# Developing Domain-Specific Mashup Tools for End Users

Florian Daniel, Muhammad Imran, Felix Kling, Stefano Soi,
Fabio Casati and Maurizio Marchese
University of Trento, Via Sommarive 5, 38123, Trento, Italy
lastname@disi.unitn.it

## ABSTRACT

The recent emergence of mashup tools has refueled research on *end user development*, i.e., on enabling end users without programming skills to compose own applications. Yet, similar to what happened with analogous promises in web service composition and business process management, research has mostly focused on technology and, as a consequence, has failed its objective. Plain technology (e.g., SOAP/WSDL web services) or simple modeling languages (e.g., Yahoo! Pipes) don't convey enough meaning to non-programmers.

We propose a *domain-specific* approach to mashups that "speaks the language of the user", i.e., that is aware of the terminology, concepts, rules, and conventions (the domain) the user is comfortable with. We show what developing a domain-specific mashup tool means, which role the mashup meta-model and the domain model play and how these can be merged into a domain-specific mashup meta-model. We apply the approach implementing a mashup tool for the research evaluation domain. Our user study confirms that domain-specific mashup tools indeed lower the entry barrier to mashup development.

## Categories and Subject Descriptors

H.m [**Information Systems**]: Miscellaneous; D.1 [**Software**]: Programming Techniques

## Keywords

Domain-Specific Mashups, End-User Development

## 1. INTRODUCTION

***Mashups*** are typically simple web applications that, instead of being coded from scratch, are developed by integrating and reusing available data, functionalities, or pieces of user interfaces accessible over the Web. ***Mashup tools*** aim at enabling non-programmers (web users) to develop own applications. Yet, similar to what happened in service composition, the mashup platforms developed so far either expose too much functionality and too many technicalities so that they are powerful and flexible but suitable only for programmers, or they only allow compositions that are so simple to be of little use for most practical applications.

We believe that ***the heart of the problem*** is that it is impractical to design tools that are *generic enough* to cover a wide range of application domains, *powerful enough* to enable the specification of non-trivial logic, and *simple enough* to be actually accessible to non-programmers. At some point, we need to give up something. In our view, this something is generality. Giving up generality in practice means narrowing the focus of a design tool to a well-defined *domain* and tailoring the tool's development paradigm, models, language, and components to the specific needs of that domain only, therefore creating a ***domain-specific mashup tool***.

Domain-specific development instruments are traditionally the object of domain-specific modeling (DSM) [2] and domain-specific languages (DSLs) [4], yet they typically target developers, with only few exceptions. Costabile et al. [1], for instance, successfully implemented a DSM-based tool enabling end user development in the context of a specific company and technological framework. Given the huge technological diversity on the Web, however, mashup tools are still too complex, and non-programmers are not able to manipulate the provided compositional elements [5] (e.g., Yahoo! Pipes comes with web services, RSS feeds, regular expressions, and the like). Web service composition approaches like BPEL are completely out of reach.

In this poster, we describe a ***methodology*** for the development of domain-specific mashup tools, defining the necessary concepts and design artifacts. The methodology targets expert developers, who implement mashup tools. We show how we used the methodology to implement a ***mashup platform*** for research evaluation. The platform targets domain experts (e.g., scientists). Finally, we report on our ***user study***, which confirms the viability of the developed platform and of the respective development methodology.

## 2. METHODOLOGY

Reverse-engineering our experience with the implementation of the mashup platform described in the next section, developing a domain-specific mashup platform requires:

1. Definition of a ***domain concept model*** (CM) to express domain data and relationships, which allow the mashup platform to understand what kind of *data objects* it must support. This is different from generic mashup platforms, which provide support for generic data formats, not specific objects.

2. Identification of a generic ***mashup meta-model*** (MM) that suits the composition needs of the domain. A variety of different mashup approaches, i.e., meta-models, have emerged over the last years, (e.g., data, user interface and process mashups).

Copyright is held by the author/owner(s).
*WWW 2012 Companion,* April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

3. Definition of a ***domain-specific mashup meta-model***. Given a generic MM, the next step is understanding how to inject the domain into it so that all features of the domain can be communicated to the developer. We approach this by specifying and developing:

   A ***domain process model*** (PM) that expresses classes of domain activities and, possibly, ready processes (which we can map to reusable components of the platform). Domain activities and processes represent the dynamic aspect of the domain. They operate on and manipulate the domain concepts.

   A ***domain syntax*** that provides each concept in the domain-specific mashup meta-model (the union of MM and PM) with an own symbol that conveys the respective functionality to domain experts.

   A set of ***instances of domain-specific components***. This is the step in which the reusable domain-knowledge is encoded in the form of components in order to enable domain experts to mash it up into new applications.

