Making mashups actionable through elastic design principles

Carmelo Ardito[1], Maria Francesca Costabile[1], Giuseppe Desolda[1], Markus Latzina[2], Maristella Matera[3]

[1]Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro
Via Orabona, 4 – 70125 – Bari, Italy
{carmelo.ardito, maria.costabile, giuseppe.desolda}@uniba.it
[2]Strategic Projects, Products & Innovation – Technology, SAP SE
Dietmar-Hopp-Allee 16 – 69190 – Walldorf, Germany
markus.latzina@sap.com
[3]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano
Piazza Leonardo da Vinci, 32 – 20134 – Milano, Italy
maristella.matera@polimi.it

Abstract. This paper discusses motivations and requirements leading to *elastic environments* where relevant information and the functions that can be performed on it can be shaped by end users at runtime. As a solution for creating elastic environments, a framework is presented which exploits methods for the *mashup of heterogeneous resources* and elastic features that permit the easy transition of information between different task contexts according to the recently proposed notion of *transformative user experience*.

Keywords: End-User Development, Transformative User Experience, Data Integration, Mashups, Composition Platforms, Elasticity, Task Semantics

1 Introduction and motivation

Due to the new technological landscape (e.g., cloud computing, the software as a service (SaaS) paradigm, the new "API economy" and the resulting service ecosystems), an extraordinarily high number of data is available online. Today, almost any person uses sophisticated mobile devices supporting the pervasive access to data and applications; this determines an increasing demand by the end users (called "users" in the rest of this paper) to effectively access, integrate, and visualize the information offered by such resources. In this respect, platforms for service composition play an important role as they let users integrate heterogeneous information that otherwise would be totally unrelated [1]. Web mashups are indeed "composite" applications constructed by integrating ready-to-use functions and content exposed by public or private services and Web APIs [2]. As compared to consuming what is offered by each single resource in isolated ways, mashup platforms enable users to aggregate information coming from the various resources and create synchronized visualizations. In such ways, mashups generate new value.

Several mashup tools proposed so far, the so-called mashup makers, provide graphical notations for combining services [3-6]. An example is Yahoo!Pipes [7] (for other examples see [2]). As compared to manual programming, such platforms alleviate the mashup composition tasks, but they require an understanding of the integration logic (e.g., data flow, parameter coupling, composition operator programming). Studies with users show that they are still difficult to use by non-technical users (e.g. [8]). According to the End-User Development (EUD) vision, enabling a larger class of users to create their own applications requires intuitive abstractions and notations. To reach this goal, we have developed a mashup platform, described in [1], which is based on the EUD vision and exploits a meta-design approach to support users in mashup creation. More details on how users create their own mashups with this platform are illustrated in [1, 9].

Our recent research on the EUD of mashups has led us to identify some strengths and weaknesses of the proposed approaches. In particular, on the basis of findings of user studies that we performed to validate our mashup platform [10], we believe there is still room for enhancing the mashup paradigm, to empower the users to play a more active role than just consuming the finally visualized information. Transitions across different usage situations, which imply different functionality to be applied on information, should become possible without requiring users to switch among multiple applications. This means that rigid schemas for information provisioning and fruition, generally adopted by isolated, pre-packaged applications, have to be overcome by instrumenting systems with an intrinsic flexibility. The application functionality must dynamically emerge at runtime, based on the users' actions that determined the current situation, i.e., the context and tasks performed.

This paper addresses such a need for *elasticity* and also presents, as a possible solution, a framework where mashup composition paradigms are revisited and potentiated through the notion of Transformative User Experience (TUX) [11]. TUX is a recently proposed approach that aims to natively support users in a variety of spontaneously self-defined task flows, not limiting them to work along highly specific use cases, as typical for applications which are driven by workflow engines or which adopt predefined patterns of guided procedures. The goal is to overcome common application boundaries enabling user interaction with information in terms of *task objects* (i.e., data elements, their visualization and specific functions used to perform a task) within dedicated, contextual task environments assembled through interrelated sets of *task containers*. The distinctive feature of such containers is that they provide functions to process the data they include that strictly relate to the current context as informed by the task actually performed by the users. Thus, the users' task flow is not predefined, but it is determined at runtime based on the users' actions, as the users select proper containers depending on the current situation and on the functionality (e.g., data manipulations) needed to further proceed with their task. In the architectural framework resulting from the integration of the mashup paradigm and TUX, elasticity is thus pursued by allowing users: *i)* to select and combine pertinent data sources through mashup composition; *ii)* to explore and manipulate the integrated data sets in ways that allow them to move across various task contexts while performing varying functions that become available depending on the current usage context. In this way, the

information displayed by the mashup becomes actionable, thus really useful with respect to the users' concrete tasks and overall purpose.

In this regard, this paper proposes a systematic approach to establishing actionable mashups, outlining a framework in Section 2. The demo description provided in [9] reports a scenario motivating such framework. Section 3 provides the conclusions.

