ResEval Mash: A Mashup Tool for Advanced Research Evaluation

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

ABSTRACT

In this demonstration, we present ResEval Mash, a mashup platform for *research evaluation*, i.e., for the assessment of the productivity or quality of researchers, teams, institutions, journals, and the like – a topic most of us are acquainted with. The platform is specifically tailored to the need 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, both running inside a common web browser. It supports the processing of also *large amounts of data*, a feature that is achieved via the sensible distribution of the respective computation steps over client and server. Our preliminary user study shows that ResEval Mash indeed has the power to enable *domain experts* to develop own mashups (research evaluation metrics); other mashup platforms rather support skilled developers. The reason for this success is ResEval Mash's domain-specificity.

Categories and Subject Descriptors

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

General Terms

Design, Experimentation

1. INTRODUCTION

Mashups are typically simple web applications (most of the times consisting of just one single page) that, rather than being coded from scratch, are developed by integrating and reusing available data, functionalities, or pieces of user interfaces accessible over the Web. ***Mashup tools***, i.e., online development and runtime environments for mashups, typically aim to enable also non-programmers to develop own applications. Yet, similar to what happened in web 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 only allow compositions that are so simple to be of little use for most practical applications. For example, mashup tools typically come with SOAP services, RSS feeds, UI widgets, and the like. Non-programmers simply do not know how to use these and what to do with them.

WWW2012, April 16-20, 2012, Lyon, France.

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.

Domain-specific development instruments are traditionally the object of domain-specific modeling (DSM) [4] and domain-specific languages (DSLs) [5], 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 [6] (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 paper we present ***ResEval Mash*** i.e., a domain-specific mashup tool, which we specifically developed for the domain of research evaluation. The development of complex evaluation metrics that go beyond the commonly adopted h-index is usually still a complex and manual task that is not adequately supported by software instruments. In fact, computing an own metric might mean extracting, combining, and processing data from multiple sources, implementing new algorithms, visually representing the results, and similar. The *Web of Science* (`http://scientific.thomson.com/products/wos/`) by Thomson Scientific, *Publish or Perish* (`http://www.harzing.com/pop.htm`), or *Google Scholar* do provide some basic metrics, such as the h-index or g-index; but they are not able to satisfy more complex evaluation logics. The goal of ResEval Mash is therefore to enable *domain experts* (typically non-programmers) to define and run their own evaluation metrics in a simple and rapid way, leveraging on suitably tailored *mashup technology*.

For explanation purpose, throughout this paper we will use a specific scenario, which we introduce in the next section. However, the tool is not restricted to one scenario only and rather aims at allowing users to develop their own mashups addressing a variety of different scenarios in the research evaluation domain.

2. EXAMPLE SCENARIO

As an example of a concrete research evaluation task, let's look at the procedure used by the central administration of the University of Trento (UniTN) to assess the productivity of the researchers of its departments. The evaluation is used to allocate resources and funding for the university departments. In essence, the algorithm compares the scientific production of each researcher in a given department of UniTN with the average production of the researchers belonging to similar departments (i.e., departments in the same disciplinary sector) in all Italian universities. The comparison uses a procedure based on a simple bibliometric indicator, i.e., a weighted publication count metric:

1. A list of all researchers working in Italian universities is retrieved, and a reference sample with similar statistical features of the evaluated department is compiled.
2. Publications for each researcher of the selected department and for all Italian researchers in the selected sample are extracted from an agreed-on data source (e.g., Microsoft Academic, Scopus, DBLP, or similar).
3. The obtained publications are weighted using a venue classification provided by a UniTN committee, which is split into three quality categories based on the ISI Journal Impact Factor. For each researcher, a weighted publication count is obtained with a simple weighted sum of his/her publications.
4. A statistical distribution – more specifically, a negative binomial distribution – of the weighted publication count metrics is then computed for the national researcher reference sample.
5. Each researcher of the selected department is then ranked based on his/her individual weighted publication count, estimating his/her percentile in the derived statistical distribution, i.e., the percentage of the researchers in the same disciplinary sector that have the same or lower values for the specific metric.

The percentile for each researcher in the selected department is used as the parameter that estimates the publishing profile of that researcher and is used for the comparison with other researchers in the same department. As one can notice, plenty of effort is required to compute the performance of each researcher, which is currently mainly done manually.

