Service Composition at the Presentation Layer using Web Service Annotations

Tobias Nestler[1], Marius Feldmann[2], Andre Preußner[1], and Alexander Schill[2]

[1] SAP Research CEC Dresden, 01187 Dresden, Germany
{tobias.nestler, andre.preussner}@sap.com
[2] Technische Universität Dresden, Department of Computer Science, Institute for Systems Architecture, Computer Networks Group
{marius.feldmann, alexander.schill}@tu-dresden.de

Abstract. In the field of Service-Oriented Architectures the implementation of business logic and business processes is well-understood and covered by existing development approaches, but concepts for a lightweight service consumption in order to build interactive service-based applications are still in a preliminary phase. This lack of service-consumer-orientation prevents users with limited IT skills to get easy access to services and their offered functionalities. The paper presents an approach that follows the idea of integration at the presentation layer enhanced by user interface (UI) related service annotations. It describes the relationship of these ideas to already existing mashup approaches and gives an insight into how services can be composed to complex interactive applications in a visual manner without the need to write any code.

1 Motivation and Background

Service-Oriented Architectures (SOA) promise to break former monolithic applications into loosely coupled services that can be distributed across several systems. These services are composed to implement business applications and processes. Even though service composition is well-understood and covered by existing approaches for technical developers using languages such as BPEL, tools and methodologies for enabling end-user service composition have been largely ignored [1]. A promising approach for bridging this gap are mashups that focus on a user-centric and lightweight UI integration [2] by combining the philosophy of SOA and approaches of End-User Development [3]. The need for such situational applications to address individual and heterogeneous needs as well as the shift to more flexible and dynamic business environments encourage the idea of integrating mashup concepts into the enterprise. Our approach shows a way of overcoming limitations of existing mashup approaches [4] in order to build complex interactive service-based applications. Following our preliminary investigations and description of related work [5], this paper discusses the following contributions:

- We propose the usage of UI related service annotations to ease service integration and composition. This limits the effort for the development of

service-based applications to a purely model-driven, visual composition of annotated services that can even be done by end-users. Although existing approaches, such as Dynvoker [6], already cover the generation of user interfaces for single web services dynamically, no solution is available for service composition.
- We adopt the idea of integration at the presentation layer [7] to compose services by combining their presentation front-ends, rather then their application logic or data [2]. Typically, web services are integrated into the application layer of a composite application via their well-defined service interfaces. The service annotations add the missing information about the UI of a single service to lift the integration to the presentation layer.
- We propose a tool environment that allows the creation of interactive service-based applications to nonprogrammers. Most of the existing lightweight composition approaches (overview provided by [8]) support the user only in building complex data representations in form of widgets or feeds, but lack sophisticated concepts following the idea of process mashups [9].
- We propose a automatic Model-driven generation approach for the designed interactive service-based applications. The models used within the approach can be applied for representing applications for various target platforms and different sorts of application partitioning.

2 Towards Visual Service Composition

This section presents our idea of composing services in a visual manner to ease and speed up the development of interactive applications that goes beyond existing visual mashup solutions. These applications combine concepts like multi-page support, dynamic UI behavior (e.g. input suggestion functionality, client-side input validation). Therefore, we introduce the concept of UI related service annotations. These are reusable information fragments attached to the service description, which are typically not available for the application developer. They cover static UI aspects, the behavior of UI elements, and relations between services. Annotations facilitate e.g. the grouping and ordering of UI elements, completion of forms, continuous field updates, or data conversion (more examples in [5]).

UI development is usually a very time consuming task and cannot be done by the targeted end-user group. A trained application developer has to build the UI and integrate the services manually. The developer has to understand the offered interface to integrate the service in an application. This is not necessary anymore, since the integration at the presentation layer is done on a much higher level of abstraction. The user (in the role of the service composer) only works with the presentation front-end of a service instead of an abstract representation in form of a predefined service widget. Therefore, UI fragments are automatically generated for the interaction with the services and represent the interfaces for service input and expected output. The fragments can be inferred from technical details such as the data types of parameters, and be further improved by leveraging the annotations attached to the service. UI fragments consist of freely arrangeable

UI elements like input fields or buttons. A manual implementation of a service wrapper, as usually required in existing visual mashup environments, is not needed anymore.

