A Taxonomy of Variability in Web Service Flows *

Sergio Segura, David Benavides, Antonio Ruiz-Cortés and Pablo Trinidad
Department of Computer Languages and Systems
University of Seville
email:{segura, benavides, aruiz, trinidad}@tdg.lsi.us.es

Abstract

The combination of Software Product Lines (SPL) and Service-Oriented Architectures (SOA) development practices is expected to become a new development paradigm maximizing reuse and business integration. However, multiples issues must be still addressed in order to clarify the connections between both fields. One of the key questions to answer is how SPL practices can be used to support service-oriented applications. In this context, identifying and managing the points of variability in composite Web services emerges as an inevitable step for making possible such integration. In this position paper we give a first step toward such direction by introducing a comprehensible overview of the main variability points in Web service flows.

1 Introduction

Software Product Lines (SPL) [8] and *Service-Oriented Architectures (SOA)* [18] approaches to software development pursue different goals from a common perspective: software reuse. On the one hand, SPL focus on managing commonalities and variabilities among a set of related software systems. On the other hand, SOA enable assembly, orchestration and maintenance of service-based solutions implementing business processes.

Contributions about the connections between both development approaches, SPL and SOA, are starting to emerge in the SPL community [22]. However, multiple issues must be still addressed for studying how SPL practices could support the development of service–oriented systems. In this context, a relevant issue to be analyzed is how managing variations for specific customers or market segments in SOA.

Service–oriented applications are not tied to an specific technology. However, most common implementation of SOA-based systems use Web services as a suitable integration technology. A Web service is a software system designed to support interoperable machine-to-machine interaction over a network using Web standards protocols [2]. The main goal is to achieve interoperability among applications in a language and platform independent manner. However, the real strength of Web services is obtained when combining them and orchestrating them in order to deliver added-value services. In this context, Web Service Flows (WS-flow) are a common way for implementing composite Web services in SOA. WS-flows are composite Web services implemented using a process-based approach. Roughly speaking, a WS-flow process defines an executable business process in which participants are Web services.

Research in the field of variability in conventional Web services [12, 16, 19] and process workflow [7, 10, 11, 15, 20] is merely addresed in the literature. In [13] a high-level classification of approaches to WS-flow adaptability is presented. A more technological classification of WS-flow variability points in service invocation is introduced by the IBM staff in [9]. However, an explicit classification of the main variability points in WS-flow is still missed.

In this paper we give a first step toward a proposal for managing variability in WS-flow in the context of SPL and SOA. In particular, we first introduce WS-flow and BPEL. Secondly, we describe and classify the main variability points in WS-flow. The goal is to provide the starting point for a base of knowledge about variability in WS-flows that can be later used for both, $i)$ evaluating the different mechanisms for implementing variability in WS-flow and $ii)$ identifying factors that affect the selection of such variability mechanisms.

The remainder of this paper is organized as follow: in Section 2 WS-flows and BPEL are introduced. The main variability points identified in WS-flows are described in Section 3. Finally, we summarize our main conclusions and describe our future work in Section 4.

*This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web–Factories (TIN2006-00472)

2 Web Service Flows

A *Web Service Flow (WS-flow)* is a composite Web service implemented using a process-based approach [13]. Similarly to conventional process workflow, WS-flows specify set of tasks which are executed by the participants of a process. Additionally, a WS-flow defines the execution order of tasks, the data exchange among the participants and the business rules. In contrast with traditional workflows, the main characteristic of a WS-flow is that it works with a single type of participants: Web services. Figure 1 depicts an example of a WS-flow of a travel agency for travel arrangement. The WS-flow invokes the Web services of different airlines, car rental companies and hotels offering to the customer a value-added service for travel reservation.

There exist multiple proposed languages for defining WS-flows such as WSCI [21], BPML [4] or BPEL [14]. However, the *Business Process Execution Language (BPEL)* is recognized as *de facto* standard in this area. BPEL introduces basic and structured activities, control structures such as loops and conditional branches, synchronous and asynchronous communication, etc. Although BPEL processes are defined in XML format, most development IDEs provide a graphical notation for it. Once a BPEL process is defined it can be executed in any BPEL-compliant execution engine such as activeBPEL [1]. The execution engine orchestrates the invocations to the participants Web services according to the process definition.

