# Supporting Semantic Interoperability in Inter-Widget-Communication-enabled User Interface Mashups

Sebastian Heil, Olexiy Chudnovskyy * and Martin Gaedke *

Chemnitz Univeristy of Technology
sebastian.heil@informatik.tu-chemnitz.de
olexiy.chudnovskyy@informatik.tu-chemnitz.de
martin.gaedke@informatik.tu-chemnitz.de

**Abstract:** Recent developments in the field of user interface mashups have acknowledged the necessity of Inter-Widget Communication. However, supporting IWC in UI mashups which are not pre-composed is not trivial. This paper illustrates the necessity of IWC for UI mashups, details disadvantages of current approaches and identifies requirements which we believe will help creating better solutions. We also present a first draft of the IWC framework we propose to address the shortcomings identified.

## 1 Introduction

Within the last years the development of web applications in the form of *user interface mashups (UI mashups)* has become increasingly important. [WDJS11] Their asynchronous and event-driven nature [SM09] and their ability to spontaneously create custom views on specific issues by composing independently developed, re-usable components, called *widgets*, have spawned a wide variety of user interface mashup applications and frameworks. [Wil12]

One of the major challenges faced by them is to enable communication between widgets, which is called *Inter-Widget Communication (IWC)*. Without IWC, each widget in a mashup works in isolation inducing redundancy as widgets are not enabled to co-operate and share common functionality thus causing them to become more extensive and also less reusable.

What is more, IWC has a strong impact on usability. [Wil12] Imagine, for instance, a mashup application concerned with travel which could feature a social travel advisory widget listing hotels highly recommended by the customer's friends, a map widget for displaying possible destinations, a translator widget to translate foreign-language hotel descriptions, a calendar widget displaying the personal calendars of the customer and a booking widget to book accommodation and flights.

*academic supervisor

Without IWC, the usage of this mashup requires several manual actions: The customer browses the list of recommended hotels and checks their locations by manually copying the address from the hotel description into the location prompt of the map widget. He then reviews the hotel description in the advisory widget. In order to translate it, he copies its text into the translator widget. After checking the room availability in the booking widget, the customer wants to cross-check the dates against his calendars. In order to do so, he has to manually leaf through the calendar widget to get to the corresponding dates. Ultimately, he books his stay and manually adds his planned vacations to his calendar by creating an event in the calendar widget.

As illustrated by this example, without interactivity between widgets, users have to perform multiple manual actions in order to synchronise widgets and simulate communication mitigating the usability of the mashup application.

## 2 Related work

Recent UI mashup projects like Geppeto[1] or Scrapplet[2] are aware of the necessity of IWC and have developed a wide range of different approaches. [SvS09] In 2011, Zuzak et al. conducted a thorough analysis of cross-context communication methods covering more than 30 systems many of which can and have been used for IWC. [ZIB11]

As of today, in 2012, there are two major competitors in the field of widget specifications: OpenSocial Gadgets and W3C Widgets. These specifications detail packaging, configuration, metadata, communication and security aspects. While specified for OpenSocial Gadgets, the W3C Widgets specifications lack a description of IWC. As more and more widget engines supporting W3C Widgets come into existence, the lack of a normative description of IWC has created a wide range of different approaches. [Goo11][CÍ1b][CÍ1a]

## 3 Current research challenges

Though many of these systems may work well in *Orchestrated UI mashups* [WDJS11] (pre-composed mashups) where experts, called mashup designers or mashup creators [SM09], build mashup applications by composing widgets with knowledge of the various data formats used and ensuring interoperability by thoroughly selecting widgets or manually defining the required transformations [SM09], there are not many solutions supporting interoperability in non-pre-composed, i.e. *Choreographed UI mashups*. [WDJS11]

However, the necessity of Inter-Widget Communication in these mashups, where end-users compose custom user interfaces by adding widgets, is evident [WDJS11], especially if the resulting solutions shall overcome being mere portal sites without any inter-activity between their components. [DST+11] The travel booking scenario previously described

[1] http://www.geppeto.fer.hr/
[2] http://www.scrapplet.com/

could refer to an orchestrated mashup as well. In order to enable IWC between arbitrary widgets employing different data formats in the messages exchanged between them, an IWC framework has to provide means of non-explicit communication while, by strictly separating semantic concepts from syntax, supporting the required transformation of data formats. Non-explicit communication refers to communication patterns which do not require explicit addressing of messages such as, for instance, broadcast messaging.

