End User Development of Multidevice and Collaborative Mashups

Matteo Picozzi

Politecnico di Milano
Dipartimento di Elettronica, Infrmazione e Bioningegneria
picozzi@elet.polimi.it,
http://home.deib.polimi.it/picozzi

Abstract. Web 2.0 and the consequent revolution influenced not only the technologies but also the way of thinking of end users, now "participating" in the creation of contents and more and more of applications. Mashups are applications that integrate contents, behaviors and interfaces of diverse resources, whether public of proprietary and are based on a simplified development process, based on the reuse of ready-to-use components. In the last years, in order to allow end users to create mashups, many tools have been proposed but the most of them had not a long life. With this research, we want to define models, composition methods and implement tools for end user development of mashups.

Keywords: Mashups, End User Development, Personal Information Spaces, Common Information Spaces

1 Introduction

Within the Web 2.0 revolution, among a pletora of novelties, new technologies, new standards and new development practices were introduced. In particular, the use of scripting languages, the diffusion of Web Services, public APIs, the increasing diffusion of basic skills in the development of Web applications and the rise of the culture of participation [12, 16] delineate the scenario in which mashups were born. Mashups are applications that integrate, at different levels, data, functionality and user interfaces from different resources which include Web Services, public APIs or enterprise databases. Mashups emerged in response to the need of users, not necessarily experts of technology, to quickly assemble Web resources to create new Web applications solving their situational needs. For their situational nature, mashups are often developed by users, *i.e.*, the people who need them are also their users. This aspect was also a great limitation for the diffusion of mashups. In fact, to open mashup development to a larger class of users, in the last years, a number of tools, based on easy composition languages, have been proposed. Unfortunately, the most of such tools, after a couple of years of activity, were dismissed.

My research aims at investigating composition models, paradigms and tools for the end user development of mashups. It originates from the need of end users in some specific communities to create applications able to satisfy flexibly their need. I indeed

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

applied the results of my work to the enterprise context [6] first, and then to a Cultural Heritage scenario where professional guides of archeological parks need to create applications to support their guided tours [3]. Some of the causes of the failure of many mashup tools are the incompleteness with respect to the users needs and the difficulty of use. For these reason the approach I'm working on aims at covering the whole mashup life cycle and includes a new composition paradigm that supports the End User Development. We have developed a new visual template-based model which allow users, through a visual paradigm, to: integrate data belonging to diverse resources, create components that can be used in a mashup composition dashboard, or generate a standalone mashup that can be executed on different kind of devices (*e.g.*, tablet or multitouch screens). Finally, we have introduced collaboration in the mashup life cycle, allowing groups of users to collaborate real-time. All these techniques and visual paradigms allow end users to participate to the social innovation by integrating data from third parties, building mashup applications with different difficulty levels and collaborate by sharing contents and editing those contents together in real-time.

This paper is structured as follows. Firstly the research question of my work and the research problem are stated. Than, on the basis of the state of the art and the adopted research methodology, some preliminary results and the definition of the approach for the end user development of mashups are presented. The results so far achieved are shortly illustrated and the conclusions outlined.

1.1 Research Question

The general research question that guides our research is the following:

How can we enable end users to develop mashups?

This implies conceiving methods, visual paradigms and metaphors to enable end users, who might also be unskilled, to develop applications that meet their situational needs by reusing already available resources with a very low effort and in a reasonable time. As described in the next session, there is a need of appropriate models, technologies and tools to support such user needs.

2 State of the Art

There are several kinds of mashups: they could consist of just a resource inclusion in a page, at one of the three tiers of a Web application (*i.e.* data, logic and interface), or could be composite applications whose modules, called *components*, are orchestrated and synchronized in a sophisticated manner. In the literature, a lot of tools have been proposed to allow the development of diverse kinds of mashups by end users. However, most of those tools were dismissed after a couple of years of activity or were research projects that have never been really adopted.

