Conflict Resolution in Collaborative User Interface Mashups

Michael Hertel, Alexey Tschudnowsky(✉), and Martin Gaedke

Technische Universität Chemnitz, Chemnitz, Germany
{Michael.Hertel,Alexey.Tschudnowsky,
Martin.Gaedke}@informatik.tu-chemnitz.de

Abstract. User Interface (UI) Mashups propose methods and techniques, which should enable non-programmers to develop their own widget-based solutions. At the moment the process of configuring UI mashups is mainly a single-user activity. Adding support for real-time collaboration to the composition tools could make the development process more "social" and, thus, further lower the entry barrier and make users more productive. The paper describes challenges and possible solutions to enable real-time collaboration in UI mashups with particular focus on resolution of conflicts, which can occur as a result of concurrent modifications. Implementation of the proposed mechanisms is demonstrated in the context of an open-source mashup platform Apache Rave.

Keywords: User interface mashups · Real-time collaboration · Operational transformation · Conflict resolution

1 Introduction

A UI mashup is a Web application, which is developed by the composition of components called widgets, which hide complexity of underlying technologies behind graphical user interfaces [8]. Interactions between widgets usually takes place over configurable connectors or emerge in a self-organized fashion based on capabilities of widgets. By developing UI mashups non-technical users should be enabled to solve situational problems faster and more efficiently [1].

Currently the majority of UI mashup composition tools are single-user applications. Enabling a collaborative use could yield added value to the platforms due to knowledge exchange, synergy effects and faster composition process [3]. However, building Web applications with real-time collaboration capabilities is a challenging task and requires a careful design of synchronization algorithms and conflict resolution mechanisms [2].

This paper describes an approach to integrate real-time collaboration capabilities to UI mashup development platforms. We demonstrate, how Operational Transformation (OT) algorithm [7] can be applied to maintain consistency of mashup models. The approach is implemented as an extension to Apache Rave[1], a publish-subscribe-based UI mashup platform.

[1] http://rave.apache.org/

P. Cimiano et al. (Eds.): ICWE 2015, LNCS 9114, pp. 659–662, 2015.
DOI: 10.1007/978-3-319-19890-3_51

2 Conflict Resolution Using OT

Conflicts emerge from concurrent actions that do not satisfy the consistency model as defined in [5]. OT as a concurrency control mechanism is well suited and increasingly applied for preserving consistency in real-time collaboration applications. OT consists of a generic control algorithm, a data model specific to a concrete application, operations and transformation functions. The fundamental concept of OT is the modification of conflicting operations by transforming them against all previously applied concurrent operations so that they eventually satisfy the consistency model.

2.1 Data Model

In the following we consider publish-subscribe-based UI mashups with user-defined communication restrictions as proposed in [6]. The following OT-specific data model is used. A mashup is a list of so called areas, which are lists of widgets. Widgets can not be placed freely on the workspace, but rather allocated to areas or re-ordered within them. Each widget has a set of properties, a view mode and a configuration of its inter-widget communication (IWC) behavior. Properties are unordered key-value pairs. Viewmode is a method of widget rendering: “normal”, “minimized” or “maximized”. IWC configuration is a set of restrictions, which describe, which receivers and which topics are forbidden to be used for message publications.

2.2 Operations

Each defined operation has to affect the OT data model for enabling consistency maintenance with OT. The concept of site identifiers is used for situations where a global unique decision is required, i.e., the same data was modified concurrently. Every operation transmits the unique identifier of its origin instance as the last parameter *sid*. It allows a global definite decision without a dedicated server. The following OT operations are defined for the synchronization of the mashup structure:

– *AddWidget*(*id*, *area*, *pos*, *width*, *height*, *viewmode*, *properties*, *iwcsettings*, *sid*) adds a new widget to the mashup into the *area* at the position *pos*.
– *MoveWidget*(*id*, *areaOld*, *posOld*, *areaNew*, *posNew*, *sid*) moves the widget to the given area *areaNew* at the position *posNew*.
– *RemoveWidget*(*id*, *area*, *pos*, *sid*) removes a component from the mashup.
– *ChangeViewmode*(*id*, *viewmode*, *sid*) sets the display mode of the widget with the identifier *id* to the given *viewmode*.
– *ReplaceWidgetProperty*(*id*, *key*, *value*, *sid*) replaces the current value of the property with the *key* for the component identified by the *id* with *value*.
– *ChangeConnectionSetting*(*publisherid*, *subscriberid*, *topic*, *state*, *sid*) specifies the rule provided by *state* that should be applied to the connection between the defined publisher and subscriber for the given *topic*.

2.3 Transformation Functions

The structural formula for each transformation T is $O'_a = T(O_a, O_b)$, which means that the operation O_a is transformed against the operation O_b and results in the transformed operation O'_a. To meet the convergence property with OT two transformation (or convergence) properties (TP) have been identified in [4]. Dedicated transformation functions have been developed for above operations, which satisfy TP1. It is assumed that the control algorithm can preserve TP2 because this embodies a common approach [7]. To validate the preservation of TP1 $Z_1 = Z_2$ must hold for all operations O_a and O_b and any initial state Z if $Z_1 = Z \circ O_a \circ T(O_b, O_a)$ and $Z_2 = Z \circ O_b \circ T(O_a, O_b)$. The composition operator $\circ$ represents the application of an operation O to the state Z resulting in the new state Z' including the impact of O in the context of $Z' = Z \circ O$. Due to the vast amount of possible constellations arising from the combination of all operations combined with every parameter proportion the verification of TP1 was automated with the help of VOTE[2].