## 4 Requirements

In order to provide basic and easy-to-use communication abilities and alleviate the shortcomings in independent development of Widgets previously described, an appropriate Inter-Widget Communication framework should comply with the following criteria:

1. Independent widget development
2. Ad-hoc widget communication
3. Syntax-independent communication about semantically relevant subjects
4. Extensibility and re-use

**1 Independent widget development** A good solution shall enable widget developers to create their widgets without any a-priori knowledge of other widgets which may or may not be present in the parent context of their own widget at runtime. This suggests that widgets shall not communicate by exchanging messages explicitly directed to one or several specific recipients nor by reading from and writing to any shared memory dedicated to but one widget or a specific group of widgets. Instead, widgets shall rather communicate in a way which allows for the participation of any interested party. The idea behind is to obey to the principle of loose coupling proven successful in Web Engineering. [PWB09]

**2 Ad-hoc widget communication** A solution complying with this criterion shall provide possibilities to inform and receive notifications from other widgets about relevant events if the developer considers this necessary. The ad-hoc feature of Choreographed UI mashups necessitates the delivery of messages to all interested widgets in a continuously changing environment: As the mashup composition is not predefined, end-users may add or remove widgets at runtime. Removing a widget shall not cause interruptions in the communication flows between other widgets, any widget newly added to the running mashup shall be able to participate in the communication immediately.

**3 Syntax-independent, transparent communication about semantically relevant subjects** In order to comply with this requirement, widgets shall be able to communicate about relevant subjects regardless of specific syntax. This implies that a good solution shall provide tools to grant a strict separation of syntax and semantics by performing data format transformations if necessary.

**4 Extensibility and re-use** If the IWC-framework does not provide a transparent data format transformation for a particular widget as described above, it shall enable the widget developer to extend its functionality by supplying an implementation of the missing transformation in a standardized way. Moreover, considering criterion 3, the solution shall enable data transformations implemented for one widget to be re-used by other widgets requiring them.

## 5 Proposed IWC framework concept

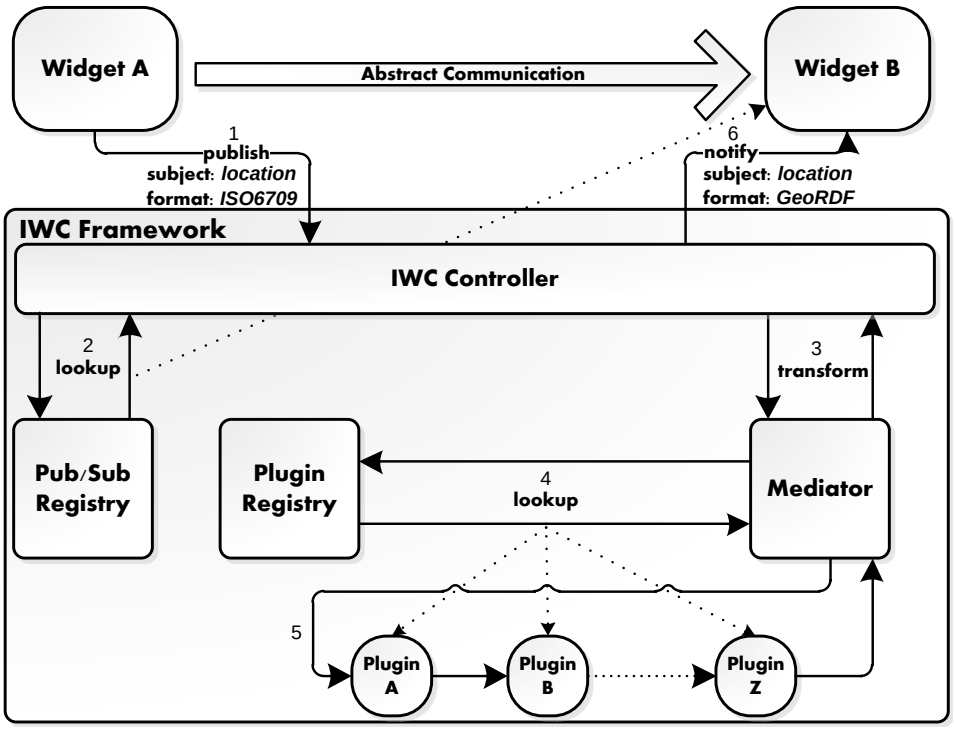


Figure 1: Proposed IWC framework architecture and publication flow

In order to solve the problems described in section 1, we propose the IWC framework architecture depicted in Figure 1. Being a "common model for choreographed communication" we join Wilson in advocating "a *publish-subscribe messaging* approach". [Wil12] It shows the publication flow triggered by Widget A publishing a message with plugins A to Z performing the transformations and subscribed Widget B receiving the message.

The framework provides a basic publish-subscribe interface. This interface is extended by additional format parameters inspired by *HTTP content negotiation*. [FGM+99] All publish-messages declare the format they are employing. Likewise, widgets subscribing to a certain subject state the formats they are able to understand, corresponding to the

"Accept header field" in an HTTP request.

