Introducing collaborative Service Mashup design

Martin Vasko and Schahram Dustdar
{vasko|dustdar@infosys.tuwien.ac.at}

Vienna University of Technology
Institute of Information Systems
Distributed Systems Group
Argentinierstreet 8, 1040 Vienna, Austria

Abstract. The adoption of REST - an architectural paradigm focusing on resources - by various providers like Google, Facebook and Yahoo strongly influences traditional Business Process design approaches layered atop of Web service description language (WSDL) and SOAP based Web services. Beside the architectural differences manifested in integration challenges for existing business process environments this new paradigm eases service development and enables lightweight integration in Web-oriented architectures. This paper introduces a model-driven approach to integrate different domains into Service Mashups: a) Orchestration information derived from WS-BPEL processes, b) Coequal integration of RESTful Web services and WS-* Web services and c) Role-based collaboration of process participants. The introduced concepts are implemented as a platform to enable collaborative Service Mashup design. The prototype is realized as a Rich Internet Application to maximize design performance and user experience.

1 Introduction

The emergence of Web services adhering to the REST (Representational State Transfer) paradigm - referred as RESTful Web services - and the widespread adoption and dispersion by different providers[1] strongly influences traditional business process environments. The benefit of the exposed services for individual business needs evolved over time from general services like for example the Google Search to specialized services like access to Social Network portals - refer to the Facebook services[2] as an example - or commercial services like Amazon's Web services. Beside the increasing number of services the ease of integration and the flexibility to changes awakes the interest to integrate these services into existing business process management approaches. The architectural differences between established Service Oriented Architectures and newly emerging lightweight resource oriented architectures complicate this endeavor.

[1] The programmable web - a directory of Web APIs, http://www.programmableweb.com, last accessed on April 2009
[2] http://developers.facebook.com, last accessed on April 2009

Different approaches [1], [2] try to solve the integration hurdles from the WS-BPEL (Web Service Business Process Execution language[15]) perspective (Pautasso [1]) whereas others provide an abstract simplified language to enable a unique service integration (Rosenberg et al. [2]). The latter is closely related to the approach introduced in this work. Whereas we do not try to provide an alternative language we encourage the model-driven approach to maintain extensibility. The abstraction of orchestration information, service integration information and participant integration information results into a simplified process abstraction. The orchestration information is derived from business processes defined in WS-BPEL. The service integration information is derived from WSDL (Web Service Description Language [16]) based Web services and RESTful Web services. Due to the lack of standards for human participant integration this work derives the role model information from BPEL4People (WS-BPEL Extension for People [14]). This enables a structured collaboration on Service Mashups. Beside the generic role model of human participants, newly emerging RESTful Web services enable the programmatic access to Social Network platforms and thus intensify the interactions with human participants. Currently different providers work on the definition of a set of functions summarizing a common access to Social Networks[3]. These trends facilitate the combination of such services with existing services to Mashups. As a consequence, not even the designers of Service Mashups might have access to collaborative Service Mashup design tools through Social Networks but also the Mashups themselves comprise Social Network Services as part of the Service Orchestrations. The underlying concepts of these correlations are described in Section 3. Existing platforms to create and administrate Mashups (like Google Mashup Editor, Intel Mash Maker, Microsoft Popfly and Yahoo Pipes to name the most prominent ones) currently provide limited support for collaborative Mashup design. Our work introduces a model-driven approach to collaboratively design Service Mashups. We developed a Rich Internet Application prototype to exemplify the introduced concepts and propose an approach to model Service Mashups.

The work is organized as follows: Section 1.1 tries to clarify the definition of Service Mashups by providing existing definitions and relating akin appellations like Mashups and Business processes. Section 2 summarizes existing approaches and relates them to the approach introduced in this paper. Section 3 introduces collaborative service orchestration based on a generic role model derived from BPEL4People, a established specification in Web service environments. Section 4 motivates the need for collaborative Service Mashup design approaches on the basis of a well-known process scenario. The combination of orchestration information, service integration information and collaboration information resulted in the Service Mashup Abstraction described in Section 5. This abstraction is realized in a framework which is implemented as a Web-based prototype and deploys a Rich Internet Application. Finally Section 6 concludes the paper and gives an outlook of the Future Work.

