Sindh Univ. Res. Jour. (Sci. Ser.) Vol.44 (2) 345-348 (2012)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Towards Workflow Mashups

N.J. RAJPER, F. H. CHANDIO, Q.U.A. NIZAMANI, Z. KHOWAJA, H.A. NIZAMANI

Institute of Maths and Computer Science, University of Sindh, Jamshoro, Pakistan
Email: chandiofida@gmail.com : q_agro@hotmail.com : zak4@le.ac.uk: hanizamani@hotmail.com
Corresponding author: N.J. RAJPER, Email: nr76@le.ac.uk

Received 28th May2012 and Revised 10th June 2012

Abstract: Enterprises need to collaborate and share and resources, as well as costs and risks on-the-fly exists. This need arises from the challenge to remain competitive in an aggressive global market. Service Oriented Architecture (SOA) enables such collaborations. The current technologies enable innovative businesses that were not possible before, such as on demand composition of offerings from distinct providers to address customers needs. This was investigated in numerous approaches under the heading ”Mashups” which are concerned with users combining services. In this paper we propose a model to compose a service dynamically to address an arising opportunity for which no service is available assuming that simple or complex services exit to provide the functionality

Keywords: Service Oriented Architecture, Virtual Organizations.

1. INTRODUCTION

From small service providers to large enterprises, a need exits to collaborate and share expertise and resources, as well as costs and risks to survive and remain competitive in aggressive global market. Service Oriented Architecture (SOA) is the most prominent attempt to realize this cooperation and collaboration through information and communication technologies. This was investigated in numerous approaches under the heading ”Mashups” which are concerned with users combining services. So far service composition has always been a design-time or off-line issue. That means that so far workflows capturing the business operation are converted into executable workflows by IT staff. Activities or tasks in these workflows are executed by services that are usually specified explicitly in the workflow. This style has worked so far because it matches traditional business practice but it allows to increase pace. What is not necessarily being realized is that ICTs are enabling novel and innovative businesses never possible in traditional setting. In an increasingly user-demand driven market an additional deciding factor for success is the time-to-market. The implication of both the factors is that new services used to be made available when users demand them. In this sense, virtual organizations pool resources to provide products or services by pooling and adapting their resources for mutual gain.

Service composition achieve the same to some extent: it allows to pool the resources, however there is no notion of adapting them. Let us consider an example. A user wishes to book a holiday that includes an organized tour. There are two providers; one offering guided tours, the flight and hotel bookings but none offers both. Hence to satisfy the users needs both providers should cooperate. This demands that the underlying architecture should be able to bring and organize different cooperations and partners (with no previous relations or even possibly competing ones) dynamically to handle a project not possible for one partners to handle and consequently grasp an emerging business opportunity. SOA has the potential to fulfill these demands if ideas from virtual organizations are included and some controlled access to restructure workflows is provided.

2 BACKGROUND

2.1 Virtual Organization

There is no single agreed upon definition of Virtual Organization (VO) but most agree that it a timely creation of temporary alliance of geographically and temporally dispersed organizations/individuals that collaborate and share their resources to achieve some benefit not possible otherwise with the possibility of partners being unfamiliar (Cummings, 2008, Mamarinha, *et al.*, 2005). From this definition we can identify the factors

that make the VO concept interesting for us. These are timely creation, unfamiliar and possibly competing partners and resource sharing.

There are some open issues in VO, in particular that companies are reluctant to open their resources due to lack of trust. Besides being reluctant to trust strangers there is a good chance that even if they decide to trust and collaborate, interopretability and integration of different organizations is difficult. This issue is tackled by ECOLEAD project (ecolead, Camarinha-Matos et al., 2006).

All organizations or individuals interested in forming a virtual alliance create a permanent alliance called virtual Enterprize (VE). The VE takes all the measures necessary for timely creation of temporary collaboration such as resolving different infrastructure integration issues of the member organizations, standardizing data interpretability mechanism, setting trust and security policies for member organization, etc. The VE forms the Breeding Environment for VO.

2.2 SRML: SENSORIA REFERENCE MODELLING LANGUAGE

SRML (Fiadeiro, et al., 2006) is defined under the umbrella of the SENSORIA to operate at the higher level of abstraction of ”business” or ”domain” architectures. In SRML, composite services are modeled through modules. A module declares one or more components that are tightly bound and defined at design time, a number of requires-interfaces that specify services, that need to be provided by external parties, and (at most) one provides-interface that describes the service that is offered by the module. A number of wires establish interaction protocols among the components and between the components and the external interfaces. Components, external interfaces and wires are specified in terms of Business Roles, Business Protocols and Interaction Protocols, respectively. The specifications define the type of the nodes. Components are instances of Business Roles specified in terms of *i)* the set of supported interactions, and *ii)* the way in which the interactions are orchestrated.