3 Variability in WS-Flows

In this section we explore the main variability points in WS-flow. In particular, we focus on the variability in the invocation of services and the workflow structure. Variability in other advanced aspects of services such as security is out of the scope of this paper because of space constraints.

3.1 Service invocation

A service invocation is an activity in which the workflow invokes another service and exchange messages with it returning control back to the workflow. Figure 2 summarizes the main variability points identified in the invocation of services using a feature model. In particular, we have identified four main variability points:

- **Binding Time**. The selection of the service to invoke can be performed either during the development or the execution of the workflow. In the first case, the service reference is defined in design-time forcing to redeploy the workflow if changes in the participants need to be done. On the other hand, most flexible approaches propose selecting participants in run-time making the application adaptable to changes in the execution environment. Additionally, partner selection during run-time can be performed either by the user or automatically according to some selection policies.

  Figure 3 shows a possible implementation of run-time automated partner selection using a so-called service registry [3]. First, the information of the services (e.g., different airlines Web services) is registered in a service registry. Then, the workflow send a query to the registry to determine a matching service according to a set of parameters (e.g., a service with time of response $\leq$ 10s) and the predefined selection policies. Finally, the service reference obtained as a result of the query is used to invoke the matching service.

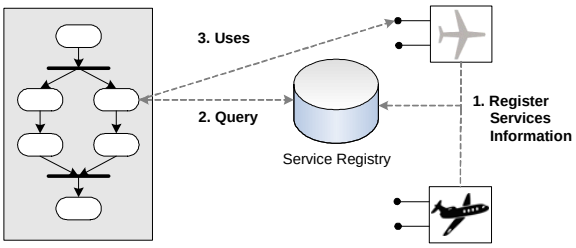

Figure 3. Service Registry

- **Partner Selection Criteria**. Selection criteria help to determine which of the available services offering the same functionality will be selected for its invocation [17]. In this context, two main variability points are identified:

  Evaluation Context. Selection criteria can be either hard-coded in the workflow or delegated to an external entity. The first option is very limited since workflow and selection criteria are highly coupled. On the other hand, defining the selection criteria in an independent manner is a preferred approach since it allows managing changes more efficiently. Figure 4 depicts an example in which the selection criteria are defined out of the scope of the workflow. Notice that changes in the selection criteria would be welcome since they would not affect the workflow.

  Definition Time. Selection criteria can be modified either in design-time or run-time. Similarly to the partner selection, the first option force to redeploy the workflow to responds to changes meanwhile the second alternative is much more flexible since it allows adapting the process workflow dynamically.

- **Messages Exchange**. Messages exchange between executable service workflows and other services are typically performed using two different communication