2 A Framework for Actionable Mashups

This section describes how to extend the coverage of mashups by augmenting information exploration, generally operated on top of mashup data sets, towards more active *prosumption* (i.e., genuinely merging "production" and "consumption") and sense making. The important feature we focus on is to support the accomplishment of sophisticated sense making tasks on the visualized information thanks to additional manipulations driven by task semantics. In other words, we aim to enable a kind of *active* sense making, in which the presented information can not only be viewed differently and in meaningful ways towards the gaining of insights, but moreover transformed effectively towards the actual accomplishment of task goals. In this regard, the visualizations of data retrieved from data sources, that in a mashup environment can occur by means of *UI templates*, are enriched by augmenting the UI templates with the notion of TUX *task containers*, i.e., elements whose role is to supply task-related functions for manipulation and transformation of task objects along user-defined task flows [12]. As a consequence, through task containers and their particular task semantics, users are empowered to interact with the displayed information in a contextual manner, thus raising information in mashups to the level of task objects the user can act upon.

As represented in Fig. 1, *system objects* (i.e., data items), resulting from the mashup, and their visualizations within UI templates (*UI objects*) can be promoted to the role of *task objects* that in turn can be endowed with and treated according to the various *task functions* offered by the containers in which they are cast. Task objects - not simply data items or their representation in UI templates - become the very objects of user interaction, with the result that the users are not only allowed to consume the information displayed by the mashup, but they are also enabled to manipulate and transform it, i.e., to *prosume* it, in accordance with the tasks they intend to perform. In principle, mashups – without considering TUX principles – can be equipped with some functionality that has task-semantic character, exceeding the mere modification of data visualizations. Yet, in such cases the task semantics would reside in the application implicitly and in a rather hard-wired fashion. For example, a component for the visualization of products could be enriched with a functionality to send emails to vendors. However, this would be a hard-coded function, which the users could not adapt flexibly into their spontaneously defined task flow. According to TUX, it would be instead possible to apply the communication capability to other object types, for example to submit inquiries on the products to consumer forums.

Fig. 1 illustrates the organization of a framework supporting this new task-centric perspective on the organization of an EUD system based on the mashup paradigm.

Modules supporting mashup composition and execution are integrated with modules for the manipulation of task objects according to TUX principles. Typical mashup modules are exploited to create the base of UI objects to be then manipulated as task objects. Within the *mashup engine*, the *data access module* extracts data from the services on which the system relies on (by means of the *mashup components* [2]). The *integration module* interprets user composition actions performed at the UI level and creates an execution model determining how system objects have to be integrated. The results, i.e., the integrated system objects, are rendered as UI objects within *UI templates*. Such UI objects provide the actionable information on which task functions can be applied. In this sense, UI objects are promoted to the level of task objects by virtue of the functionality provided by the *task-semantic layer*. In Fig. 1, the dotted-dashed line connecting a UI template (used for rendering various views of UI objects) and a task container (hosting task objects) makes this promotion explicit.

The task-semantic layer then provides for the identification of the current task contexts, based on the interpretation of user actions as they manipulate *task objects* by applying container-specific task functions; it also supports the *casting* of task objects within and across various *task containers*. At the UI level, a task container "wraps" mashup UI templates, so that the user can act on the displayed UI objects by means of the task-related manipulations. This results in treating UI objects as task objects by virtue of their interpretation through the context, which is defined and provided by each task container. Different UI templates within a task container can be used for providing different views of the same task objects without changing however the semantics of the objects as implied by the task container. Changes of views would in fact still be in line or even supportive of the particular task semantics.

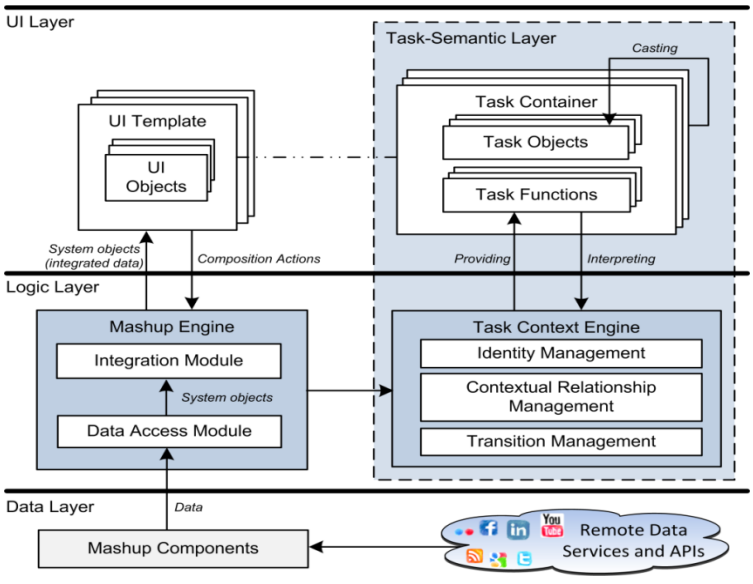

Fig. 1. Overall organization of the framework supporting the interaction with mashups enhanced according to TUX principles

It is worth noticing that, in order to associate different task semantics to data extracted from heterogeneous resources, it is important to maintain continually the relation of the elements representing the task objects to their original context. According to the framework shown in Fig. 1, establishing and maintaining the identity of task objects (as data returned by a given resource) is supported by the task context engine, in particular by its *identity management* component.