Using the Mediator component, the framework transforms data from source to target format. This is done by querying the Plugin Registry for a list of plugins supporting transformation from source to target format. By performing their transformations forwarding the output of a plugin to the subsequent one in the order given by the Plugin Registry, the Mediator finally yields the data in the desired target format.

The idea of chaining plugins adds support for transformations where there is no plugin available which explicitly performs the transformation required, but which can be achieved by a sequence of transformations with the corresponding plugins available.

Unfortunately, the use of several plugins introduces a certain degree of complexity into the transformation process. To address this issue, we advocate the use of a central data format, a lingua franca as it were, for each topic so plugins merely have to transform a specific data format from and to this intermediate format consequently limiting complexity by restricting actual chain lengths to two plugins.

Our approach aims at identifying communication subjects commonly encountered in user interface mashups. This is inspired by the default intents collated by the Web Intents Task Force. [Web12] In addition to that, the *Web Intents* specification enables to register and request any service by defining the "action" parameter. Moreover, the "type" parameter which is a "string indicating the type of the data payload" [BHK12] ensures the description of the syntax employed in the intent data. We resemble this with subject and format.

So far, we have identified the following default IWC subjects:

1. Geo Location
2. Date
3. Search
4. Message (Email / SMS / Chat) Received / Sent
5. Content Created / Uploaded
6. Selection

Additionally, we aim at supporting any subject apart from those listed above in the same way the Web Intents specification does by defining a "Typed Datum Multicast" which allows to publish data on any subject and of any type.

## 6 Conclusion and future work

To conclude, we underlined the necessity of IWC for UI mashups and outlined the problems encountered in current approaches, in particular, for choreographed UI mashups. We consider the lack of independency in widget development which is introduced by recent solutions the most crucial issue. In order to alleviate this, this paper proposed several requirements for UI mashup IWC approaches and presented a first draft to increase widget development independency and semantic interoperability by applying well-known patterns such as plugin architecture, content negotiation and Web Intents to the IWC domain.

As our research on this topic still is at an early stage, we will elaborate on these requirements and adapt the architectural draft accordingly. Moreover, we intend to identify example scenarios for IWC-enabled choreographed UI mashups and derive further default subjects from them. This will also affect the decision on how to perform the transformations. The next step will be to implement the IWC framework, test the implementation in these scenarios and validate the framework against our requirements. Moreover, the process of implementing will likely yield interesting insights which we would like to report on by composing them into a set of best practices.

## References


[BHK12] Greg Billock, James Hawkins, and Paul Kinlan. Web Intents. *W3C Editor's Draft*, 2012.

[CÍ1a] Marcos Cáceres. Widget Interface. *W3C Candidate Recommendation*, 2011.

[CÍ1b] Marcos Cáceres. Widget Packaging and XML Configuration. *W3C Recommendation*, 2011.

[DST+11] Florian Daniel, Stefano Soi, Stefano Tranquillini, Fabio Casati, Chang Heng, and Li Yan. Distributed Orchestration of User Interfaces. *Information Systems*, 37(6):539–556, 2011.

[FGM+99] Roy Fielding, J Getty, J Mogul, H Frystyk, L Masinter, P Leach, and Tim Berners-Lee. RFC 2616: Hypertext Transfer Protocol – HTTP/1.1, 1999.

[Goo11] Google Inc. OpenSocial Core Gadget Specification 2.0.1, 2011.

[PWB09] Cesare Pautasso, Erik Wilde, and U C Berkeley. Why is the Web Loosely Coupled ? A Multi-Faceted Metric for Service Design. In *Proceedings of the 18th International World Wide Web Conference (WWW2009)*, Madrid, Spain, 2009.

[SM09] Michele Stecca and Massimo Maresca. An execution platform for event driven mashups. *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services - iiWAS '09*, page 33, 2009.

[SvS09] Srbljic Siniša, Dejan Škvorc, and Daniel Skrobo. Widget-Oriented Consumer Programming. *AUTOMATIKA: Journal for Control, Measurement, Electronics, Computing and Communications*, 50:252–264, 2009.

[WDJS11] Scott Wilson, Florian Daniel, Uwe Jugel, and Stefano Soi. Orchestrated user interface mashups using w3c widgets. In *Proceedings of ComposableWeb*. Springer, 2011.

[Web12] Web Intents Task Force. webintents.org, 2012.

[Wil12] Scott Wilson. Design challenges for user-interface mashups: user control and usability in inter-widget communications, 2012.

[ZIB11] Ivan Zuzak, Marko Ivankovic, and Ivan Budiselic. A Classification Framework for Web Browser Cross-Context Communication. *Arxiv preprint arXiv:1108.4770*, 2011.