Towards Awareness and Control in Choreographed User Interface Mashups

Alexey Tschudnowsky[1], Stefan Pietschmann[2], Matthias Niederhausen[2]
and Martin Gaedke[1]
[1]{alexey.tschudnowsky,martin.gaedke}@informatik.tu-chemnitz.de,
[2]{stefan.pietschmann,matthias.niederhausen}@t-systems.com

ABSTRACT

Recent research in the field of user interface (UI) mashups has focused on so-called choreographed compositions, where communication between components is not pre-defined by a mashup designer, but rather emerges from the components' messaging capabilities. Though the mashup development process gets simplified, such solutions bear several problems related to awareness and control of the emerging message flow. This paper presents an approach to systematically extend choreographed mashups with visualization and tailoring facilities. A first user study demonstrates that usability of the resulting solutions increases if proposed awareness and control facilities are integrated.

Categories and Subject Descriptors

D.2.6 [**Software**]: Software Engineering

Keywords

inter-widget communication, user interface mashup, widget

1. INTRODUCTION

User interface (UI) mashups have become a popular approach for end-user development. Based on autonomous but cooperative visual components called widgets, they promise to significantly lower the barrier for Web application development [1]. A key feature of UI mashups is the inter-widget communication (IWC), which refers to the process of exchanging data between widgets with the goal of synchronizing internal states of the aggregated components. Current mashup platforms enable definition of the IWC behaviour either explicitly by a mashup designer (*orchestrated mashups*) or implicitly in a self-organized fashion based on the capabilities of the integrated components (*choreographed mashups*) [4]. The third (currently only theoretical) approach envisions a combination of the first two (*hybrid mashups*). While platforms with the orchestration strategy aim at providing maximal flexibility during mashup development, choreographed solutions focus on keeping the development process lean and fast. With regard to IWC, however, choreographed mashups pose some challenges related to awareness and control of what is happening in a mashup [2, 3]. The major awareness problem caused by implicitly defined IWC is that mashup users do not know which pairs of widgets *could* and which actually *do* communicate. The major control problem is that, being defined implicitly and not as first-level concepts, communication paths cannot be blocked or modified directly by end-users. This paper proposes a generic and systematic approach to extend choreographed mashup platforms with the missing functionality and, thus, elevate them to the hybrid ones.

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2577309.

2. HYBRID UI MASHUP ENGINEERING

Three steps should be done to extend a particular choreography-based mashup platfrom towards awareness and control functionality. First, a communication model suitable for end-user-friendly visualization and tailoring of communication paths should be defined. Second, awareness and control facilities based on the model should be introduced. Finally, architecture of the mashup platform should be extended towards support of the model and the tailoring facilities. In the following, a generic approach applicable for all choreography-based systems is described.

Generic communication model. The model is defined in a way, so that it can be shared among different choreography strategies (message passing, shared memory, publish-subscribe etc.). The assumption for the definition of the model is that – from an end-users' point of view – widgets communicate in pairs by means of unidirectional message transfers. The model M is therefore defined as a graph $G = (V, E)$ with $V = \{v | v = (name, s)\}$ being set of widgets with state $s \in \{ENABLED, ISOLATED\}$ and $E = \{e | e = (v_1, v_2, s)\}$ being set of possible communication paths between $v_1, v_2 \in V$ with state $s \in \{ENABLED, BLOCKED\}$. A widget v is allowed to communicate only if its state s is $ENABLED$. A message m from widget v_1 is allowed to be delivered to widget v_2 only if $\exists e \in E : e = (v_1, v_2, ENABLED, t)$. In order to be used in a concrete choreography mashup platform, a choreography strategy-specific algorithm for deriving the model M from widget capabilities should be defined (e.g. identifying possible communication paths E based on publish-subscribe capabilities of widgets).

Awareness and control mechanisms. States $s \in \{ENABLED, ISOLATED\}$ of widgets are visualized using different widget borders. Possible communication paths $e = (v_1, v_2, s, t) \in E$ can be represented using arrows between widgets v_1 and v_2. The arrow color indicates the state of