[3] `http://www.opensocial.org`, last accessed on April 2009

1.1 Service Mashups

The increasing number and diversity of RESTful Web services exposed on the internet blurs an adequate classification of orchestration paradigms. In the domain of Web applications lightweight integration approaches - summarized as RESTful Web services - dominate the service infrastructure. In the domain of Enterprise Computing the "Big" Web service technology stack (SOAP, WSDL, WS-* specifications, WSBPEL, etc.) delivers interoperability for heterogeneous service infrastructures. The architectural differences of these two worlds result in challenging combination efforts (outlined by Pautasso et al. [3]). From the orchestrational point of view both provide interesting techniques: WSBPEL is an XML based orchestration language to formulate business processes in Web service environments. The orchestration of services is achieved by a set of activities providing simple and complex Web service interaction capabilities. In contrast to this specification Mashups - refered as combinations of Web API calls - currently lack a comparable notation. Mashups indicate a way to create new Web applications by combining existing Web resources and Web APIs. According to [4] Service Mashups combine WS-BPEL based orchestrations with RESTful Web services. This work adheres to this definition and extends the idea of integrating new paradigms by the use of a model-driven approach.

2 Related Work

Hoyer et al. [5] sketch design principles for emerging Enterprise Mashups. The authors identify several shortcomings in established implementations of Service Oriented Architectures due to: a) high technical complexity of the relevant standards b) their inflexibility to react on changing requirements quickly and c) missing involvement of actual end-users. By allowing end-users to compose collections of services according to their individual needs, Mashups empower end-users to create their own workspaces which fit best to solve their heterogeneous business problems.

Swashup DSL introduced by Maximilien et al. [6] provides a domain-specific language to represent Mashups. The proposed language is focused on the description and propagation of data in Mashups. The approach is implemented by the use of the Rails framework and identifies three main concepts of mashups: data and mediatiation, service APIs and a means to generate Web applications from the resulting mashups. Curbera et al. [7] introduce Bite, a workflow-based composition model for Web applications. Bite allows the definition of interactive, asynchronous workflows with multiparty interactions and enables comprehensive data integration.

HOBBES [8] deploys an Adobe Flex - based WSBPEL designer and enables the collaborative modeling and administration of Business processes. The approach implements a Rich Internet Application to design and share WSBPEL processes. In contrast to HOBBES the approach introduced in our work focuses on the collaborative design of Service Mashups.

Tran et al. [10] introduce a novel approach to integrate existing process modeling languages at different abstraction levels using the concept of an architectural view. Their approach overcomes the divergence in term of syntax, semantics and levels of abstraction of existing process modeling approaches. Our work derives the concept of integrating different modeling languages at different abstraction levels using the concept architectural view and applies this idea on the domains of orchestration information, service integration information and collaboration integration.

3 Collaborative Service Orchestration

With the increasing number of services exposed by different providers on the internet the combination of these services to Service Mashups gains popularity. The broadening of functionality enables the recombination of Services to advanced Service Mashups orchestrating services from different domains. This trend leads to increasing specialization of Mashups and requires know-how from different domains: For example HousingMaps.com[4] combines data from craigslist.org[5] with Google Maps[6]. This Mashup requires knowledge in the domain of real estate markets and web application development. The trend to interdisciplinary combination of Services is encouraged by the effort to coequal integration of RESTful Web services and WS-* Web services as exemplified in the sample process scenario in the next section. This evolution of process environments from closed company-intern systems to open Web service environments crossing organizational boundaries requires the collaboration of different experts on the design of Service Mashups. Existing platforms to administrate Mashups currently lack support for this endeavor. Even simple role models are not supported.