Many factors can significantly impact on the results of this evaluation process (e.g., the data sources or the sampling criteria), and people (e.g., administrative employees and researchers) want to check the results of different possible metrics. If manually done, this would cost too much time and human resources. The task, however, has all the characteristics of a *mashup*, especially if the mashup logic comes from the users.

3. THE RESEVAL MASH TOOL

The above scenario, the domain, and our target user group, i.e., domain experts, pose a set of peculiar requirements to the development of the ResEval Mash tool. In the following we summarize the design principles that underlie ResEval Mash and where the domain specifics come into play. Then, we provide some insight into its internals and implementation.

3.1 Principles and Requirements

ResEval Mash is based on the following principles and requirements:

1. ***Intuitive graphical user interface.*** Enabling domain experts to develop their own research evaluation metrics, i.e., mashups, requires an intuitive and easy-to-use user interface (UI) based on the concepts and terminology the target domain expert is acquainted with. Research evaluation, for instance, speaks about metrics, researchers, publications, etc.
2. ***Intuitive modeling constructs.*** Next to the look and feel of the platform, it is important that the functionalities provided through the platform (i.e., the building blocks in the composition design environment) resemble the common practice of the domain. For instance, we need to be able to compute metrics, to group people and publications, and so on.
3. ***No data mappings.*** Our experience with prior mashup platforms, i.e., mashArt [2] and MarcoFlow [3], has shown that data mappings are one of the least intuitive tasks in composition environments and that non-programmers are typically not able to correctly specify them. We therefore aim to develop a mashup platform that is able to work without data mappings.
4. ***Runtime transparency.*** In order to convey to the user what is going on during the execution of a mashup especially when it takes several seconds, we provide transparency into the state of a running mashup. We identify two key points where transparency is important in the mashup model: components and processing state. At each instant of time during the execution, the runtime environment should allow the user to inspect the data processed and produced by each component. In addition, to convey the processing state of each component and thus the mashup model the environment should graphically show the state.
5. ***Data-intensive processes.*** Although apparently simple, the chosen domain is peculiar in that it may require the processing of large amounts of data (e.g., we need to extract all the publications of the Italian researchers' sample for a given scientific sector). While runtime transparency is important at the client side, data processing should however be kept at the server side. In fact, loading large amounts of data from remote services and processing them in the browser at the client side is unfeasible, due to bandwidth, resource, and time restrictions.

3.2 The Domain

Some of the above requirements require ResEval Mash to specifically take into account the characteristics of the research evaluation *domain*. Doing so produces a platform that is fundamentally different from generic mashup platforms, such as Yahoo! Pipes (`http://pipes.yahoo.com/pipes/`). We achieve domain-specificity as follows:

To provide users with a mashup environment that has an *intuitive graphical UI* we design first a ***domain syntax***, which provides each object in the composition environment with a visual metaphor that the domain expert is acquainted

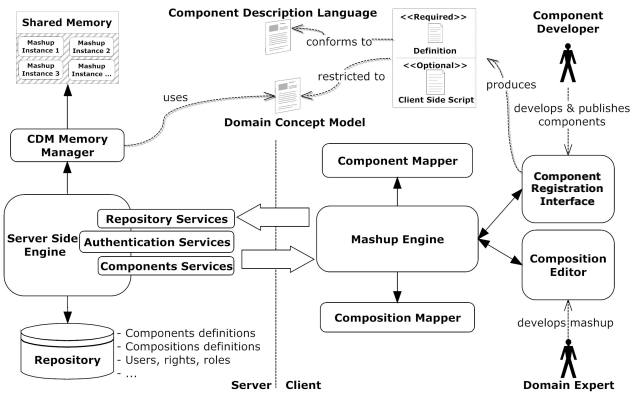

Figure 1: The ResEval Mash architecture

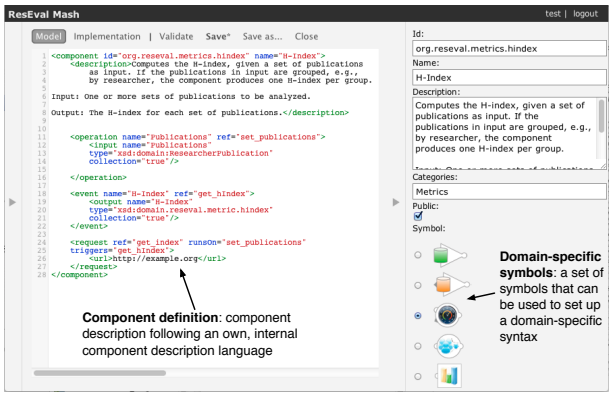

Figure 2: Component registration interface

with and that visually convey the respective functionalities. For instance, ResEval Mash uses a gauge for metrics and the icons that resemble the chart types of graphical output components.