Some projects have focused on easing the creation of effective presentations on top of Web services, to provide a direct channel between the user and the service. *SOA4All* [17] is a tool that facilitates the creation of service infrastructures and increases the interoperability between large numbers of distributed and heterogeneous functionalities

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

on the Web. *SOA4All* concentrates on the establishment of an instance of a service delivery platform that is optimized and tailored to the needs of Web services. *ServFace* [20] adopts a model-driven development approach that applies Service annotations to a visual authoring of Service-based interactive applications. *Dynvoker* [22] is a tool that allows users to generate forms dynamically to invoke services in many scenarios, including rapid service testing and dynamic inclusion of services as plugins into applications. However, such approaches do not allow the composition of multiple services into an integrated application.

There is also a considerable body of research on mashup tools, the so-called mashup makers, which provide graphical user interfaces for combining mashup services. *Yahoo! Pipes* [1] is one of the most popular mashup tools. Pipes is a wired environment in which mashup feeds are accessed and "piped" through user-selected functionality (combine, filter, sort, split, count, truncate and so on) to process the feed. It provides also a community for building, sharing, rating and modifying mashups. *MashArt* [8] is a Web tool that provide universal composition as a service in form of an easy-to-use graphical development tool equipped with an environment for fast deployment and execution of composite Web applications though the pipe metaphor.
With respect to manual programming, such platforms certainly alleviate the mashup composition tasks. However, to some extent they still require an understanding of the integration logic (*e.g.* data flow, parameter coupling, and composition operator programming). In some cases, building a complete Web application equipped with a user interface requires the adoption of additional tools or technologies. Even when they offer an easy composition paradigm, they do not guide at all the composition process. A study about users' expectations and usability problems of a composition environment for the the *ServFace* tool provides evidence of a fundamental issue concerning conceptual understanding of service composition (i.e. end users do not think about connecting services) [18].

Dapper [11] is a tool powered by *Yahoo!* that allows one to collect information from web pages or feeds and create a *Dapp*. A *Dapp* is a container of content that could be of different formats, *e.g.*, XML, CSV, JSON, HTML but also Google Gadget, Google Maps and iCal. *Dapper* therefore produces customized contents using Web pages data. Our tools also allow users to combine such data with other resources. We can thus consider *Dapper* as useful addition to the *Component Editor* (see Section Ideas and Results); we are indeed planning to include Dapps into our *Mashup Dashboard* and allow users to use them as mashup components.

JackBe Presto and *IBM Damia* are enterprise-oriented tools which offer support to integration, reporting and visualization of enterprise data. *JabkBe Presto* [15] is a Real-Time Operational Intelligence software with real-time Business Intelligence analytic. *IBM Damia* [21] is a data integration platform that allows business users to quickly and easily create data mashups that combine data from desktop, web, and traditional IT sources into feeds that can be consumed by AJAX and other types of web applications. It is easy to customize our platform in order to obtain an environment similar to the two previous tools. If enterprise databases are queried by an *ad hoc* service, it is easy to wrap components able to visualize data. In particular we have chart component that can visualize data aggregations through diverse kind of information visualizations.

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

Another investigated aspect in the state of the art is the collaborative nature of mashups. For example some tools, as *e.g. Yahoo! Pipes*, also provide a community for mashups developers. Collaboration in mashup-based development can be beneficial in collective intelligence scenarios [13], where teams of people co-create knowledge by sharing integrated information spaces with professional peers, in meta-design environments [12], where end users shape up their tools in collaboration with expert developers, or in scenarios where people, not able to develop by themselves their own applications, ask for help and advice from experts within reference communities in a kind of crowdsourced Web Engineering [19]. In [14] Heinrich et al. describe widgets for collaborative Web applications which can be re-used in different applications. Such collaboration approaches are similar to the ones we adopt in our tools because they are inspired to the collaboration mechanisms that nowadays are offered by the most popular collaborative environments, one for all *Google Documents*. However in our tools we do not only capture the real-time information in order to notify what other users are doing, but we also provide other kinds of asynchronous collaboration, such as annotations [2].

3 Research Methodology

The methodology applied in this research is composed by the following steps: (i) analysis of the state of the art and outline of lacks or not considered aspects; (ii) proposal of solutions to fill the lacks, *e.g.*, models, composition paradigms and technology solutions; (iii) implementation of the proposed solutions; (iv) validation through user experiments; (v) refinements trough iterations of the previous points.