4. ***Implementation*** of the domain-specific mashup tool (DMT) whose expressive power is that of the domain-specific mashup meta-model.

## 3. THE RESEVAL MASH TOOL

ResEval Mash [3] is a mashup platform (a DMT) for research evaluation, i.e., for the assessment of the productivity or quality of researchers, teams, institutions, journals, and the like. The platform is specifically tailored to the needs of sourcing data about scientific publications and researchers from the Web, aggregating them, computing metrics (also complex and ad-hoc ones), and visualizing them. ResEval Mash is a hosted mashup platform with a client-side editor and runtime engine running inside a common web browser.

Developing ResEval Mash required addressing the specific requirements coming from the research evaluation domain. The first step to characterize this domain was the definition of a suitable *domain concept model* (CM). Research evaluation deals with publications, researchers, conferences, journals, metrics (e.g., h-index or citation counts), and so on. We encoded a respective CM in a suitable XML schema.

Next, composing the above concepts into a new, complex evaluation logic in essence means processing data (next to visualizing the output graphically). As *generic mashup meta-model* we therefore chose a data flow based meta-model, which focuses the attention of the user to the passing of data (e.g., publications) from one computing step to another.

Turning this meta-model into a *domain-specific mashup meta-model* then required selecting a set of abstract domain activities, i.e., defining the *domain process model*. Here we have identified data source extraction activities (e.g., for Google Scholar or Scopus), metric computation activities (e.g., h-index, g-index), aggregation and filtering activities, and finally visualization activities (e.g., UI widgets). After that, we implemented a set of *instances of domain-specific components* for the identified domain activities. For instance, we developed a Google Scholar and a Microsoft Academic data component, a h-index component, a citation count component, a filter component, a bar chart component for the visualization of metrics, and others. We then equipped these components with a *domain syntax* that clearly distinguished between data sources, metrics, filters and visualization components. Figure 1 shows an example mashup in ResEval Mash.

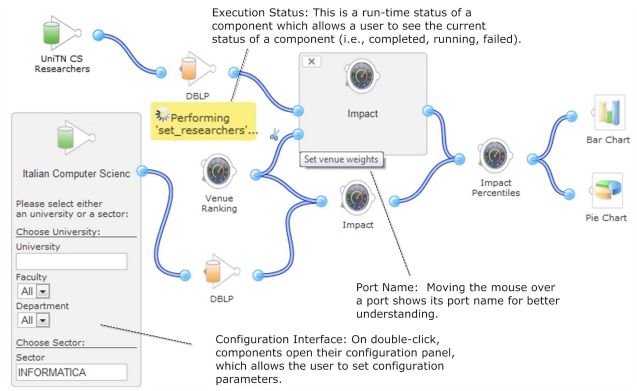


**Figure 1: ResEval Mash in action**

## 4. EVALUATION AND LESSON LEARNED

We have performed a *user study* of ResEval Mash with 10 users (5 with and 5 without IT skills and with different domain expertise). Participants were asked to fill in a questionnaire about their computing and research evaluation skills before the test, to watch a video tutorial about ResEval Mash, and to use the tool. This interaction was filmed, as was the interview that followed task completion. The results of the user study show that end users indeed feel comfortable in a mashup environment that resembles the domain they are acquainted with. The intuitiveness of the used components, which represent well-known domain concepts and actions, prevails over the lack of composition knowledge the users (the domain experts) may have and help them to acquire the necessary composition skills step by step by simply "playing" with ResEval Mash.

With ResEval Mash, we constrain the mashup language to a single domain and the mashup components to the domain's concept model. While this might be an additional burden on the component developer, it allows us to shield the user from one of the most complex aspects of mashups, i.e., data mappings. Users only need to think about the data flow, then the components know themselves which data to use. This is a very simple, but powerful simplification.

## 5. REFERENCES

[1] M. F. Costabile, D. Fogli, G. Fresta, P. Mussio, and A. Piccinno. Software environments for end-user development and tailoring. *PsychNology Journal*, pages 99–122, 2004.

[2] R. France and B. Rumpe. Domain specific modeling. *Software and Systems Modeling*, 4:1–3, 2005.

[3] M. Imran, F. Kling, S. Soi, F. Daniel, F. Casati, and M. Marchese. ResEval Mash: Advanced Research Evaluation for Domain Experts. In *WWW'12*, 2012.

[4] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, 2005.

[5] A. Namoun, T. Nestler, and A. De Angeli. Service Composition for Non Programmers: Prospects, Problems, and Design Recommendations. In *Proceedings of ECOWS*, pages 123–130. IEEE, 2010.