The core of the platform are the *functionalities* exposed to the domain expert in the form of *modeling constructs*. These must address the specific domain needs and cover as many as possible mashup scenarios inside the chosen domain. To design these constructs, a thorough analysis of the domain is needed, so as to produce a so-called ***domain process model***, which specifies the classes of domain activities and, possibly, ready processes that are needed (e.g., data sources and metrics). Next, a set of ***instances of domain activities*** (e.g., an h-index algorithm) must be implemented, which can be turned into concrete mashup components.

Finally, in order to relieve users from the definition of data mappings, ResEval Mash is based on an explicit ***domain concept model***, which expresses all domain concepts and their relationships. If all instances of domain activities understand this domain concept model and produce and consume data according to it, we can omit data mappings from the composition environment in that the respective components simply know how to interpret inputs.

3.3 Architecture and Implementation

Figure 1 shows the architecture of ResEval Mash, which is divided into two parts, i.e., client side and server side.

The ***mashup engine*** is the most important part of the platform. It is developed for client-side processing, that is, we control data processing on the server from the client. The engine is primarily responsible for running a mashup composition, triggering the component's actions, and managing the communication between client and server. The engine provides for data flow processing. The ***composition editor*** provides the mashup design canvas to the user. It shows a components list, from which users can drag and drop components onto the canvas in order to connect them. The editor implements the domain syntax. From the editor, it is also possible to launch the execution of a composition through a run button and hand the mashup over to the *mashup engine* for execution. The composition editor and its various parts are shown in Figure 3. ***Component and composition mappers*** parse component and composition descriptors to represent them in the *composition editor* at design time and to bind them in the *engine* at run time.

The platform also comes with a ***component registration interface*** for developers, which aids them in the setup and addition of new components to the platform. As shown in Figure 2, the interface allows the developer to define components starting from ready templates. In order to develop a component, the developer has to provide two artifacts: (i) a *component definition* and (ii) a *component implementation*. The implementation consists either of JavaScript code, for client-side components, or a web service, for server-side components, which is linked by the component definition.

The whole client-side part of ResEval Mash is developed in JavaScript, using the *Google Closure* and *WireIt* libraries.

On the server side, we have a set of RESTful web services, i.e., the ***repository services***, ***authentication services***, and ***components services***. Repository services enable CRUD operations for components and compositions. Authentication services are used for user authentication and authorization. Components services manage and allow the invocation of those components whose business logic is implemented as a web service. These web services, together with the client-side components, implement the *instances of domain activities* inside the *domain process model*. The ***common data model (CDM)*** implements the *domain concept model* and supports the checking of data types in the system. The CDM is a *shared memory* that provides a space for each mashup instance. All data processing services read and write to this shared memory. In order to configure the CDM, the *CDM memory manger* generates corresponding Java classes (e.g., in our case POJO classes, annotated with JAXB annotations) from an XSD that encodes the domain concept model. The ***server-side engine*** is responsible for managing all the modules that are at the server side, e.g., the CDM memory manager, the repository, and so on. The server-side engine is the place where requests coming from the client side are fulfilled.

In Figure 3, we illustrate the final mashup model of our research evaluation scenario as developed with ResEval Mash. The model starts with two parallel flows: one computing the weighted publication number (the "impact" metric in the specific scenario) for all Italian researchers in the Computer Science disciplinary sector. The other computing the same "impact" metric for the researchers belonging to UniTN Computer Science department. The first branch defines the distribution of the Italian researchers for the Computer Science disciplinary sector, while the second branch is used to compute the impact value of UniTN's researchers and to determine their individual percentiles, which are finally vi-