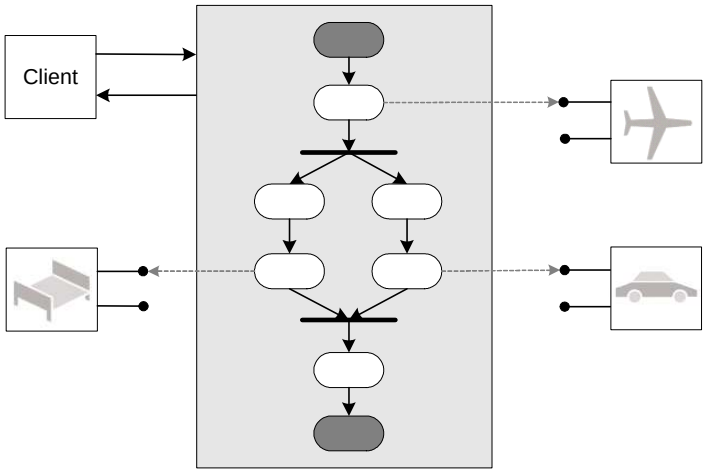

Figure 1. A possible WS-flow for Travel Arrangement

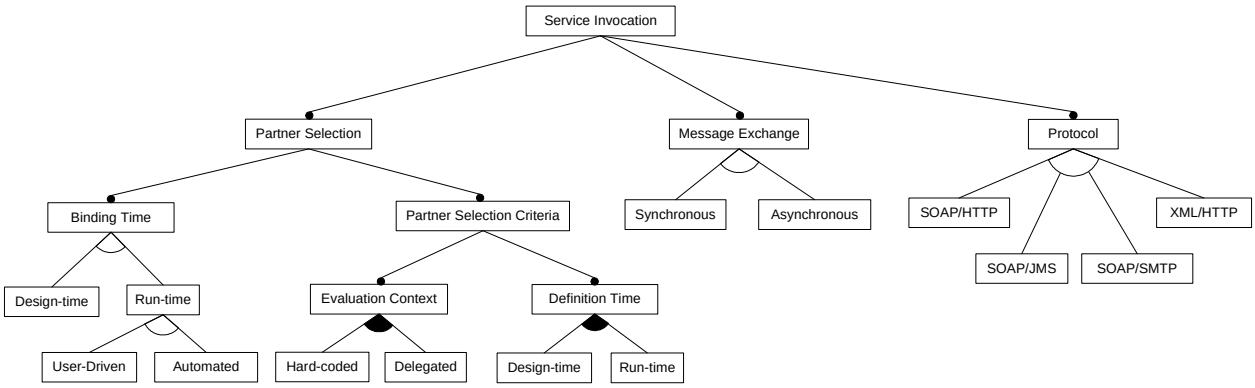

Figure 2. Variability Points in Service Invocation

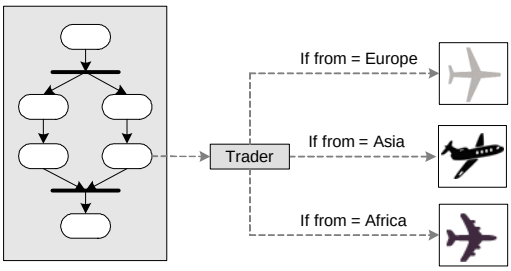

Figure 4. Workflow-independent selection criteria

patterns: synchronous or asynchronous. Synchronous request/response message exchange consists on sending a request message to the service and wait for it to response. Although this is the most common and natural approach, it is clearly not feasible if the services require significant time to response since it blocks the workflow processing. Hence, when the participant services can take a long time to response and such response is not needed for workflow processing, an asynchronous pattern is typically used.

In the asynchronous model the communication is performed between two workflows, the so-called service provider and service requestor or client. In this situation, the client need not block on the call. Instead, the client implements a callback interface, and once the results are available, the service provider simply makes a callback invocation on the client. Figure 5 illustrates an example of asynchronous messages exchange.

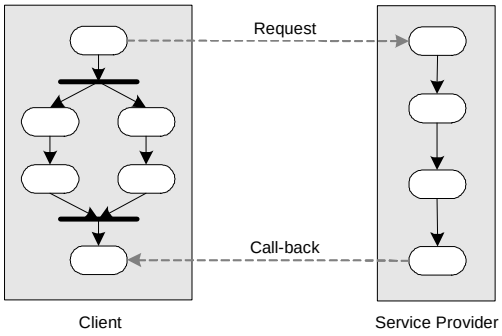

Figure 5. Asynchronous Model

- **Protocols**. Multiples protocols can be used for service interactions over a network, i.e. SOAP/HTTP, SOAP/JMS, XML/HTTP, etc. Thus, the selection of a suitable set of protocols for the communication with services is a key variability point.

3.2 Process Workflow Structure

The process workflow structure determines all the aspects related to the way in which the process is executed: the execution order, the data exchange between participants, the business rules, the errors treatment, etc. Hence, two main variability points are identified in this context:

- **Control Flow**. The workflow structure determines the tasks to be executed, the execution order and even the type of participant in the process. Therefore, the control flow will commonly have locations likely to change in response to changes in the business process. Hence, for instance, suppose the travel agency decides to change the order in which flight fares are consulted for certain customers, e.g. prioritizing low-cost airlines for young people.

- **Data Flow**. During the execution of a WS-flow participants exchange different kind of data in XML format. Similarly to the control flow, data is likely to change as consequence of changes in the business process. As an example, suppose the travel agency is asked to provide additional security information in the cases in which passengers travel to a specific country.