The importance of role models in collaborative environments was depicted by Ellis et al. [9]. In the domain of Service Oriented Architectures BPEL4People[7], a specification proposed by IBM and SAP, shape up as an effort to integrate and structure human participants based on roles in WS-BPEL based Business processes. The specification defines a generic role model consisting of three human roles:

- The *process initiator* triggers the process instance at its creation time
- *Process stakeholders* can influence the progress of a process instance, for example, by adding ad-hoc attachments
- *Business administrators* are allowed to perform administrative actions on the business process, such as resolving missed deadlines.

The role model covers the whole business process lifecycle from design time to runtime issues. To continue the combination of Service Oriented Architecture principles with REST based web application paradigms the approach presented

[4] `http://www.housingmaps.com/`, last accessed on April 2009
[5] `http://www.craigslist.org`, last accessed on April 2009
[6] `http://maps.google.com`, last accessed on April 2009
[7] WS-BPEL Extension for People, BPEL4People

in this paper introduces a role model derived from the BPEL4People roles. In contrast to BPEL4People the role model introduced in this paper covers design time issues only as the integration of runtime issues is part of the future work described in section 6. The derived role model consists of the following roles:

- *Creator* - the *Creator* is determined automatically by the infrastructure during design time and refers to the participant creating the initial Service Mashup design
- *Stakeholder* - the *Stakeholder* may influence the progress of the Mashup design by adding attachments, process notes or forwarding tasks but has no privileges to change the Mashup design
- *Administrator* - the *Administrator* is allowed to perform administrative actions on the Mashup design. In addition to the privileges granted to *Stakeholders Administrators* are able to change the Service Mashup design

Beside the role model the visibility and propagation of changes to the instance edited by different members is crucial in collaborative service orchestration. In contrast to real-time collaboration systems enabling the concurrent editing and session sharing between members the approach introduced in this work is realized regarding relaxed WYSIWIS (What-You-See-Is-What-I-See). A Service Mashup instance is locked exclusively by the member editing the instance. After propagating all changes to the server the instance is released and the involved members are able to check changes by loading the instance. This roundtrip approach adheres to the REST paradigms as Service Mashups are exposed as resources by the server. By updating the changes of the process model only, the introduced architecture minimizes server processing load. The disadvantages of late propagation of changes is compensated by full access to the Mashup design by the processing member.

4 Process Scenario

The coequal integration of RESTful Web services and WS-* Web services into established business process environments neglects the different underlying architectures of these two concepts. The growing interest and the low integration hurdles of RESTful Web services result in the dispersion into non-technical domains. This trend amplifies the coequal integration of mostly publicly available RESTful Web services and existing company-internal WS-* Web services into daily business processes.

To exemplify the previously stated assumptions we introduce a sample process scenario illustrated in figure 1. The process executes a revisited example of a travel agency using existing Web 2.0 services: A user submits a holiday request through the Web frontend containing details of the start date, the end date and the destination. This order is submitted automatically to the agency-internal Order Processing Web service to track incoming orders. After the successful completion of this Web service operation the order is assigned to a responsible travel agent by the agency-internal HR Service. This service administrates all

travel agents and automatically assigns the appropriate travel agent to the user request on the basis of the agent's expertise. The assigned travel agent prepares the user order. This is modeled in a separate process administrated by the travel agent. The travel agent searches for the best photos of the destination, plans on-site trips and searches for the cheapest hotels. After the travel agent arranged the on-site trip details and ordered them he assembles all details into the resulting trip. This step concludes the Process Order under his responsibility. He propagates the package containing the order details to the main process. The Order Manager responsible for the main process publishes the trip and responds to the user request.

The Web Application Expert is responsible for the appropriate execution of the process. He maps the service requirements defined by the travel agent and the Order Manager to available Web services. As already mentioned the task *Assemble Order* is performed by the Order Service. The assignment of the order request to an adequate travel agent is done by the HR service. Both services are company-internal WS-* Web services hosted on the Travel Agency servers. To enable the search for the best photos of the destination the Web Application expert decides to integrate the Flickr Web service. To plan on-site trips and calculate the distances he decides to use the Google Maps Web service. The search for the cheapest hotels is enabled by the Hotels Combined Web service. All of these services expose their APIs as RESTful Web services.