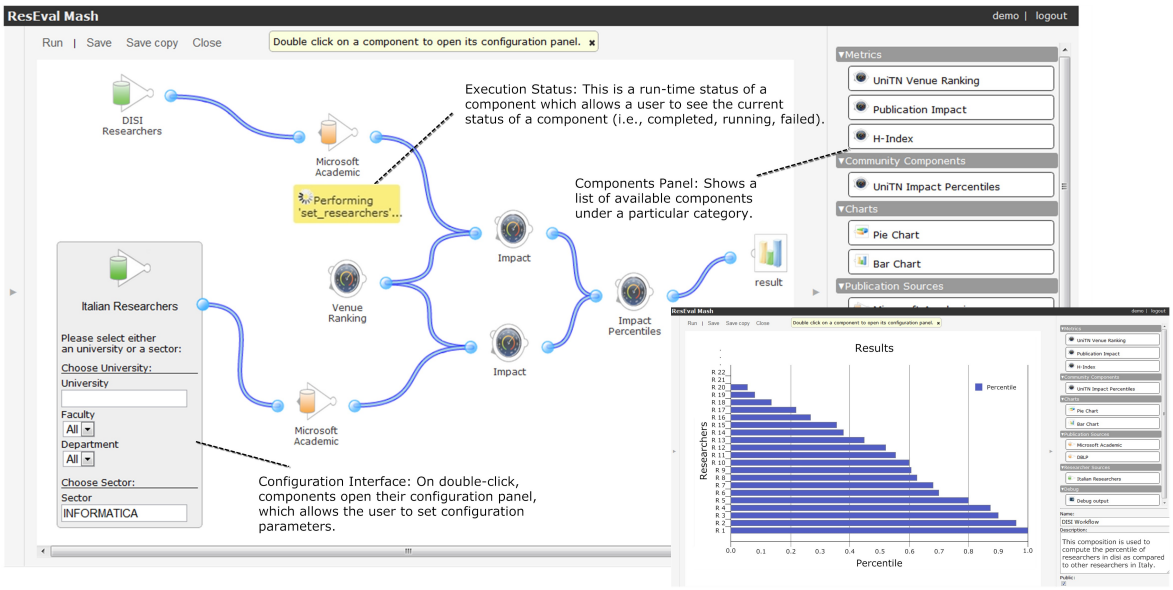

Figure 3: ResEval Mash in action: screen shots of the modeling canvas and the final mashup output

sualized in a bar chart (for obvious reasons, we anonymized the respective data).

4. DEMONSTRATION STORYBOARD

The live demo will be presented starting from an introduction of the reference domain (i.e., research evaluation) and the motivation behind the implementation of ResEval Mash. A guided walk-through the tool will be presented to introduce the modeling paradigm of the tool. We will compose a few example scenarios and describe the various features provided by the tool. After this introduction, we will ask people from the audience to try the tool and to develop their own simple research evaluation mashups. A short introduction of the component editor will be given to show the component creation and deployment process. Finally, the platform architecture will be presented to highlight the various appealing aspects of the tool.

A screencast and a continuously updated prototype of ResEval Mash is available at `http://open.reseval.org/`.

5. EVALUATION AND LESSONS LEARNED

ResEval Mash stems from the actual needs in our university and from our own needs in term of research evaluation. It also results from the observation that in general composition technologies failed to a large extent to strike the right balance between *ease of use* and *expressive power*. They define seemingly useful abstractions and tools, but in the end domain experts are still not able to understand and use them. What we have pursued in the development of ResEval Mash, in essence, is to constrain the language to the domain and to provide a domain-specific notation so that it becomes easier to use and in particular does not require users to deal with one of the most complex aspects of process modeling (at least for end users), that of data mappings.

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, while being filmed.

The comparison between the two groups of users highlighted good performance independently of participants' computing skills. The request for higher training emerging from a few less expert users appeared to be rather linked to a weaker domain knowledge than to their computing capabilities. A major finding is related to the ease with which our sample understood that components had to be linked together so that information could flow between different services. This is a well-acknowledged problem evinced in several user studies of EUD tools (e.g., [6]), which did not occur at all in the current study. To a large extent, this result can be achieved thanks to the fact that ResEval Mash relieves users from the definition of data mappings.

Acknowledgment: This work was supported by EU project OMELETTE (contract no. 257635).

6. REFERENCES

[1] M. F. Costabile, D. Fogli, G. Fresta, P. Mussio, and A. Piccinno. Software environments for end-user development and tailoring. *PsychNology Journal*, 2(1):99–122, 2004.

[2] F. Daniel, F. Casati, B. Benatallah, and M.-C. Shan. Hosted Universal Composition: Models, Languages and Infrastructure in mashArt. In *ER'09*, pages 428–443.

[3] F. Daniel, S. Soi, S. Tranquillini, F. Casati, C. Heng, and L. Yan. From People to Services to UI: Distributed Orchestration of User Interfaces. In *BPM'10*, pages 310–326.

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

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

[6] 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.