We set our research with an incremental method. The first part of the research focused on technological solutions for the composition of mashups and the synchronization among components by means of a tool for mashup delevopent [6]. The second part focused on the component creation, with particular attention to the execution of the created components by standalone execution environments on different devices. The last part was related to the collaborative creation of mashups and components. At the end of each part of the research, we conducted an experiment in order to evaluate and validate the achieved results and proceed by resolving the emerged issues. In the next section, ideas and results emerged from the research are presented and, in the following section, the user-based evaluation will be described.

4 Ideas and Results

The analysis of the state of the art, in particular the aforementioned tools and [10], suggests a critical analysis on the life cycle of mashups. If we consider the life cycle of collaboration-oriented mashups in Figure 1, we can notice that most of the mashup tools so far proposed do not cover many of the development activities.

We identified the activities reported in Figure 1 as fundamentals for providing users with feasible approaches where they are enabled to create components, not only to compose them, and to collaborate with their peers. For this reason we have concentrated on trying to develop a tool that aims at being more complete, flexible and as easy-to-use as possible. Our conclusion was that such tool must (i) cover the whole mashup life

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

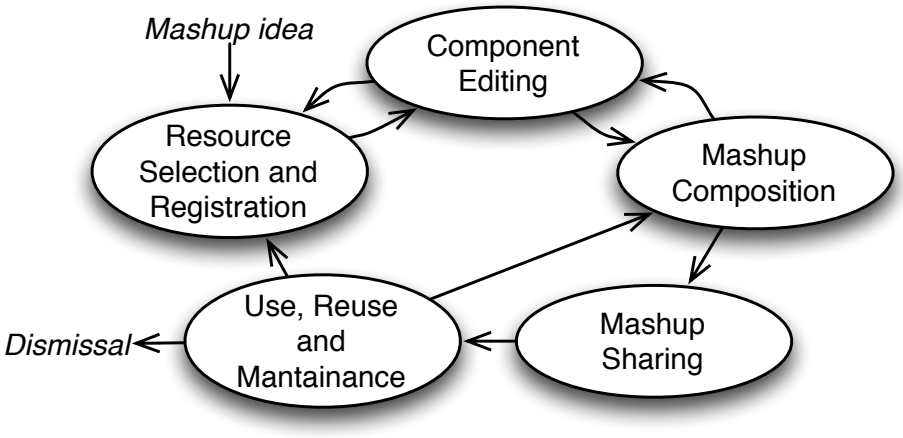

Fig. 1. Life cycle of a collaboration-oriented mashup. Based on the lifecycle described in [9].

cycle; (ii) be able to cover as many component types as possible; (iii) be flexible, *i.e.*, must allow users to add more resources that are deemed relevant for their applications; (iv) support the deployment of mashups on as more devices as possible and (v) be social and collaborative.

4.1 A Platform for End User Development of Mashups

Given the previous objectives, we propose PEUDOM, a Platform for End User Development of Mashups. PEUDOM is composed by two main environments, *i.e. Component Editor* and *Mashup Dashboard* and supports the sharing of components and the live collaboration among users. Figure 2 illustrate the overall architecture that we describe in detail inn the sequel of the section.

Component Editor The *Component Editor* allows end users to register REST services in the platform, to query them in order to use their data to assemble components. Component descriptors can be exported and executed on the *Mashup Dashboard* or as standalone application, coded according to the target device technology, able to interpret it, retrieve data and instantiate the component schema on the specific device. Those applications could be deployed on diverse devices like smartphones, tablets or large multitouch displays. Figure 2 shows a screenshot of the *Component Editor*. Selecting different services the data will be fused in the visual template, according to a lightweight, client-side data integration technique [4]. Through the *Service Manager* module (Figure 2), the *Component Editor* offers support for querying REST services, displaying the retrieved results in a visual format, and visually defining selection and projection queries over such results. A visual mapping process indeed allows the user to select data attributes and associate them to user interface elements playing the role of data collectors. The association of data from multiple services to a same UI data

