ResEval Mash: A Mashup Tool that Speaks the Language of the User

Muhammad Imran
Department of Information Engineering and Computer Science. University of Trento, Via Sommarive 5, 38123, Trento, Italy
imran@disi.unitn.it

Maurizio Marchese
Department of Information Engineering and Computer Science. University of Trento, Via Sommarive 5, 38123, Trento, Italy
marchese@disi.unitn.it

Florian Daniel
Department of Information Engineering and Computer Science. University of Trento, Via Sommarive 5, 38123, Trento, Italy
daniel@disi.unitn.it

Fabio Casati
Department of Information Engineering and Computer Science. University of Trento, Via Sommarive 5, 38123, Trento, Italy
casati@disi.unitn.it

CHI'12, May 5-10, 2012, Austin, Texas, USA.
ACM 978-1-4503-1016-1/12/05.

Abstract
End-user development (i.e., enabling end-users without programming skills to build their own applications) is undergoing a revolution, as mashups are widely considered to be the most appealing development tool for the situational, short-span applications. Plain technology (e.g., SOAP/WSDL web services) or simple modeling languages (e.g., Yahoo! Pipes) don't convey enough meaning to non-programmers. In this paper, 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 exemplify the approach by implementing a mashup tool for a specific domain (research evaluation) and describe the respective user study. The results of a first user study confirm that domain-specific mashup tools indeed lower the entry barrier to mashup development.

Keywords
Mashups, End-User development, Research Evaluation Domain-Specific mashups

ACM Classification Keywords
H.5.m [Information Interfaces and Presentation]: Miscellaneous;

General Terms

Design, Experimentation

Introduction

Mashups are web applications that typically used to combine data or functionality from other web sources to provide a new service. Mashup tools, i.e., online development and runtime environments for mashups, typically aim to enable also non-programmers to develop their own applications. Along with the prevalence of mashups, it is also considered to be a focal point of major technologies like web 2.0, situational applications, and end-user development. 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.

Yet, being amenable to non-programmers is increasingly important as the opportunity given by the wider and wider range of available online applications and the increased flexibility that is required in both businesses and personal life management raise the need for situational (one-use or short-lifespan) applications that cannot be developed or maintained with the traditional requirement elicitation and software development processes.

Problem Statement and Approach

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, since reducing expressive power would mean supporting only the development of toy applications, which is useless, while simplicity is our major aim. 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.

In this paper, we therefore champion the notion of domain-specific mashup tools and describe what they are composed of, how they can be developed, how they can be extended for the specificity of any particular application context, and how they can be used by non-programmers to develop complex mashup logics within the boundaries of one domain.

The Domain

A domain is a delimited sphere of concepts and processes; domain concepts consist of data and relationships; domain processes operate on domain concepts and are either atomic (activities) or composite (processes integrating multiple activities). The domain defines the "universe" in the context of which we can define domain-specific mashups. It defines the information that is processed by the mashup, both conceptually and in terms of concrete data types (e.g., XML schemas). It defines the classes of components that can be part of the process and how they can be combined, as well as a notation that carries meaning in

the domain (such as specific graphical symbols for components of different classes).

In this paper, 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.

Principles and Requirements
Turning the previous consideration into practice, the development of ResEval Mash (i.e., a domain-specific tool) will be driven by the following key principles:

1. ***Intuitive graphical user interface.*** Enabling end-users (i.e., domain experts) to develop 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 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.

The ResEval Mash Tool

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[1]. 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 with and that visually convey the

[1] http://pipes.yahoo.com/pipes/

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

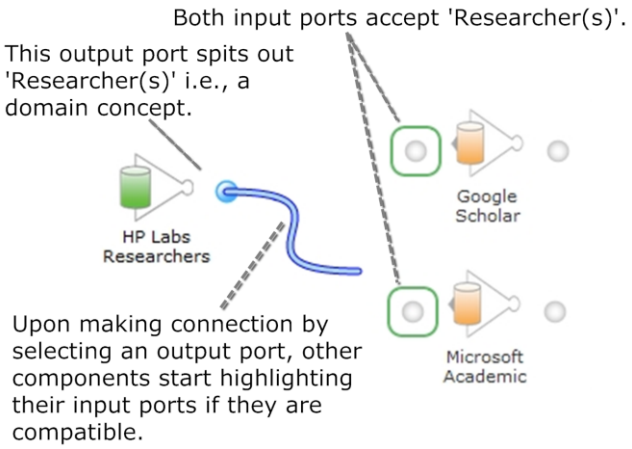

Figure 1: Connecting components in ResEval Mash

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 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.