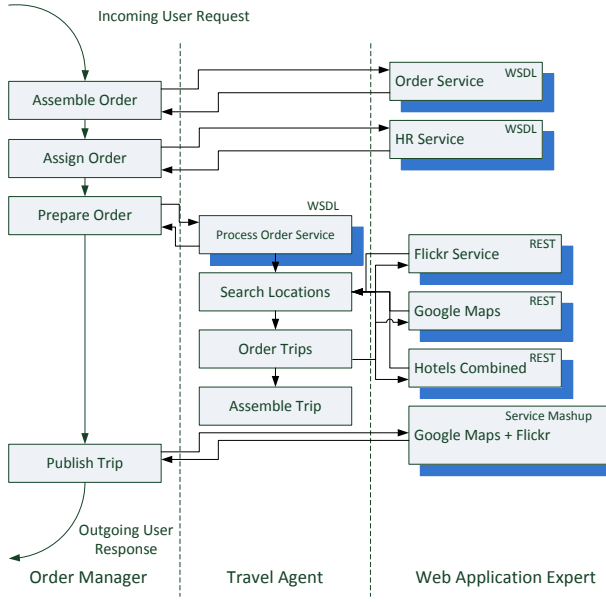

Fig. 1. Travel Agency scenario revisited

As a special service for the customer all on-site trips packaged in the resulting trip are exposed to the travel agency web site. The user is able to retrace the arranged trips on-site by the combinatorial use of Google Maps and Flickr. This Service Mashup was created by the travel agency to present a reproducible booking process to the customer. Before the user starts his trip he may familiarize with local details of the desired destination.

The process outlined in figure 1 exemplifies the coequal consumption of RESTful Web services and WS-* Web services. In the following section the abstraction is outlined to examine both Web service concepts coequally and motivate the use of role-based collaboration in the design of Service Mashups.

5 Service Mashup Abstraction

The scenario illustrated in the previous section outlines the blur of distinction of service integration into conventional process environments. Beside the coequal integration of RESTful Web services and WS-* based Web services the demand of human participant integration rises complexity of the process landscapes. In the SOA paradigm BPEL4People shape up as an accepted approach to introduce generic role models to structure human participant integration. As until now in the domain of Mashups no comparable approach is widely accepted this work proposes a role model derived from the BPEL4People model as introduced in section 3. Beside the two mentioned domains (Service Integration and Participant Integration) the integration of orchestration information into Service Mashups is of vital interest.

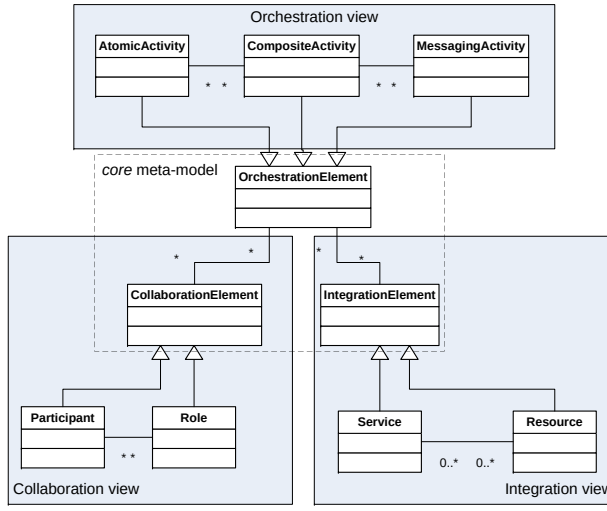

Fig. 2. The Service Mashup Meta-Model

In the WS-* based Web service environment WS-BPEL is the standard to describe and design the orchestration of Web services. WS-BPEL is layered atop of WSDL and decouples the orchestration logic from the service invocation logic by the use of Partner Links. The consequent separation of invocation details is reflected in the use of basic activities which refer to Partner Links and hide the concrete invocation mechanisms from the process definition. The Service Mashup Abstraction continues this separation of invocation details and extends the approach introduced by Tran et al. [10]. Tran et al. use the concept of an architectural view to integrate business process modeling languages at different abstraction levels. The approach maps process descriptions onto appropriate high-level or low-level views. In contrast to the approach elaborated by Tran et al. our work introduces a Control flow view and a Collaboration view on a higher abstraction level from WS-BPEL. This concept maps to the architectural views introduced by Tran et al.