Another challenge is to deal with the need of users to endow objects with meanings that depend on the task they choose to accomplish. From the system perspective, a *contextual relationship management* module (see Fig. 1) allows task objects to be augmented by users with subjective meanings and functions that relate to the task semantics of the selected containers where the interaction with the objects takes place. More specifically, this is handled by the generic function of *casting,* which implies that task objects are exposed to the aforementioned container-specific task semantics.

In any concrete scenario, users may interact with task objects in a sequence which spans multiple containers, along spontaneously defined trajectories that however have to keep track of the sequence of the various task semantics a given set of task objects was subjected to. The overall process can thus be considered a kind of "sequential casting" that in the framework of Fig. 1 is managed by the *transition management* module.

3 Concluding Discussion

Mashups are data-centric applications that assist users in easily composing heterogeneous data sources to support information retrieval tasks. They are considered a solution for the important trend of data exploration processes, which exceed one-time interactions and allow users to progressively seek for information. However, some factors are still preventing a wider use of mashups in real contexts. New ways are needed to support sense making on information composed through mashups. We are confident that the introduction of task containers, as entities which carry task semantics, can accommodate such user requirements.

The development of a suitable methodology based on the possible synergies between TUX principles and mashup composition methods has a foundational character that can solve several challenges. We are aware that much work still needs to be done in order to obtain a working platform. With this paper we however aim to stimulate a new way of thinking towards the definition of systems that really support users in shaping the software environments they interact with, according to their actual and emerging needs.

Acknowledgments. This work is partially supported by the Italian Ministry of University and Research (MIUR) under grant PON 02_00563_3470993 "VINCENTE" and by the Italian Ministry of Economic Development (MISE) under grant PON Industria 2015 MI01_00294 "LOGIN".

References

1. Ardito, C., Costabile, M.F., Desolda, G., Lanzilotti, R., Matera, M., Piccinno, A., Picozzi, M.: User-Driven Visual Composition of Service-Based Interactive Spaces. Journal of Visual Languages & Computing 25(4), pp. 278-296 (2014)
2. Daniel, F., Matera, M.: Mashups - Concepts, Models and Architectures. Springer (2014)
3. Ennals, R., Brewer, E., Garofalakis, M., Shadle, M., Gandhi, P.: Intel Mash Maker: join the web. SIGMOD Rec. 36(4), pp. 27-33 (2007)
4. Copeland, T.: Presenting archaeology to the public. In: Merriman, T. (ed.), Public Archaeology, pp. 132-144. Routledge, London, UK (2004)
5. Wong, J., Hong, J.I.: Making mashups with marmite: towards end-user programming for the web. In: SIGCHI Conference on Human Factors in Computing Systems (CHI '07), San Jose, California, USA. pp. 1435-1444. ACM, New York, NY, USA (2007)
6. Daniel, F., Casati, F., Benatallah, B., Shan, M.-C.: Hosted Universal Composition: Models, Languages and Infrastructure in mashArt. In: Laender, A.F., Castano, S., Dayal, U., Casati, F., Oliveira, J. (eds.), Conceptual Modeling - ER 2009. Vol. LNCS 5829, pp. 428-443. Springer, Berlin Heidelberg (2009)
7. http://pipes.yahoo.com/pipes/ - Last access: March, 4th, 2015
8. Namoun, A., Nestler, T., De Angeli, A.: Conceptual and Usability Issues in the Composable Web of Software Services. In: Daniel, F., Facca, F. (eds.), Current Trends in Web Engineering - ICWE '10. Vol. LNCS 6385, pp. 396-407. Springer, Berlin Heidelberg (2010)
9. Ardito, C., Costabile, M.F., Desolda, G., Latzina, M., Matera, M.: Hands-on actionable mashups. In this volume
10. Ardito, C., Bottoni, P., Costabile, M.F., Desolda, G., Matera, M., Picozzi, M.: Creation and Use of Service-based Distributed Interactive Workspaces. Journal of Visual Languages & Computing 25(6), pp. 717-726 (2014)
11. Latzina, M., Beringer, J.: Transformative user experience: beyond packaged design. Interactions 19(2), pp. 30-33 (2012)
12. Beringer, J., Latzina, M.: Elastic Workplace Design. In: Wulf, V., Randall, D., Schmidt, K. (eds.), Designing Socially Embedded Technologies in the Real-World. Springer, London (2015)