In order to relieve users from the definition of complex data mappings, ResEval Mash is based on an explicit domain concept model, which expresses all domain concepts and their relationships. As shown in Figure 1, 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.

The research evaluation domain comes with a requirement of dealing with large amounts of data. For instance, in order to compute the h-index of all the researchers in a country or even in a university for a specific field requires to fetch a large dataset from a given source and to process it. For this purpose, ResEval Mash is based on an architecture which sensibly distributes big computational steps over client and server sides as shown in Figure 2.

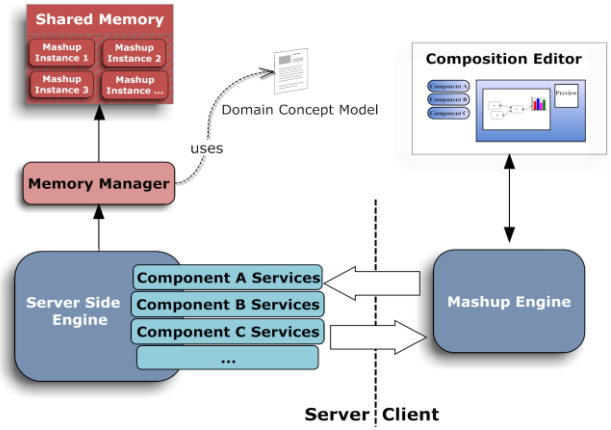

Figure 2: Architecture for handling large amount of data

A good practice is to implement a server side component (i.e., a component binded with a web