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

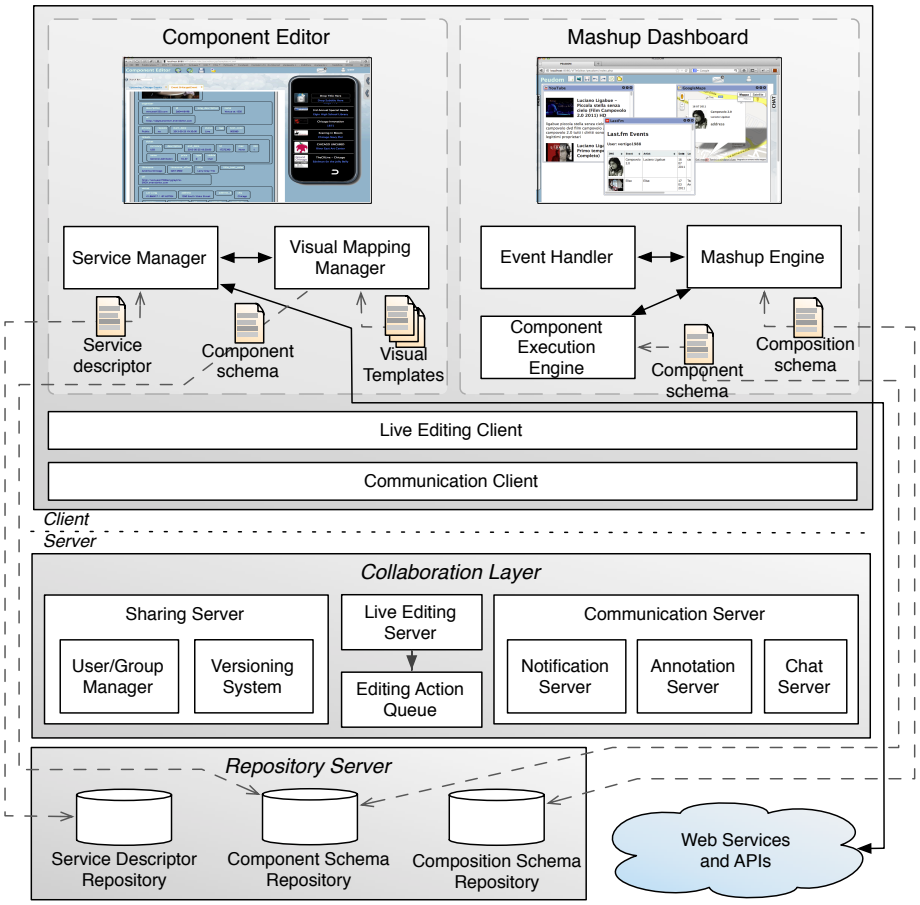

Fig. 2. Overall architecture of PEUDOM

collector also defines integration queries [4]. The editor, through the *Visual Mapping Manager*, translates the visual actions into an XML-based component schema. At runtime the execution engines will be able to interpret this rule and to render the selected data, retrieved by the service querying, at the right place in the visual template.

Mashup Dashboard In our mashup platform, component integration complies with an event-driven, publish-subscribe paradigm that enables the synchronization of components behaviors at the presentation level. Components wrap (remote or local) services and expose events and operations. The coupling of components within an integrated workspace is based on the subscription of operations, which become listeners for events exposed by other components. Subscriptions are expressed through a composition schema represented in an XML-based domain specific language [6], which is then used to govern the execution of the mashup, *i.e.*, the synchronization of the different

