BPMashup: Dynamic Execution of RESTful Processes

Xiwei Xu[1], Ingo Weber[1, 2], Liming Zhu[1, 2], Yan Liu[3, 2], Paul Rimba[1, 2], Qinghua Lu[1, 2]

1 Software System Research Group, NICTA, Australia
2 School of Computer Science and Engineering, UNSW, Australia
3 Pacific Northwest National Laboratory, USA
{first.last}@nicta.com.au, Yan.Liu@pnnl.gov

1 Introduction

While WS*-based Service-Oriented Architecture (SOA) is employed heavily in the enterprise application & integration space, end-user-oriented organizations such as Facebook, Google or Yahoo! adopted the REST paradigm. Web service ecosystems [1] have been established around web service offerings like social networking, where open platforms enable third-party developers to easily leverage the infrastructure provided by the social networks, to build web applications and plugged-in services for a massive user base. Such a web service ecosystem typically comprises a service provider opening up their product public service platform, a set of external value-added-resellers, and a community of users building and sharing customizations [2]. The lower layers of the traditional SOA-based WS* standards stack provide a loosely coupled infrastructure for Web service ecosystems. However, process layers on top of the standards stack introduce a comparatively tight coupling between the process logic and the WSDL interface definition [3], which tends to be brittle.

Composition of RESTful web services is usually achieved as light-weight Mashups – focusing on combining data from various sources, or handling events – or by using textual documentation to allow developers to understand processes involved. Traditional process-centric composition methods hardly fit the new paradigm.

REST and Resource-Oriented Architecture principles [4] are well established, and have been applied to web-based cross-enterprise business processes [5, 6] as an alternative way of implementing Web services. However, most existing approaches focus on building a RESTful facade to traditional service technologies without fundamentally using the REST principles. In traditional SOA, many approaches have been proposed to extend BPEL, e.g., with adaptation mechanisms using aspect-oriented programming [7] or rules [8]. These approaches still introduce tight coupling between process definition and Web service description. Some approaches [9] use WSDL-like descriptions for RESTful services, which arguably means losing most of the benefits.

In contrast, we present *BPMashup* in this demonstration: a framework that tailors REST principles towards process-aware information systems. BPMashup consists of the previously published server component, *RESTfulBP* [10], as well as a novel client-side JavaScript library – the Localized Process Execution Engine, *LPEE* – for executing processes and rendering UI widgets referring to individual service invocations. It has previously been shown that RESTfulBP can improve the adaptability and interoperability of process-aware systems [10]. Through the comprehensive framework of BPMashup, these benefits are applied also to processes combining services from more than one source. This is achieved by splitting business processes into distributed process fragments that are transferred dynamically at runtime.

A demonstration video and a technical report are available[1]. The report gives in-depth details on related work and technology, such as built-in security and encryption mechanisms and the coverage of workflow patterns.

2 BPMashup Overview

BPMashup provides a RESTful infrastructure for mashing up processes in web service ecosystems, using process fragments and dynamic next-step pointers that link to other services. A client-side process execution engine allows the processes to be executed at the edge of the system, to enable local decision making and improve adaptability of the business processes. While this demonstration focuses on the process execution phase, a BPMN-based modeling tool has been implemented[2], including a translation from BPMN to the artifacts required for process execution in BPMashup. Fig. 1 provides an overview of the system.

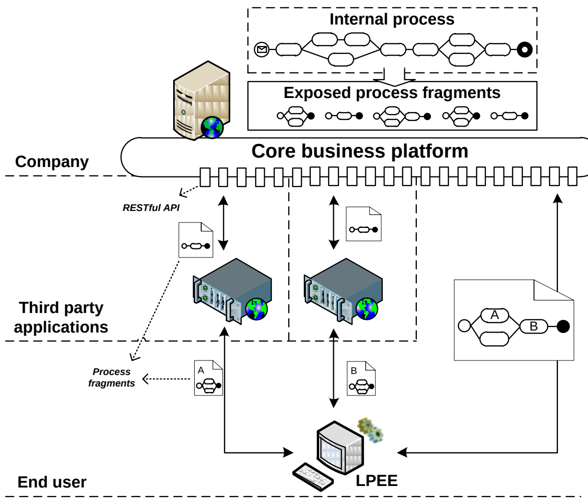


Fig. 1. BPMashup architecture.