4 Conclusions and Future Work

In this paper we expose the need for an explicit classification of variability in WS-flow as a starting point for handling variability through services in the context of SPL and SOA. In particular, we identified and classify the main variability points in the invocation of services and the workflow structure. In some cases the distinction between development-time and run-time is exposed explicitly because of its relevance. However, we emphasize that the time in which variability is resolved will depend mainly of the technology used.

Many challenges remain for our future work. Once the main variability points are identified is time to consider the available technological approaches for implementing it. Hence, we are already evaluating the different implementation proposals paying special attention to the way in which they support the variability points presented in this paper.

Finally, our main goal is to develop a prototype development tool for the generation of a SPL of composite Web services. Although our work is still immature we plan to develop a framework for the automated or semi-automated generation of BPEL code from a given extended feature model [6]. The framework will implement a core business process in which variable parts will be generated automatically according to the feature selection. For such purpose, we will start by associating features and feature attributes

to Web services and Quality-of-Service (QoS) parameters respectively [5].

References

[1] ActiveBPEL. www.activebpel.org/.

[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer-Verlag, 2004.

[3] D. Ardagna and B. Pernici. Adaptive Service Composition in Flexible Processes. *IEEE Transactions on Software Engineering*, 33(6):369–384, 2007.

[4] A. Arkin. Business Process Modeling Language (BPML). Version 1.0, 2002. http://www.bpmi.org/.

[5] D. Benavides, A. Durán, M.A. Serrano, and C. Montes-Oca. Quality of service variability in system families based on web services. In *Simposio de Informática y Telecomunicaciones SIT 2002*, pages 205–218, Sevilla, Spain, 2002.

[6] D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Automated reasoning on feature models. *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, 3520:491–503, 2005.

[7] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan. Adaptive and Dynamic Service Composition in eFlow. In *Conference on Advanced Information Systems Engineering*, volume 1789, pages 13–31. Springer Verlag, 2000.

[8] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering. Addison–Wesley, August 2001.

[9] G. Goldszmidt and C. Osipov. Make composite business services adaptable with points of variability. choosing the right implementation. IBM, April 2007. http://www.ibm.com/developerworks/library/ar-cbspov1/.

[10] Y. Han, A. Sheth, and Chr. Bussler. A Taxonomy of Adaptive Workflow Management. In *CSCW-98 Workshop, Towards Adaptive Workflow Systems*, 1998.

[11] P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, and M. Teschke. A comprehensive approach to flexibility in workflow management systems. *SIGSOFT Softw. Eng. Notes*, 24(2):79–88, 1999.

[12] A. Ruokonen J. Jiang and T. Systa. Pattern-based Variability Management in Web Service Development. IEEE, November 2005.

[13] D. Karastoyanova and A. Buchmann. Extending Web Service Flow Models to Provide for Adaptability. In *Proceedings of the OOPSLA '04 Workshop on "Best Practices and Methodologies in Service-oriented Architectures: Paving the Way to Web-services Success"*, Vancouver, Canada, October 2004.

[14] OASIS. Web Services Business Process Execution Language Version 2.0, May 2006. http://www.oasis-open.org/.

[15] M. Reichert and P. Dadam. ADEPT flex -supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[16] S. Robak and B. Franczyk. Modeling Web Services Variability with Feature Diagrams. In *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems*, pages 120–128, London, UK, 2003. Springer-Verlag.

[17] A. Ruiz-Cortés, O. Martín-Díaz, A. Durán-Toro, and M. Toro. Improving the automatic procurement of web services using constraint programming. *Int. J. Cooperative Inf. Syst*, 14(4):439–468, 2005.

[18] E. Thomas. *Service-Oriented Architecture: A Field Guide to Integrating Xml and Web Services*. Prentice Hall, 2004.

[19] N. Yasemin Topaloglu and R. Capilla. Modeling the Variability of Web Services from a Pattern Point of View. In *ECOWS*, pages 128–138, 2004.

[20] van der Aalst, W. M. P., and S. Jablonski. Dealing with workflow change: identification of issues and solutions. *International Journal of Computer Systems Science and Engineering*, 15(5):267–276, September 2000.

[21] W3C. Web service choreography interface 1.0, August 2002. http://www.w3.org/TR/wsci/.

[22] C. Wienands. Synergies between Service-Oriented Architecture and Software Product Lines, 2006. Siemens Corporate Research. Princeton, NJ.