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

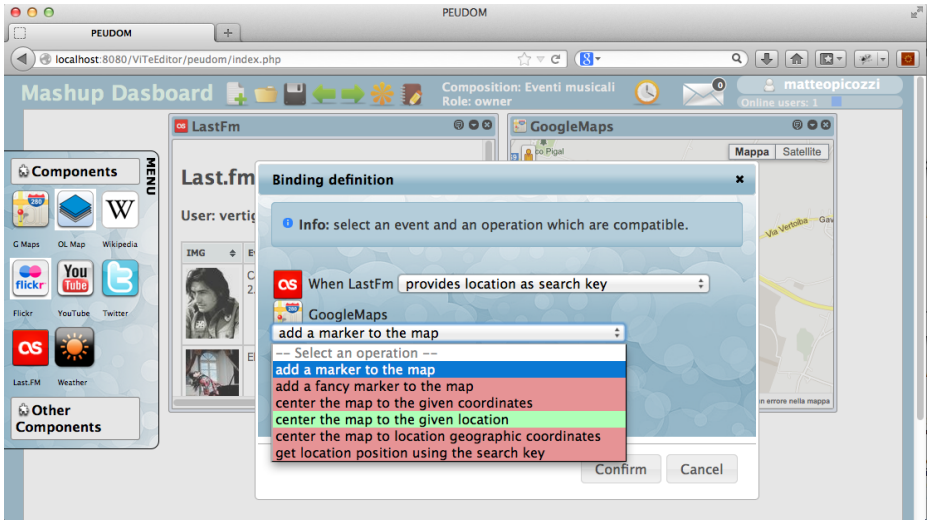

Fig. 3. Screenshot of the mashup dashboard

components according to the defined listeners. The *Mashup Dashboard* is a Web environment where users can create new mashups or modify already exiting ones, which could be developed by the same end user or by other users and shared with him/her. Components are represented as windows (as shown in the *Mashup Dashboard* screenshot in Figure 2) and can be dragged around the workspace. A left-handed palette allows the inclusion of other registered components. In the top-left corner of the component window, there is the component icon: when other components are included in the same mashup, users can drag this icon of a component (source) on another component (target) and, if the target component is compatible, it will be possible to drop the icon on it, enabling the subscription of the target component operation to the source component event, by selecting the event and the operation from two drop down lists.
As shows in Figure 2, from an architectural point of view, an *Event Handler* on the client-side intercepts the visual composition actions executed by the end user, and automatically translates them into elements, *i.e.*, listeners and property values, of the mashup. Based on the so created schema, the *Mashup Engine* acts as an event bus: it listens and handles the events raised by the interaction with each single components, and activates the subscribed operations as prescribed by the listeners in the composition schema.

A relevant characteristic of our approach is the interleaving of the design and execution phases [6]: users immediately experience the effect of their composition actions (*i.e.*, the composition schema is immediately interpreted and executed); thus they can iteratively and interactively refine the resulting applications. This is in line with some end user development principles, which state that immediate and continuous feedback of what end users are doing, help them in the development.

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

Collaboration Modules PEUDOM exploits a "lightweight" execution paradigm, hosting all the modules for composing and executing the composite information spaces at the client-side. As highlighted in Figure 2, server-side modules instead manage the sharing of resources by multiple users and the synchronous and asynchronous communication. A schema versioning, an annotation system and an activity log system support a synchronous communication. Instant messaging and live editing then enable synchronous collaboration. For example, if a user performs an action on a client that modifies locally a shared mashup, this action is propagated (through the *Live Editing Client*) to a *Live Editing Server* in charge of updating the composition schema on each listening client. The server maintains a representation of all the distributed editing actions: every editing session on a shared mashup is associated with an *Editing Action Queue* from which messages are broadcasted to all the active shared mashup instances, except the one where the modification originated. On a client listening to modification actions, the *Live Editing Client* interprets the received actions and actuates the changes on the local schema. The server-side management of the editing action queue ensures the synchronized evolution of all the active mashup instances. With respect to the paradigm adopted for mashup construction, the shared mashup composition schema is now enriched with status meta-data (*e.g.*, parameter values to query single components, items selected in a given data set), so that each instance is synchronized not only with respect to the composition structure (*i.e.*, components and listeners), but also with respect to behavioral aspects, *e.g.*, the displayed data set filtered out by different actions of concurrent users. Therefore, the composite application is now long-lasting and stateful: both structure and state variables are maintained across different sessions.