Our approach facilitates the development of interactive single- and multi-page applications. A page acts as a container for UI elements and represents a screen in the final application. The integration of services (as described above) and the actual service composition can be done for each page separately. The service composer can define data flows between the integrated service operations on a single page (intra-page flow) and between pages (inter-page flow). These data flows can be partially derived from service dependencies defined in the annotations or modeled manually by the service composer in a visual way. Different approaches to support this specific task are currently under investigation. One solution could be the selection of a specific output field of service operation A and drawing a line to the input field of the service operation B. Another way could be that each generated UI fragment offers all of its outputs and the user can select the associated service operation via a context menu or wizard. To transport the idea of multi-page applications to nonprogrammers we use a metaphor which most of the people are familiar with - Microsoft PowerPoint. Each page (or screen) in the final application will be presented like a slide in PowerPoint. Furthermore, it is possible to define a master layout that all pages will use. To build multi-page applications, the pages can be linked to each other by specifying a navigation path.

3 End-User Centric Tool Support

Our visual composition editor which implements the concepts introduced in Sec. 2 is currently under development in the frame of the EU funded project ServFace [10]. The main focus of the composition editor is the empowerment of end-users to develop interactive applications. Multiple design decisions were made based on this requirement. The tool is designed as a rich internet application (RIA) which runs in the web browser of the user and makes an installation dispensable. The annotations facilitate the understanding and simplify the composition of web services. Finally, the visual composition concepts guide the user through the development process by providing intuitive metaphors and hide the complexity of the actual programming task. Our user-centric implementation approach involves iterative evaluations with end-users.

The composition editor is integrated into a three step methodology for the development of interactive applications as explained in [11]. The annotations are created by an IT expert and stored in an annotation model based on a formally defined Meta-model. The visual service composition tool imports in a first step the functional interface descriptions of the web services and their attached annotation models. The result is a platform-specific object model structure kept in the tool representing the complete designed application.

Figure 1 shows a mockup of the envisioned composition editor. The user can import annotated services that shall be used in the application. These services

are displayed with their operations in the `Service Operations` palette. The user can drag service operations from the palette to the composition area. The editor displays the UI fragment inferred from the operation interface and the service annotations. The user can refine the layout, delete unwanted UI widgets or add additional ones from the `Widgets` palette, and define intra-page data flows. Besides this basic mashup editor functionality our composition editor pro-

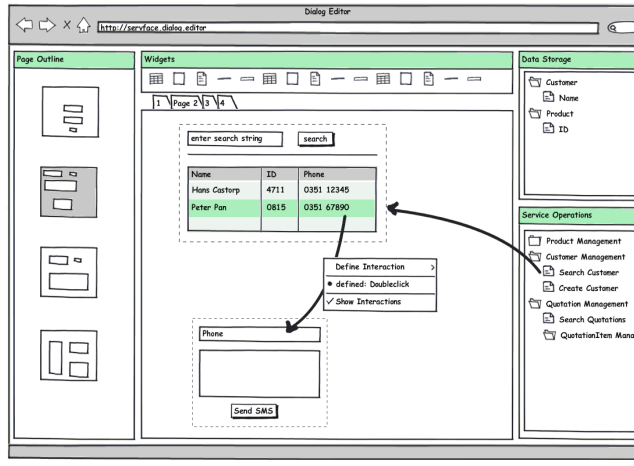

Fig. 1. Mockup of the Visual Composition Editor

vides innovative features especially designed towards the development of process mashups. The user can define inter-page data flows by dragging parameters or return values to the `Data Storage` and use them to fill UI elements at other pages. The editor supports the definition of the navigation flow of multi-page applications. This can be done either implicitly by separating the input and output of an operation to different pages, or explicitly by creating a new page and adding navigation buttons for the page transition. Concepts for an end-user friendly design of features like the inclusion of additional operations for data filtering or conversion, and the merging of UI elements to call multiple operations with one user interaction are under investigation.

After finishing the application development, the mentioned object structure is serialized to a model coined Composite Application Model (CAM). Its underlying Meta-model is reused for representing applications for various platforms. A serialized CAM is used as storage format for the composition tool and as input for generating executable application as described in the next section.

