher, A., Dontcheva, M., Lau, T., Nichols, J. (eds.): No Code Required - Giving Users Tools to Transform the Web. Morgan Kaufmann (2010)
3. Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., Yan, L.: From People to Services to UI: Distributed Orchestration of User Interfaces. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 310–326. Springer, Heidelberg (2010)
4. Di Lorenzo, G., Hacid, H., Paik, H.-Y., Benatallah, B.: Data Iintegration in Mashups. SIGMOD Rec. 38(1), 59–66 (2009)
5. Harel, D.: Can Programming Be Liberated, Period? Computer 41, 28–37 (2008)
6. Ko, A.J., Myers, B.A., Aung, H.H.: Six learning barriers in end-user programming systems. In: VLHCC 2004, pp. 199–206 (2004)
7. Leshed, G., Haber, E., Matthews, T., Lau, T.: CoScripter: Automating & Sharing How-To Knowledge in the Enterprise. CHI Letters: Human Factors in Computing Systems 10(1), 1719–1728 (2008)
8. Mukherjee, D., Dhoolia, P., Sinha, S., Rembert, A.J., Gowri Nanda, M.: From Informal Process Diagrams to Formal Process Models. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 145–161. Springer, Heidelberg (2010)
9. Oracle White Paper. State of the Business Process Management Market (August 2008), `http://tinyurl.com/3c4u436` (accessed November 20, 2009)
10. Pettey, C., Goasdu, L.: Gartner Reveals Five Business Process Management Predictions for 2010 and Beyond. Gartner Press Release (January 13, 2010), `http://www.gartner.com/it/page.jsp?id=1278415` (accessed September 2, 2010)

[7] `http://www.smartservicescrc.com.au`

11. Reijers, H.A., van Wijk, S., Mutschler, B., Leurs, M.: BPM in Practice: Who Is Doing What? In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 45–60. Springer, Heidelberg (2010)
12. Richardson, C., Vollmer, K., Clair, C.L., Moore, C., Vitti, R.: Business Process Management Suites, Q3 2009 – The Need For Increased Business Agility Drives BPM Adoption. Forrester TechRadar For BP & A Pros (August 13, 2009)
13. Robertson, C., Rabhi, F., Peat, M.: Consumer Information Systems and Relationship Management: Design, Implementation and Use. In: A Service-Oriented Approach towards Real Time Financial News Analysis. IGI Global (2011)
14. Schurter, T.: BPM State of the Nation 2009. bpm.com, `http://www.bpm.com/bpm-state-of-the-nation-2009.html` (accessed November 25, 2009)
15. Stoitsev, T.: End-User Driven Business Process Composition. PhD thesis, TU Darmstadt, Fachbereich Informatik, Telekooperation (2009)
16. Weber, I., Paik, H., Benatallah, B., Gong, Z., Zheng, L., Vorwerk, C.: FormSys: Form-processing Web Services. In: WWW 2010: Proceedings of the 19th International World Wide Web Conference, Demo Track (2010)
17. Weber, I., Paik, H.-Y., Benatallah, B., Vorwerk, C., Gong, Z., Zheng, L., Kim, S.W.: Managing Long-Tail Processes Using FormSys. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 702–703. Springer, Heidelberg (2010)
18. Wong, J., Hong, J.: What Do We "Mashup" When We Make Mashups? In: WEUSE 2008, pp. 35–39 (May 2008)
19. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding Mashup Development. IEEE Internet Computing 12(5), 44–52 (2008)