2.3 STPOWLA

The Service-Targeted Policy-Oriented Workflow Approach (StPowla) (Bocchi et al., 2008) is designed to support policy driven business modeling over general SOAs. It combines workflows with policies and services which gives the flexibility of changing workflow instances through policies. These changes are applied at run-time and hence are temporary and tailored to different contexts. StPowla integrates three main ingredients: a graphical workflow notation, a policy language, and the SOA. The business process core is defined in terms of sequential, parallel and decision-based composition of building blocks called tasks, and finer details of the business process are expressed by policies. Tasks are executed by services.

3 CASE STUDY

To exemplify our approach we use a case study, which has a number of services, offered as part of workflows. The user requirements are discussed below.

The available services in the case study are a hotel booking and a flight booking service, offered as part of a composed service called TravelBooking as shown in **(Fig.1)**. Further, there is a TourBooking Service which is composed of two alternative tour packages: one offering a library and museum visit, the other a Safari park and country side visit as shown in **(Fig. 2).** Finally there is a a Transport service, providing car rental facility. Usually SOA would provide for each of these services to be used in their entirety or not at all.

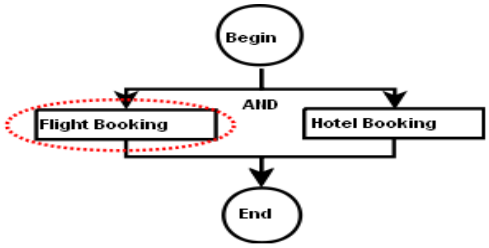

Fig.1: Travel Booking service

However, recall that service providers allow for some changes to their workflows. Each of these service’s providers have also published some part of its workflow, which allows for changes to the workflow, assuming that essential constraints are maintained.

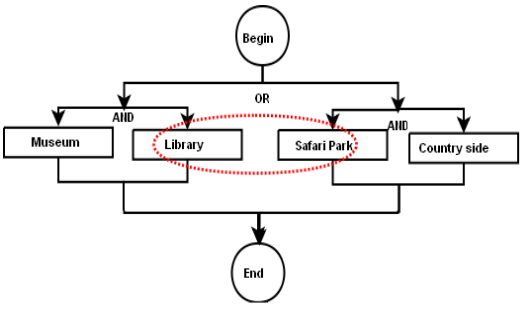

Fig. 2: Tour Booking service

Let us now assume that a customer visits this broker with the demands to get a service which provides him a flight booking facility. He is also interested, if that service also arranges for him a visit to local library and the Safari Park. However, if this

second requirement can not be satisfied, customer does not mind. But if the second requirement does get satisfied, the customer also requires a car to be rented for him in addition. This also says, that if car can not be rented then he doesn't want the"visit" service either. These demands look something like **(Fig.3).**

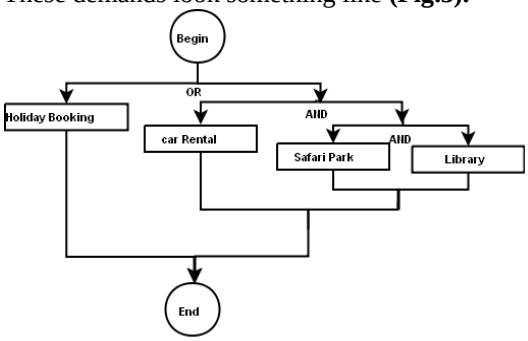

Fig. 3: Tour Booking service

The Service Broker first searches its repository. If such service is found, it is invoked, but if no such service is found the search result returns a list of services which partly satisfy the requirement. This start the process of workflow formation with the aim to satisfy the requirements. This workflow is presented to the user and forms the basis of a contract i.e., if the user accepts the workflow it will be executed (from the user's point of view). From a system point of view a few more tasks have to be completed. Once approved by the customer the next step is the realization of a composite service based on this workflow. Any service which can be configured to provide some of the tasks from workflow is selected and requested for the desired configuration. Finally when all the services are configured, data and control flow for the newly composed service is defined based on newly created workflow and the service is executed.

4. "MASHUP" Composite Service Module

As said before, in SRML each composite service is represented as a module. It typically has an orchestrator component which is charged with exactly the process and ensuring that the required services are found and invoked when required and the user is provided with the expected output. Here we extend this notion of composite service module to contain some additional functionality.

Each "mashable" composite service component will have: the usual **Orchestrator Component** extended with a"reconfiguration" mode. The orchestrator acts as usual it is placed in reconfiguration mode, when the normal flow will be changed by a policy component. the **Policy Component** interacts with the broker's workflow formation mechanism to establish the required changes to the workflow and instructs the orchestrator on the new behavior. In order to ensure that the orchestrator is instructed in a way that is consistent with the service provider's constraints the policy component negotiates with the Provider constraints Component. It is in this component that the provider stores information about the flexibility of the workflow, both in terms of what can be allowed and what cannot be allowed.

Considering deployment of these services, if a service is atomic, it is simply published in the repository as usual. When a composite service is published, it will also provide a list of services that the composite service wants to take part in different collaboration settings as well as the constraints. However it will not be mentioned whether those services are provider's own services or outsourced services.