4 Generating Applications

In regards of bringing the composed interactive application to execution, the decision has been made to use a code generation mechanism. In comparison

to deploying the design-time outcome on a specific interpreter, code generation promises a higher efficiency. The chosen approach is realized in a model-driven manner. In order to bring the instance of the CAM Meta-model closer to the executable application and to resolve the annotations that are explicitly represented within the CAM to runtime information, a Model-to-Model transformation is applied in a first step. As it is the case for the CAM, the target Meta-model (named PROSAIC) can be reused for representing applications for a variety of platforms. A major challenge in realizing this approach has been to define a *reference architecture* for service-based interactive applications reflected in this Meta-model that can be used as an abstraction from concrete platforms and frameworks. Figure 2 shows an example of the control and data flow within the reference ar-

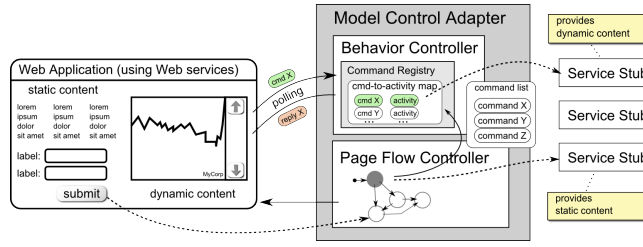

Fig. 2. UI- to Service-interaction within the reference architecture

chitecture developed for our approach. To support single page applications (e.g. RIAs) as well as multi-page applications, a differentiation between a page flow controller and a behavior controller has been introduced. Both controllers are located within the Model-Control-Adapter (MCA). Its major task is to coordinate the interaction between the user interface and the service infrastructure. The page flow controller contains a set of states and transitions between states. On state activation the state registers a set of commands within the behavior controller. These commands are mapped to a set of activities where an activity can contain actions such as the invocation of a service or assigning values to global variables. The commands are used for realizing the behavior of the user interface of the page associated with the state. For example if a widget displaying stock information should be updated in regular intervals, it triggers a command in a loop and sends it to the MCA (e.g. via Ajax functionalities). The MCA calls the service that returns the stock data and sends a reply to the UI that includes the new stock values into the widget.

This proposed reference architecture is reflected within the PROSAIC Metamodel. During several tests it has been evaluated that instances of this Metamodel can be transformed via Model-to-Model and Model-to-Code transformation to several platform and framework specific source code. Besides transforming it to Web applications (Dojo toolkit and the Spring framework) it has been proven that it can be applied for generating fat clients e.g. for mobile devices (Google Android applications).

The generation of the resulting application is done *completely automatically* by using an automatic build script for Ant that is triggered by the composition editor and that starts the M2M transformation implemented in ATL and the M2C transformation implemented by using openArchitectureWare. Furthermore this script enables the packaging and deployment of web applications.

Yet an open issue is the formal definition of the relations between the service annotations *kept explicitly* within the CAM and the PROSAIC Meta-model. This formal definition promises a starting point for a simplification of the template creation for the M2M transformation.

5 Conclusion and Future Work

The concept of presentation integration can be seen as the next major step in the integration area [7]. Our paper presented an approach to lift the service composition to the level of presentation integration via UI related service annotations. The presented visual composition concepts as well as the associated tool will empower nonprogrammers to create composite applications, which suit their requirements and individual needs. The active involvement of the actual service consumer in the integration and composition process can result in a more sufficient usage of knowledge, specific for their domain, and raise their productivity.

References

1. Ro, A., Xia, L.S.Y., Paik, H.Y., Chon, C.H.: Bill Organiser Portal: A Case Study on End-User Composition. In WISE (2008)
2. Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding UI Integration: A survey of problems, technologies, and opportunities. IEEE Internet Computing (May/June 2007)
3. Hoyer, V., Stanoevska-Slabeva, K.: The Changing Role of IT Departments in Enterprise Mashup Environments. In 2nd International Workshop on "Web APIs and Services Mashups" (Mashups08) (2008)
4. Nestler, T.: Towards a Mashup-driven End-User Programming of SOA-based Applications. In 10th International Conference on Information Integration and Web-based Applications & Services (iiWAS) (2008)
5. Nestler, T., Feldmann, M., Schill, A.: Design-Time support to create user Interfaces for service-based applications. In International Conference WWW/Internet (2008)
6. Spillner, J., Feldmann, M., Braun, I., Springer, T., Schill, A.: Ad-hoc Usage of Web Services with Dynvoker. Towards a Service-Based Internet, First European Conference, ServiceWave 2008, Madrid, Spain (2008)
7. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A Framework for Rapid Integration of Presentation Components. In WWW'07 (2007)
8. Hoyer, V., Fischer, M.: Market Overview of Enterprise Mashup Tools. In ICSOC (2008)
9. Young, O.: The Mashup Opportunity. In Forrester Research Report (May 2008)
10. ServFace Consortium: ServFace Research Project (2008) http://www.servface.eu/.
11. Feldmann, M., Janeiro, J., Nestler, T., Hübsch, G., Jugel, U., Preussner, A., Schill, A.: An Integrated Approach for Creating Service-Based Interactive Applications. In INTERACT 2009 (to appear)