The **business platform** exposes parts of the internal business processes of a company, as far as this is needed for partners and customers of the company to interact with a given process. The process coordination mechanism of BPMashup defines the exposed parts of processes as a set of loosely connected process fragments that can be transferred among participants to enable localized process execution.

[1] http://nicta.info/bpmashup

Third-party applications in a web service ecosystem based on BPMashup allow the external development of applications, faster co-creation and execution of cross-enterprise processes. The extensions are available as fragments as well.

BPMashup supports **end-users** through client-side process execution and localized decision making by providing LPEE (Localized Process Execution Engine). The end-users can thus drive the process execution, by selecting the most suitable process fragment from a group of candidate fragments in the current execution state.

Most notably, LPEE executes an overall composition (shown on the right in Fig. 1), which can include **sub-processes** that are provided by third-party applications (examples *A* and *B* in Fig. 1). A sub-process can be an atomic service invocation or a **process fragment**, which, in turn, may refer to other sub-processes. Furthermore, the messages exchanged with a service include both payload information and a process fragment's control flow. As such, sub-process implementations can be modified at any point in time, without breaking the overall composition. Process fragments from third-party applications can be included in the overall composition (e.g., fragments *A* and *B* in Fig. 1). However, due to the same-origin policy implemented in most browsers, all traffic of the JavaScript-based LPEE has to go through the platform.

3 Example Scenario

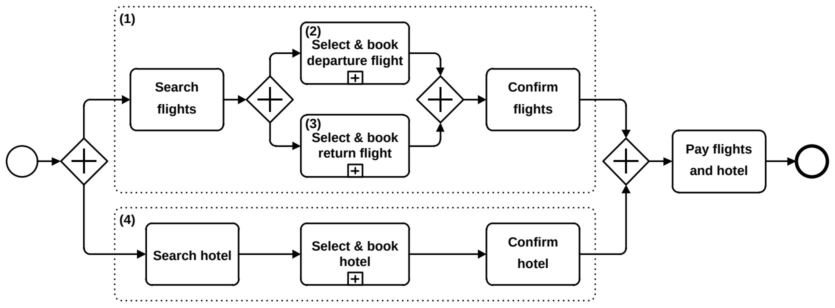
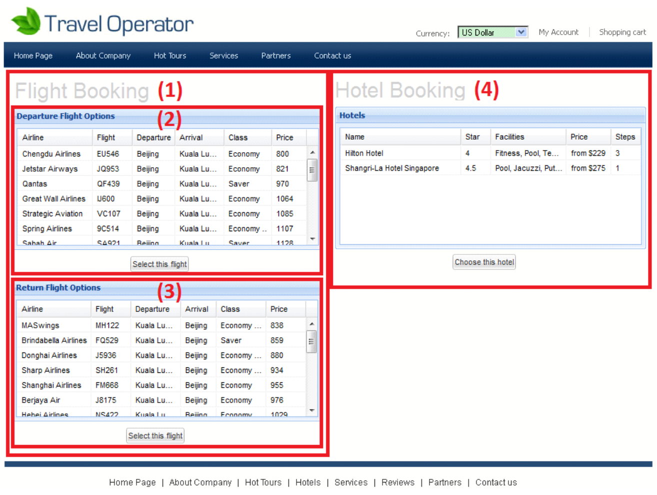


Fig. 2. Process model of travel agent, with enumeration of some areas.