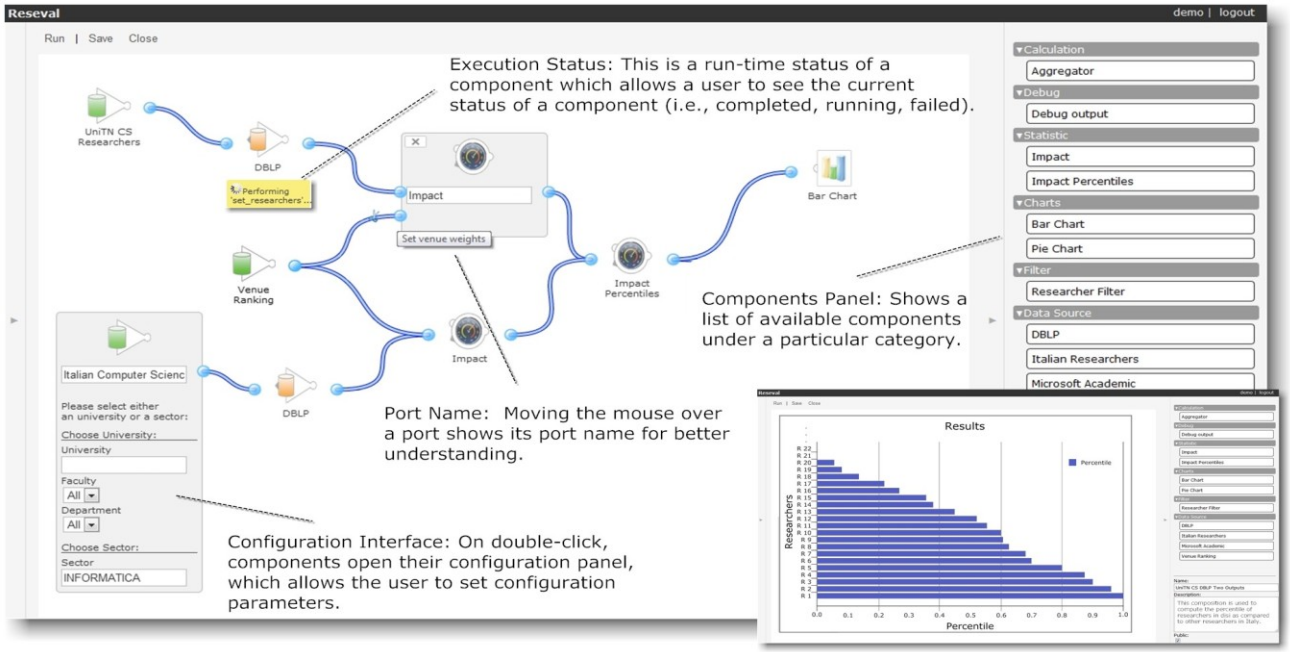

Figure 3: ResEval Mash in action: screen shots of the modeling canvas and its various parts

service) which requires to process big data. All the compositions run at the client side. Only the sever side components calls their corresponding services in order to perform their respective functionality. A service takes data either from a database or shared memory depending on data availability in shared memory. After processing, the data is again stored back to the shared memory. Only a control signal is passed to the respective component in order to convey the state of processing, which is then passed to next components. In Figure 3 we show ResEval Mash composition editor and its various parts along with their description.

Related work

The requirement for more intuitive development environments and design support for end-users clearly emerges from research on end-users (EUD), for example for web services [1, 2], little is available to satisfy this need. There are currently two main approaches to enable less skilled users to develop programs: in general, development can be eased either by simplifying it (e.g., limiting the expressive power of a programming language) or by reusing knowledge (e.g., copying and pasting from existing algorithms).

Web mashups [3] emerged as an approach to provide easier ways to connect together services and data sources available on the Web [4], together with the claim to target non-programmers. In general mashups aim to bring together the benefits of both simplification and reuse. In the case of domain-specific mashup environments, we aim to push simplification even further compared to generic mashup platforms by limiting the environment to the needs of a single, well-defined domain only. Reuse is supported in the form of reusable domain activities, which can be mashed up.

The idea of focusing on a particular domain and exploiting its specificities to create more effective and simpler development environments is supported by a large number of research works [5, 6, 7]. Mainly these areas are related to Domain Specific Modeling (DSM) and Domain Specific Language (DSL). In DSM, domain concepts, rules, and semantics are represented by one or more models, which are then translated into executable code. Managing these models can be a complex task that is typically suited only to programmers.

Evaluation and Lesson Learned

We have performed a preliminary user study of ResEval Mash in the form of contextual interviews with 10 domain experts (5 with and 5 without IT skills). We mainly checked two aspects, i.e., the user preference between generic (in our case Yahoo! Pipes) versus a domain-specific tool (ResEval Mash) and the intuitiveness of a domain-specific tool. The results clearly show that users feel more comfortable with a domain-specific mashup approach as compared to a generic one. Users positively rated the balance between the flexibility, usability, and complexity of ResEval Mash.

In 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, the components know themselves which data to use. This is a very simple, but powerful simplification.

Refrences

[1] Namoun, A., Nestler, T., and De Angeli, A. Conceptual and Usability Issues in the Composable Web of Software Services. ICWE 2010. Springer, 396-407.

[2] Namoun, A., Nestler, T., and De Angeli, A. Service Composition for Non-Programmers: Prospects, Problems, and Design Recommendations. ECOWS 2010, IEEE, 123-130.

[3] Yu, J., Benatallah, B., Casati, F., and Daniel, F. Understanding Mashup Development. IEEE Internet Computing 12, 44-52.

[4] Hartmann, B. and Doorley, S. and Klemmer, S.R. Hacking, Mashing, Gluing: A Study of Opportunistic Design and Development. Pervasive Computing, 46-54.

[5] Costabile, M. F., Fogli, D., Fresta, G., Mussio, P., and Piccinno, A. Software Environments for End-User Development and Tailoring. PsychNology Journal 2, 1, 99-122.

[6] Mernik, M. and Heering, J. and Sloane, A. M. When and how to develop domain-specific languages. ACM Comput. Surv. 37, 4, 316-344.

[7] France, R. and Rumpe, B. Domain specific modeling. Software and Systems Modeling 4, 1-3.