CRUISe: Composition of Rich User Interface Services

Stefan Pietschmann, Martin Voigt, Andreas Rümpel, and Klaus Meißner

Technische Universität Dresden
01062 Dresden, Germany
{Stefan.Pietschmann,Martin.Voigt,Andreas.Ruempel}@inf.tu-dresden.de,
Klaus.Meissner@inf.tu-dresden.de

Abstract. As reuse and technology-independence are key issues of both software and web engineering, web services have gained momentum and are heavily used in modern web-based applications. However, they are only expedient for the business logic layer, while the Web lacks uniform models for the encapsulation and reuse of UI components. Thus, web UIs are usually hand-crafted and static, which complicates both development as well as maintenance and upgrade. We address these issues with a novel approach facilitating dynamic, service-oriented composition of user interfaces for web applications. UI parts therein are provided as reusable services and can therefore be selected, customized and exchanged dynamically with respect to a particular context.

1 Introduction and Motivation

In recent years the Internet has evolved to a stable application platform for a large number or *Software-as-a-Service* (SaaS) solutions, featuring rich UIs and interaction metaphors. This enables location- and time-independent access, but has dramatically complicated application development, especially of the UI.

In the back end, modern web applications use web services – technology-independent building blocks that facilitate reuse and exchange of business logic. There also exist numerous frameworks and libraries that allow for the composition of web-based UIs from components, e. g., Portlets, JSF and WebParts, but they all restrain to specific technologies or platforms with individual interfaces and APIs. Once a choice has been made, it is naturally irrevocable. Future UI changes and updates are costly as there is no uniform model for the technology-independent development and cross-technology integration of web UIs.

Additionally, heterogeneous device, user and usage contexts are decisive factors for a web application UI. "Adaptive Hypermedia" research has addressed this issue for years. Yet, the majority of approaches suffers from the "open corpus" problem [1] – they only work with predefined structures and preindexed or annotated documents – and usually don't support *Rich Internet Applications* (RIA). Despite all these facts, presentation integration, i. e. technology-agnostic UI composition and reuse, has not undergone much research [2]. Thus, the development and maintenance of context-aware RIAs is complicated and hence very time- and money-consuming.

M. Gaedke, M. Grossniklaus, and O. Díaz (Eds.): ICWE 2009, LNCS 5648, pp. 473–476, 2009.

To address the above-mentioned problems, we apply the service-oriented paradigm to the presentation level of web applications. In CRUISe[1], a user interface is composed from services providing reusable, configurable components. This greatly simplifies development and maintenance of web UIs. Furthermore, in contrast to present integration concepts (cf. [3,4]), the dynamic, context-aware selection, configuration and exchange of these services enables adaptation of modern web applications at run time.

2 Dynamic, Service-Based Composition of Web UIs

A growing number of applications is built from services which provide data and business logic via generic interfaces or APIs. We argue that future web-based applications can be solely based on services that provide either data, business logic or **user interfaces**. As current solutions do not support such a universal paradigm, the requirements for their deployment and hosting herald the time for a new architectural style, e. g., the "Thin Server Architecture" (TSA) [5].

Figure 1 gives an overview of our concept. Its central idea is the dynamic composition of a web application UI from distributed services to exploit the advantages of service-oriented architectures, like reusability, customizability and technology-independence, at the presentation layer. We do this by encapsulating generic, reusable web UI components as so-called *User Interface Services* that are dynamically selected, configured and integrated into a homogeneous, web-based UI. Conforming to the TSA style, the integration of all services is carried out on the client. Thus, our concept is most suitable for mashup applications aiming at the lightweight service orchestration on the presentation level.

2.1 User-Interface-as-a-Service

User Interface Services. (UIS) form an integral part of our concept. They facilitate the distributed deployment and technology-independent provision of generic, configurable UI building blocks. A trend towards such services for the presentation layer can already be witnessed, prominent examples being Google's Maps or Visualization APIs that allow for the client-side binding and integration of configurable and interactive UI components from a remote server. We generalize such techniques and propose a concept, in which the whole web application UI results from the integration and composition of UIS, or, more precisely, the UI components provided by them.

Specific APIs and technologies are hidden behind a generic UIS interface, which is used for their configuration and initialization and covers the typical functionality exposed by certain class of UI component, e. g. a *Map UIS*. Furthermore, it facilitates interoperability and run time exchangeability of UI parts. Of course, this requires the definition of universal UIS interfaces beforehand. In

[1] The CRUISe project is funded with means of the BMBF under promotional reference number 01IS08034-C.

contrast to classic integration systems, we can rely on browser technologies (especially JavaScript) as "glue code" . Due to the openness and simplicity of this approach, the majority of components and services available on the Web (based on JavaScript, JavaFX, Flash, etc.) can be provided as UIS.