Domain Specificity The composition paradigm illustrated above is not tied to any specific domain. Our platform is indeed open in its nature, being it conceived for the integration of heterogeneous services, based on different visual templates and user interfaces supporting mashup composition. This openness facilitates the customization of the platform with respect to the characteristics and needs of specific communities of end users. Customization, for example, occurs by selecting and registering into the platform services and data sources (public or private) that can provide content able to fulfill relevant user information needs. Service registration is kept simple, to allow even inexperienced users to add new services as they need them. Registration requires the user to input, by means of visual forms, the service URI and the value of some search keys to execute basic service queries. Domain specificity is also obtained by the development of execution engines for the components created by the *Component Editor*, using specific terminology and customizing the interaction on the target users needs, as outlined by Casati in [7].

New visual templates for data visualization can also be easily added, by instrumenting the design environment for the mashup composition based on the new templates, and by extending the execution environments with mechanisms for the instantiation of the corresponding user interface. In our current version of the platform, we have implemented map-based, list-based and graphic-based templates. It is worth noting that the schema generated by the *Component Editor* makes use of conceptual elements, called visual renderers, which are independent of the chosen visual template. Visual renderers

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

are generic receptors of data that, based on the user choice, act as placeholders for data visualization during the component execution. The component schema consists of a set of *visual renderers*, each one representing the integration of data items coming from different resources. The way visual renderers are displayed according to a given layout then depends on the visual template selected by the user at design time, which in turn implies a specific mapping between the visual renderers in the abstract model and the widgets in the concrete visualization layout. The addition of a new template basically requires the definition of a new visualization layout, and of the mapping between the abstract visual renderers and the concrete elements of the layout that will be used to instantiate the component user interface. The composition paradigms and the rule for schema generation remain unchanged. Different from service registration, the definition of new visual templates necessarily requires the intervention of technicians (typically platform administrators). Nevertheless, this extra-effort is justified by the fact that the customization of visual templates introduces a level of specificity, in relation to specific domains, that is needed to increase the value of the platform with respect to the addressed end users.

4.2 User Study Evaluation

In order to evaluate our approach, so far we have conducted three user studies to test the main parts of the platform, following the incremental flow described in the Section Research Methodology. Each part ended with a user study session to evaluate the current step. The first study was about the composition of mashups on the first version of the Mashup Dashboard, the second study was about the component creation and the exporting and use of the created component on mobile devices and the last study is about the collaboration mechanisms. All the evaluations follow the same four-phase methodology: (i) pre-experience and placement questionnaire; (ii) brief explanation of the experience and demo of the tool; (iii) user experience and observation; (iv) post-experience evaluation questionnaire.

With the user study we want to answer to three main questions in order to understand if the tool it is intuitive and easy to use and if end users would use again the tool in the future.

1. Do end users feel comfortable with the tool?
2. Are they able to accomplish to some tasks that let them to use the most important features and functionalities of the tool?
3. Are they satisfied with the use of the tool and with the work done through it?

During the experiments we observed two different categories of users: the novice users and the expert users with respect to mashups and, in general, to Web application development. We measured their performance in terms of task execution time and number and type of errors. From the analysis of the data collected through questionnaires and observation, it emerged that there is not a relevant difference in the behaviors of these two categories of users. This can be considered on aspect in favor of the ease-of-use of our tools for inexperienced users, who are enabled to achieve a similar performance as the more skilled users. Moreover, we already considered the user's comments

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

and the reactions that we relieved during the evaluation session and we corrected accordingly the platform. In order to conclude the evaluation we will organize other user studies by comparing the behavior of expert users versus novice users and also with respect to another, already existing and successful, tool, *e.g. Yahoo! Pipes*, in order to evaluate and validate our approach and platform and understand other user needs and how to improve our work.