3 Implementation

Apache Rave has been extended to support real-time collaboration and conflict resolution. We extended its mashup sharing functionality, which enabled to share configurations among platform users. However, further modifications to the shared mashup configuration were not synchronized in "real-time" and no conflict resolution was performed. We used the ShareJS[3] OT library for real-time mashup model synchronization and consistency preservation. Figure 1 shows

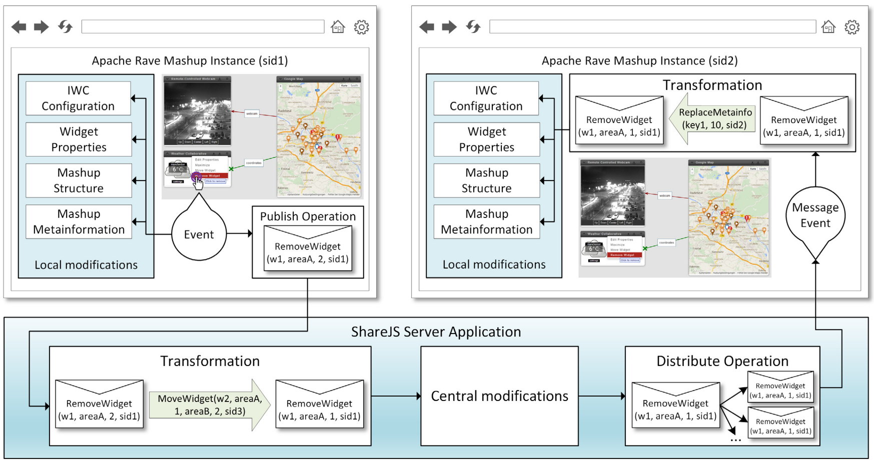
Apache Rave Mashup Instance (sid1)
IWC Configuration
Widget Properties
Mashup Structure
Mashup Metainformation
Local modifications
Event
Publish Operation
RemoveWidget (w1, areaA, 2, sid1)
Apache Rave Mashup Instance (sid2)
Transformation
RemoveWidget (w1, areaA, 1, sid1)
ReplaceMetainfo (key1, 10, sid2)
RemoveWidget (w1, areaA, 1, sid1)
Message Event
ShareJS Server Application
Transformation
RemoveWidget (w1, areaA, 2, sid1)
MoveWidget(w2, areaA, 1, areaB, 2, sid3)
RemoveWidget (w1, areaA, 1, sid1)
Central modifications
Distribute Operation
RemoveWidget (w1, areaA, 1, sid1)
RemoveWidget (w1, areaA, 1, sid1)
RemoveWidget (w1, areaA, 1, sid1)

Fig. 1. Application of OT to collaborative UI mashups

[2] http://www-sop.inria.fr/coprin/urso/logiciels/
[3] http://sharejs.org/

the general process of the OT-based synchronization. Once an event initiates a local modification, a dedicated OT operation is issued to the ShareJS server. The server performs necessary transformations against all concurrent operations received previously. After that it applies the resulting operation to the global mashup configuration maintained by the server and propagates the operation to other collaborators. The receiving clients eventually transform this operation against not yet propagated local ones and apply the result.

Demonstration: An online demo of the approach can be found at https://vsr.informatik.tu-chemnitz.de/demos/conflict-resolution-ui-mashups.

4 Conclusions and Outlook

This paper demonstrated an approach to enable conflict resolution in collaborative UI mashups. Consistency preservation mechanisms based on OT were applied to publish-subscribe-based UI mashups.

Future research will focus on application of these mechanisms to other types of UI mashups. Another subject that is not yet considered are security concerns and rights management. Finally, support of OT undo functionality could further improve usability of mashups with real-time collaboration capabilities.

References

1. Cappiello, C., Daniel, F., Matera, M., Picozzi, M., Weiss, M.: Enabling End User Development through Mashups: Requirements, Abstractions and Innovation Toolkits. In: Piccinno, A. (ed.) IS-EUD 2011. LNCS, vol. 6654, pp. 9–24. Springer, Heidelberg (2011)
2. Heinrich, M., Lehmann, F., Springer, T., Gaedke, M.: Exploiting single-user web applications for shared editing: A generic transformation approach. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 1057–1066. ACM, New York (2012)
3. Holsapple, C.W., Sims, K., Whinston, A.B.: Groupware. In: Encyclopedia of Computer Science, pp. 759–761. John Wiley and Sons Ltd., Chichester (2003)
4. Ressel, M., Nitsche-Ruhland, D., Gunzenhäuser, R.: An Integrating, Transformation-oriented Approach to Concurrency Control and Undo in Group Editors. In: Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work, CSCW 1996, pp. 288–297. ACM, New York (1996)
5. Sun, C., Chen, D.: A consistency model and supporting schemes for real-time cooperative editing systems. Australian Computer Science Communications **18**, 582–591 (1996)
6. Tschudnowsky, A., Pietschmann, S., Niederhausen, M., Hertel, M., Gaedke, M.: From Choreographed to Hybrid User Interface Mashups: A Generic Transformation Approach. In: Casteleyn, S., Rossi, G., Winckler, M. (eds.) ICWE 2014. LNCS, vol. 8541, pp. 145–162. Springer, Heidelberg (2014)
7. Xu, Y., Sun, C., Li, M.: Achieving convergence in operational transformation: conditions, mechanisms and systems. In: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW 2014, pp. 505–518. ACM, New York (2014)
8. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding mashup development. IEEE Internet Computing **12**(5), 44–52 (2008)