This solution helps to keep the autonomy and privacy of the provider while opening new collaboration opportunities for the service. Note that the configuration is through the external broker and its interaction with the policy component is limited to deciding which components/functionalities the VE require and which it does not. For any service invoked for which there has not been any negotiation about the workflow, it is used in the traditional way that is in its entirety as offered.

5 2-Stage Mash-up Process

As we have seen before, we require two stages: that of constructing the workflow requested by the user and that of creating the mashup. We consider these in more detail now.

5.1 Stage-1: Workflow Development from User Requirements

First the user requirements are captured

i) The client selects required services based on a list of available services which of these are optional and which are mandatory (e.g in a holiday package a service specifies providing a visit to a museum could be optional while hotel and flight booking are mandatory).
ii) The client must also decide which services he considers complementary. Complementary services are those service which are not mandatory, but if one is provided further services must also be provided (the tour package in the example is of this type: it leads to a need for a car hire).
iii) For each service the client can also specify his/her preferences/demands which will provide nonfunctional properties to refine the choice of services made later (if there are options).

iv) Having obtained the requirements we turn the attention to constructing the workflow: Mandatory activities are placed in sequence, the order is chosen by the system automatically.

v) Next, optional services are added. Optional services are places in parallel to mandatory services, using an or-branch[1].

vi) Finally complimentary services need to be added. Complimentary services are usually tied more closely to one or more of the optional services and they are placed in parallel with those using an and-bound[2].

vii) The completed workflow will be approved by the user and then forms the basis of the contract under which the mashup will be formed and executed. One can see this step akin to common practice in other domains, e.g an architect will show the plans to a client to seek their approval. Upon approval we enter into stage 2, i.e., the creation and execution of the mashup.

5.2 Stage-2: Mashup Creation and Execution

i) There are already two types of composition modules in SRML, Service module and Activity module.

ii) There introduce a third module called mashup module. This module has:
a.) An external-Provides Interfaces for the end-customers. Serves Interface, through which the Business Process Workflow is passed as input.

b.) An orchestrator which creates the actual orchestration based on Workflow and creates the required External-Requires Interfaces.

It might seem that External-Requires interface is not required as it is assumed that the chosen services are available within the Broker. However it might be the case that there are many such services and then one would be chosen dynamically. Or if for some reason no suitable internal service is available, then having the flexibility is desirable.

The mashup module provides the interleaving of different service's workflows through a common External-Provides Interface. For example in the SRML TravelBooking Scenario when orchestrator positively replies to a bookTrip event, then it waits for the client to commit to it, and once the client commits only then orchestrator commits to hotel and flight booking and continues the next procedure. Now suppose that we want to book a tour as well, as in the case study. After the reply the mash-up does not sent a commit or cancel event, rather it invokes the TourBook service. Once the mashup module gets a positive reply from TourBook service, the commit or cancel event is issued to both services.

1 an or-branch is one where the related merge operation allows the workflow to proceed even if not all sub branches where successful

2 an and-branch is one where the related merge operation does not allow the workflow to proceed, unless all sub branches where successful

6 CONCLUSION

In this paper we propose an extension to the SRML composite service module, together with a process to take care of composing workflows. Our approach not only treats workflows as components that can be joined together, but also considers them as flexible building blocks that can be reshaped in order to help fulfill a business goal. This approach allows for service providers to engage in new, unforeseen collaborations in ways that make use of partial offerings by them and hence opens new opportunities for them.

REFERNCES:

Cummings, J., T. Finholt, I. Foster, and C. Kesselman, (2008) Beyond being there: A blueprint for advancing the design, development, and evaluation of virtual organizations, (Final Reoprt from Workshops on Building Effective Virtual Organizaions), Technical report, National Science Foundation.

Camarinha-Matos, L. M., H. Afsarmanesh, and M. Ollus, (2005) Virtual Organizations: Systems and Practices', Springer, Berlin, Heidelberg.

European Collaborative networked Organizations Leadership Initiative, viewed 01/5/2012, http://ecolead.vtt.fi.

Camarinha-Matos, L. M., and H. Afsarmanesh, (2006) Creation of Virtual Organizations in a breeding environment. In Proceedings of INCOM'06, St. Etienne, France.

Fiadeiro, J. L., A. Lopes, and L. Bocchi, (2006) A formal approach to service component architecture. Springer Verlag, Berlin, Heidelberg, 193–213.

Bocchi, L., S. Gorton, S. Reiff-Marganiec, (2008) Engineering Service Oriented Applications: From StPowla Processes to SRML Models. Springer-Verlag, Berlin, Heidelberg, 163–178.

Norman, T. J., A. Preece, N.R. Jennings, M. Luck, V. D Dang, T. D. Nguyen, V. Deora, J. Shao, W.A. Gray, and N. J. Fiddian, (2004) Agent-based formation of virtual organisations. Knowledge-Based Systems, 17 **(24):** 103-111.