The Service Mashup Abstraction emerges from the model-based integration of different domains into one model. Figure 2 illustrates the underlying meta-model.

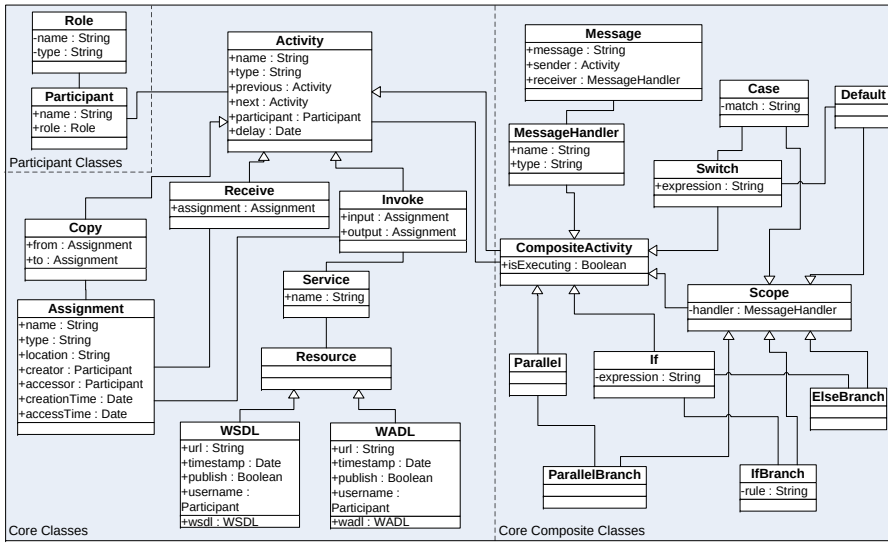

Fig. 3. The Service Mashup Abstraction represented using UML

The core meta-model consists of three elements each representing the base element for the particular view. This language design relates the views and enables a flexible combination of different views. Consider a combination of an Orchestration element like an Activity with a Collaboration Element like a Participant. As Orchestration Element is the parent of each Activity and is related to the Collaboration Element which represents the basis for Participant, each Activity

might have one or more Participants. The Integration Element is not directly connected to the Collaboration Element as the Service Mashup Abstraction reflects the decoupling principle of WS-BPEL to hide invocation details. The relation of Collaboration Elements with Service Invocations is achieved by linking elements of Orchestration Elements. A detailed class structure of the emerging Service Mashup Abstraction is illustrated in figure 3.