UIS metadata is stored in a *UIS Registry* and used to match application requirements and context data with available UIS at run time to enable context-aware web user interfaces.

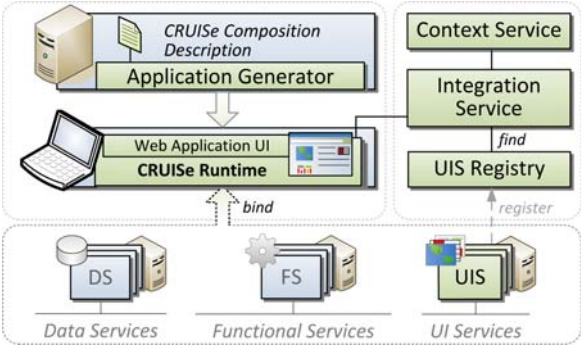

Fig. 1. Architectural overview of the CRUISe infrastructure

2.2 Dynamic, Context-Aware UI Composition

To build a UI based on UIS, a developer declaratively specifies configuration, composition and control flow between them. This *Composition Description* is processed and transformed into an executable web application by the *Application Generator*. It contains "placeholders" for UI parts provided by UIS with their associated configuration parameters, and runs within the *CRUISe Runtime*.

When the application is initialized, the Runtime passes requirements and parameters specified for each placeholder to an external *Integration Service* (CIS). This task can be carried out on the server, as well as on the client. The CIS is responsible for finding those UIS in the UIS Registry that match the given application requirements and context, ranking them by their accuracy of fit and returning the integration code for the best match. In our application domain this is typically a script which is embedded in the application and interpreted dynamically on the client. Conflicts and redundancies between UIS and underlying libraries are resolved by the Runtime, as well.

Once the integration process has finished, the Runtime controls the event and data flow between UI parts as specified in the composition description. It also serves as a homogeneous access layer that facilitates binding of backend services to the UI services. Furthermore, it monitors context data on the client (e. g., device capabilities and user interactions) and sends them to the Integration Service for later use in the discovery and ranking phase. In the end, it also carries out dynamic adaptations, like UIS reconfigurations and exchange.

3 Implementation

To verify our concepts we built an exemplary web application used to manage contacts and track their current locations. To this end, a number of UIS were developed, encapsulating UI components of different technologies, including JavaScript (Google Maps, Dojo Tables) and Flash (Flex Datagrid). In our prototype, they are dynamically integrated on the server side by an Integration Service based on Axis2. The client-side Runtime was realized as a JavaScript component using the widget framework jMaki.

This prototype exemplifies the run time composition of a web UI including data binding with the back end (contact data is provided by application services), communication between UI parts as well as the technology-independence of UIS used. It forms the basis for our current work towards the dynamic, context-aware UI adaptation. Lessons learned include, that a client-side binding of the Integration Service might be advantageous for pure TSA applications.

4 Conclusion

Our concept implies the dynamic, context-aware composition of web user interfaces based on so-called *User Interface Services*. Those provide arbitrary UI parts and are integrated and composed via a uniform configuration and communication interface. This facilitates reusability and technology-independence and offers a great simplification of the development and maintenance of web application UIs. Furthermore, it allows for context-aware rich web UIs by dynamically selecting and configuring UIS depending on the user and usage context.

At the moment, we are evaluating our prototype against other integration and mashup approaches to derive requirements for the composition description and client-side service binding. We are further working on a classification and a semantic description language for UIS to allow for semantic run time matching between web and UI Services. Future work includes the development of an authoring tool and more sophisticated, dynamic adaptation mechanisms.

References

1. Brusilovsky, P., Henze, N.: Open Corpus Adaptive Educational Hypermedia. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 671–696. Springer, Heidelberg (2007)
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 11(3), 59–66 (2007)
3. Liu, X., Hui, Y., Sun, W., Liang, H.: Towards Service Composition Based on Mashup. In: IEEE Congress on Services, pp. 332–339 (2007)
4. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A Framework for Rapid Integration of Presentation Components. In: WWW 2007: Proc. of the 16th Intl. Conf. on World Wide Web, pp. 923–932 (2007)
5. Prasad, G., Taneja, R., Todankar, V.: Life above the Service Tier (October 2007)