For illustration purposes, we demonstrate BPMashup via the Virtual Travel Agent (VTA) example, as shown in Fig. 2. In BPMashup's VTA solution, all participating providers platformize their business as process fragments. For existing services, this means implementing a BPMashup wrapper. The hotel and airline partners are the third-party application providers, offering availability/price checking and booking in process fragments. The travel agent provides the platform's composition of these fragments, along with an integrated payment system. Messages from BPMashup include payload data, process fragments, and visualization information for steps. LPEE renders the process fragments according to the visualization instructions, as shown in Fig. 3, where the areas highlighted in red correspond to the enumerated fragments in Fig. 2.



Airline	Flight	Departure	Arrival	Class	Price
Chengdu Airlines	EU546	Beijing	Kuala Lu...	Economy	800
Jetstar Airways	JQ953	Beijing	Kuala Lu...	Economy	821
Qantas	QF439	Beijing	Kuala Lu...	Saver	970
Great Wall Airlines	IJ600	Beijing	Kuala Lu...	Economy	1064
Strategic Aviation	VC107	Beijing	Kuala Lu...	Economy	1085
Spring Airlines	9C514	Beijing	Kuala Lu...	Economy ..	1107
Sabah Air	SA921	Beijing	Kuala Lu	Saver	1128

Airline	Flight	Departure	Arrival	Class	Price
MASwings	MH122	Kuala Lu...	Beijing	Economy ...	838
Brindabella Airlines	FQ529	Kuala Lu...	Beijing	Saver	859
Donghai Airlines	J5936	Kuala Lu...	Beijing	Economy ...	880
Sharp Airlines	SH261	Kuala Lu...	Beijing	Economy ...	934
Shanghai Airlines	FM668	Kuala Lu...	Beijing	Economy	955
Berjaya Air	J8175	Kuala Lu...	Beijing	Economy	976
Hebei Airlines	NS422	Kuala Lu	Beijing	Economy	1029

Name	Star	Facilities	Price	Steps
Hilton Hotel	4	Fitness, Pool, Te...	from $229	3
Shangri-La Hotel Singapore	4.5	Pool, Jacuzzi, Put...	from $275	1

Fig. 3. Snapshot of hotel and flight selection page, where red boxes highlight certain snippets.

4 Conclusion

The BPMashup framework enables executing process-centric compositions of RESTful web services. Following REST principles, BPMashup decouples the relationships between the process participants, while allowing to hide internal business logic behind exposed process fragments. The process fragments are executed on the client side, allowing flexible process definitions which can be adapted dynamically: fragments can be changed at runtime, as long as the overall composition remains intact.

References

1. R. Veryard, "Ecosystem SOA," in Richard Veryard on Architecture, ed, 2009.
2. S. Jansen, et al., "A Sense of Community: A Research Agenda for Software Ecosystems," 31st International Conference on Software Engineering (ICSE'09), 2009.
3. C. Pautasso and E. Wilde, "Why is the web loosely coupled? A multi-faceted metric for service design," 18th International Conference on World Wide Web (WWW'09), 2009.
4. L. Richardson and S. Ruby, *RESTful Web Services*: O'Reilly Media, 2007.
5. H. Overdick, "Towards Resource-Oriented BPEL," The 2nd ECOWS Workshop on Emerging Web Services Technology (WEWST'07), 2007.
6. C. Pautasso, "BPEL for REST," 7th International Conference on Business Process Management (BPM'08), 2008.
7. A. Charfi and M. Mezini, , "AO4BPEL: An Aspect-oriented Extension to BPEL," Journal of World Wide Web 10 (3), 309-344, 2007.
8. L. Baresi and S. Guinea, "Self-Supervising BPEL Processes," IEEE Transactions on Software Engineering 37 (2), 247-263, 2011.
9. WADL, http://www.w3.org/Submission/wadl/ (accessed 06/10/2010)
10. X. Xu, *et al.*, "An Architectural Style for Process-Intensive Web Information Systems," 11th International Conference On Web Information System Engineering (WISE'10), 2010.