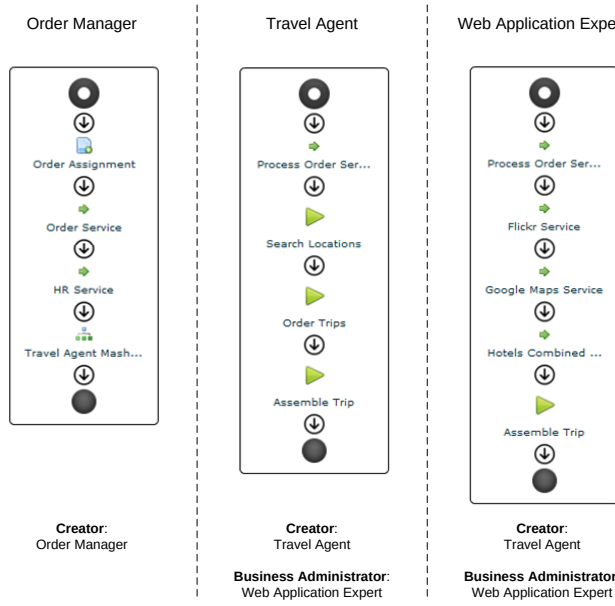

Fig. 4. Process Scenario resolved

```
<Process name="Web Application Expert Process" creator="Travel Agent">
  <Variables>
    <Variable name="Variable1" />
    ...
  </Variables>
  <Invoke ... type="WSDLInvoke" input="Variable1" output="Variable2">
    <WSDL id="1c21aefb-78d2-4963-af0f-9ba8f16df294" operation="GetValues"/>
  </Invoke>
  <Invoke ... type="RESTInvoke" input="Variable5" output="Variable6">
    <Resource uri="http://api.flickr.com/services/..." />
  </Invoke>
  <Invoke ... type="RESTInvoke" input="Variable7" output="Variable8">
    <Resource uri="http://maps.google.com/maps?..." />
  </Invoke>
  <Invoke ... type="RESTInvoke" input="Variable9" output="Variable10">
    <Resource uri="http://www.hotelscombined.com/api?..." />
  </Invoke>
  <Activity name="Assemble Trip" type="Activity" user="martin"/>
</Process>
```

Listing 1.1. Web Application Expert Process

Applying Service Mashup Abstraction on the process scenario introduced in section 4 results in the models illustrated in figure 4. The Order Manager orchestrates two WS-* Web services and refers to the Travel Agent process. The Travel agent models the abstract actions Search Location, Order Trips and Assemble Trip. After modeling this sequence she adds the Web application expert as Business Administrator to the Service Mashup. The Web Application Expert has now full access to the Service Mashups and refines the abstract activities by the according Service invocation elements. The resulting Service Mashup is depicted in listing 1.1. The process model is exposed by the prototype as a resource and might by accessed by HTTP GET operations.

5.1 Service Integration Pitfalls

The abstraction of Service descriptions comes with a risk to neglect the original Service integration paradigm. This might lead to a flippant orchestration of Web services. To prevent misleading orchestrations of Web services this work introduces pitfalls occurring in the coequal integration of RESTful Web services and WS-* Web services.

REST Service enforcement According to Fielding [12], a central concept in a resource oriented architecture is the focus on resources. To adhere to the REST paradigm a uniform interface provides stateless access to resources. Requests to RESTful Web services should include all data needed to fulfill a certain service function. The current trend to expose services of existing Web applications by the use of RESTful Web services to a broader audience blurs these conventions. Consider the RESTful access to a Photo sharing portal[8] as an example: The RESTful Web service client requests a list of 10 photos and exposes these photos on a Web site. The visitor of the web site can navigate through the lists of photos. Each navigation step triggers the RESTful Web service client to request the next list of 10 photos from the photo sharing portal. Figure 5 a) illustrates a stateful service design: The Web service increments and stores the lists to be able to respond to the next request. The second invocation trace in figure 5 b) illustrates a stateless RESTful Web service: The Web service client includes in the invocation, which list of photos it requests. All data needed to fulfill the request is included. This design ensures scalability (Web services are distributable across different load-balancers) by shifting state management responsibility to the client.

The trend to enforce a remote procedure call alike behavior by exposing RESTful Web services must be kept in mind during the design of Service Mashups consuming REST APIs.

[8] f.e. flickr services `http://www.flickr.com/services/api/`, last accessed on April 2009

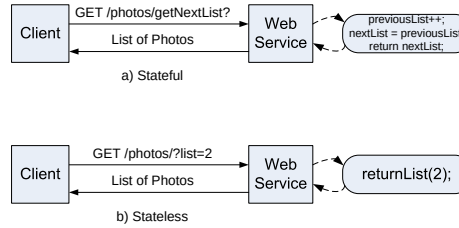

Fig. 5. REST Service enforcement

Protocol neglect HTTP provides different methods for requests[9]. To minimize integration hurdles existing RESTful Web services tend to reduce the HTTP protocol method support to GET and POST requests only. This results in exposing functions like sending a POST request to a RESTful Web service to delete an existing resource. Pautasso et al. [3] refer to the classification of supported HTTP verbs as *Hi-REST* and *Lo-REST*. The former refers to the support for four verbs (`GET`, `POST`, `PUT`, `DELETE`) and the latter refers to the use of `GET` and `POST` verbs only.