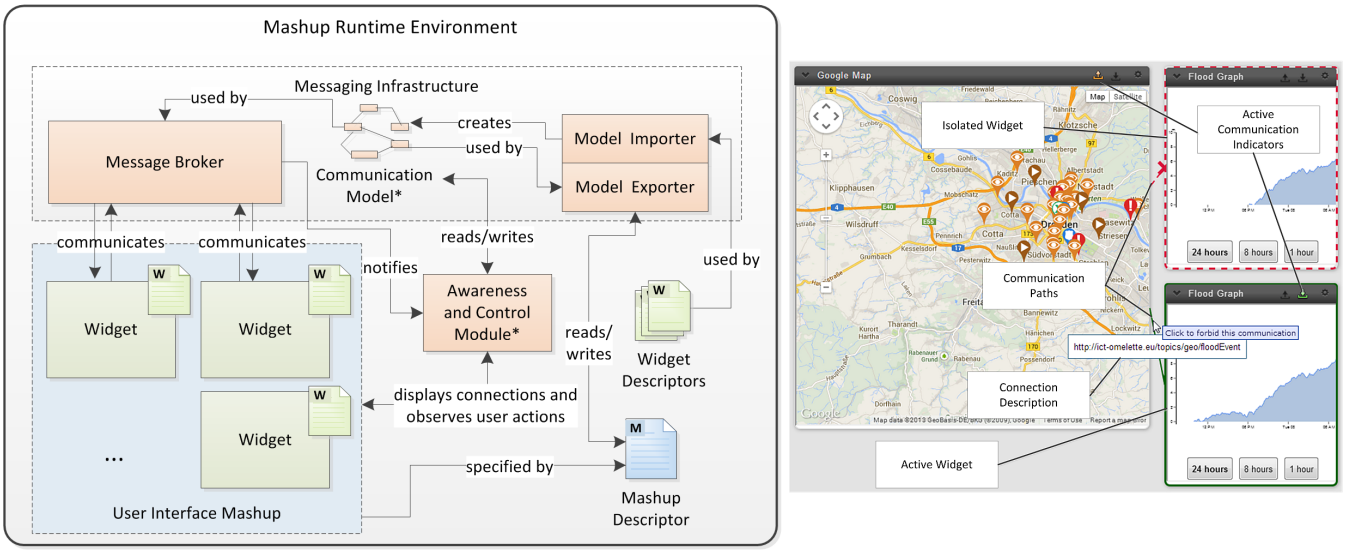

Figure 1: Reference architecture of extended choreographed mashup platforms (left) and screenshot of the proposed awareness and control mechanisms (right). * - implementation can be shared among platforms with different choreography strategies

the communication path $s \in \{ENABLED, BLOCKED\}$. Flashing icons on widget borders show, which widgets v_1 and v_2 are currently communicating. For every widget v, visualization of its state as well as in-/outgoing connections can be toggled to avoid cognitive overload in case of strong connectivity. The state can be changed from the widget context menu. For each edge $e \in E$, its state $s \in \{ENABLED, BLOCKED\}$ can be toggled by clicking on the corresponding arrow.

Reference architecture. The reference architecture (cf. Figure 1) supports both building hybrid mashups from scratch and by extension of exiting choreography platforms. *Widget and Mashup Descriptors* describe mashup configuration and are used by the *Model Importer* to derive the *Communication Model* as introduced above. The resulting model is visualized by the generic *Awareness and Control Module*. The latter also updates the *Communication Model* upon user-triggered changes on the composition canvas. The *Message Broker* routes and eventually blocks emitted messages from senders to receivers according to the particular choreography approach and the state of the *Communication Model*. The *Message Broker* notifies the *Awareness and Control Module* about active traffic so that this information can be communicated to mashup users.

The proposed approach has been implemented based on Apache Rave[1], a publish-subscribe-based UI mashup platform. A user study with 27 participants has been conducted to test effectiveness and usability of the integrated mechanisms. Users had to solve several awareness and control-related tasks. Experiments revealed that the efficiency of users didn't significantly increase if proposed facilities were activated. However, vast majority of the participants described the new facilities as helpful and easy to use. A post-questionnaire indicated possible improvements such as alternative non-ambiguous visualization techniques and more prominent placement of control mechanisms.

[1]http:/rave.apache.org

Online Demonstration. A demonstration of the approach is available at `http://vsr.cs.tu-chemnitz.de/demo/iwc-control`.

3. CONCLUSIONS AND OUTLOOK

Missing understanding of IWC dependencies and lack of control facilities can significantly impact usability and user experience within choreographed UI mashups. This paper presented an approach to extend the mashup platforms with awareness and control mechanisms. The resulting solutions differ from the current state of the art in that they both enable self-emerging ("automatic" from end-users' point of view) IWC and keep users in control of how widgets communicate at the same time. The future work will explore further awareness and control-related challenges not covered in this work, such as making users aware of *what* is being transferred between widgets and empowering them to *add* new communication paths.

Acknowledgment. This work was supported by the European Commission (project OMELETTE, contract 257635). Authors thank to Vadim Chepegin and TIE Kinetix b.v. for their contributions to the user study.

4. REFERENCES

[1] C. Cappiello, F. Daniel, M. Matera, M. Picozzi, and M. Weiss. Enabling end user development through mashups: requirements, abstractions and innovation toolkits. In *IS-EUD'11*, pages 9–24, 2011.

[2] O. Chudnovskyy, S. Pietschmann, M. Niederhausen, V. Chepegin, D. Griffiths, and M. Gaedke. Awareness and control for inter-widget communication: challenges and solutions. In *ICWE'13*, pages 114–122, 2013.

[3] E. Isaksson and M. Palmer. Usability and inter-widget communication in PLEs. In *Mupple-10*, pages 167–182, 2010.

[4] S. Wilson, F. Daniel, U. Jugel, and S. Soi. Orchestrated user interface mashups using w3c widgets. In *ComposableWeb'11*, pages 49–61, 2011.