Concerning Service Mashup design *Hi-REST* architectures ease the integration of RESTful Web services by adhering stricter to the REST paradigm. The support for `PUT` and `DELETE` verbs indicate more clearly the operation on the resource and therefore the implications on the overall Service Mashup. RESTful Web services adhering to the *Lo-REST* architecture do not provide transparent access to the resources and therefore bear the risk to misleading orchestrations in Service Mashups.

These Service Integration pitfalls might be expanded to a collection of Antipatterns [13] as part of a Service Mashup framework in future work.

6 Conclusion and Future Work

This work introduces a model-driven integration approach towards: a) coequal integration of RESTful Web services and WS-* Web services, b) coequal Web service orchestration and c) collaborative Service design of Service Mashups. The approach derives concepts from established standards and provides an emerged Service Mashup Abstraction. The approach is exemplified in a web-based prototype. The implementation enables the asynchronous design of Service Mashups by implementing a Rich Internet Application. The prototype is reachable under `http://www.expressflow.com`.

Section 3 introduces the role model derived from BPEL4People. This role model is currently limited to design time issues of collaborative Service Mashup design. The expansion of this role model to runtime issues is planned for future work.

[9] For an exhaustive list of methods the interested reader may refer to RFC 2616

Section 5 introduces the Service Mashup Abstraction from the consequent model-driven abstraction of different Web service domains. Whereas the WS-BPEL based business process abstraction seems to be straight forward the co-equal integration of WS-* Web services and RESTful Web services comes with diverse risks. The pitfalls described in this work might be expanded to the definition of Antipatterns as part of the future work.

References

1. Pautasso, C.: BPEL for REST. 7th International Conference on Business Process Management (2008)
2. Rosenberg, F., Curbera, F., Duftler, M. J., Khalaf, R.: Composing RESTful Services and Collaborative Workflows: A Lightweight Approach. IEEE Internet Computing (2008) 24–31
3. Pautasso, C., Zimmermann, O., Leymann, F.: RESTful Web services vs. "Big" Web services: making the right architectural decision. Proceedings of the 17th international conference on World Wide Web (2008) 805–814
4. Benslimane, D., Dustdar, S., Sheth, A.: Service Mashups: The New Generation of Web Applications. IEEE Internet Computing (2008) 13–15
5. Hoyer, V., Stanoesvka-Slabeva, K., Janner, T., Schroth, C.: Enterprise Mashups: Design Principles towards the Long Tail of User Needs. IEEE International Conference on Services Computing (2008) 601–602
6. Maximilien, E. M., Ranabahu, A., Tai, S.: Swashup: situational Web applications Mashups. Conference on Object-oriented programming systems and applications (OOPSLA) (2007) 797–798
7. Curbera, F., Duftler, M., Khalaf, R., Lovell, D.: Bite: Workflow Composition for the Web. Proceedings of the 5th international conference on Service-Oriented Computing (2007) 94–106
8. Held, M., Blochinger, W.: Collaborative BPEL Design with a Rich Internet Application. 8th IEEE International Symposium on Cluster Computing and the Grid (2008) 202–209
9. Ellis, C. A., Gibbs, S. J., Rein, G.: Groupware: some issues and experiences. Communications of ACM (1991) 39–58
10. Tran, H., Zdun, U., Dustdar, S.: View-Based Integration of Process-Driven SOA Models at Various Abstraction Levels. Model-Based Software and Data Integration (2008) 55–66
11. Van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases (2003) 5–51
12. Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine, PhD Thesis (2000)
13. Brown, W., Malveau, R., Mowbray, T.: AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. Wiley (1998)
14. WS-BPEL Extension for People, IBM and SAP Specification `http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/` last accessed April 2009
15. Web Services Business Process Execution Language, OASIS Standard `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html`, last accessed April 2009
16. Web Service Description Languag, W3C Technical Report `http://www.w3.org/TR/wsdl`, last accessed April 2009