5 Conclusions

The research illustrated in this paper aims at enabling end users to develop mashup applications by their own. Thanks to tools like the one we propose, end users can build customized applications that fulfill their specific needs in a reasonable time and with a limited effort. We, in particular, propose PEUDOM, a platform that allows end users to develop their mashup applications by visual paradigms, which allow users to accomplish different development activities, from the services registration and creation of components to the mashup composition and the collaboration with other users. The platform covers as much as possible the life cycle of the mashup development and enable the reuse of components and mashups created and shared by other users. Comparing our tool with other tools in literature we notice that there are not other tools that cover all these phases and there are not other tools that support a live collaboration paradigm. Moreover, the creation of components integrates data from different resources and produces a XML-based descriptor that can be used also by standalone applications developed for diverse devices from mobile devices to large multi-display screens. As future work we will (i) improve the real-time collaborative editing; (ii) enable the registration of different kind of services other than REST services; (iv) apply recommendation techniques in order to help users selecting adequate resources [5] and (v) increase the flexibility and usability of our environments.

References

1. Yahoo! pipes. `http://pipes.yahoo.com/pipes/`.
2. Ardito, C., Bottoni, P., Costabile, M. F., Desolda, G., Matera, M., Piccinno, A., and Picozzi, M. Enabling end users to create, annotate and share personal information spaces. In *IS-EUD* (2013).
3. Ardito, C., Costabile, M. F., Desolda, G., Matera, M., Piccinno, A., and Picozzi, M. Composition of situational interactive spaces by end users: a case for cultural heritage. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12, ACM (New York, NY, USA, 2012), 79–88.
4. Cappiello, C., Matera, M., and Picozzi, M. End User Development of Mashups. In *Proc. of CHIInternational*, LNCS, in print, Springer (2013).
5. Cappiello, C., Matera, M., Picozzi, M., Daniel, F., and Fernandez, A. Quality-aware mashup composition: issues, techniques and tools. In *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, IEEE (2012), 10–19.
6. Cappiello, C., Matera, M., Picozzi, M., Sprega, G., Barbagallo, D., and Francalanci, C. Dashmash: a mashup environment for end user development. In *Web Engineering*. Springer, 2011, 152–166.

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).

7. Casati, F. How end-user development will save composition technologies from their continuing failures. In *End-User Development*. Springer, 2011, 4–6.
8. Daniel, F., Casati, F., Benatallah, B., and Shan, M.-C. Hosted universal composition: Models, languages and infrastructure in mashart. In *Conceptual Modeling-ER 2009*. Springer, 2009, 428–443.
9. Daniel, F., Matera, M., and Weiss, M. Next in mashup development: User-created apps on the web. *IT Professional 13*, 5 (2011), 22–29.
10. Daniel, F., Matera, M., and Weiss, M. Next in mashup development: User-created apps on the web. *IT Professional 13*, 5 (2011), 22–29.
11. Dapper. `http://open.dapper.net`.
12. Fischer, G. Understanding, fostering, and supporting cultures of participation. *Interactions 18*, 3 (2011), 42–53.
13. Grasso, A., and Convertino, G. Collective intelligence in organizations: Tools and studies. *Comput. Supported Coop. Work 21*, 4-5 (Oct. 2012), 357–369.
14. Heinrich, M., Grüneberger, F. J., Springer, T., and Gaedke, M. Reusable awareness widgets for collaborative web applications - a non-invasive approach. In *ICWE* (2012), 1–15.
15. JackBePresto. `http://jackbe.com`.
16. Jenkins, H. *Confronting the challenges of participatory culture: Media education for the 21st century*. Mit Press, 2009.
17. Krummenacher, R., Norton, B., Simperl, E., and Pedrinaci, C. Soa4all: enabling web-scale service economies. In *Semantic Computing, 2009. ICSC'09. IEEE International Conference on*, IEEE (2009), 535–542.
18. Namoun, A., Nestler, T., and De Angeli, A. Conceptual and usability issues in the composable web of software services. In *Current Trends in Web Engineering*. Springer, 2010, 396–407.
19. Nebeling, M., Leone, S., and Norrie, M. C. Crowdsourced web engineering and design. In *Web Engineering*. Springer, 2012, 31–45.
20. Nestler, T., Feldmann, M., Hübsch, G., Preußner, A., and Jugel, U. The servface builder-a wysiwyg approach for building service-based applications. In *Web Engineering*. Springer, 2010, 498–501.
21. Simmen, D. E., Altinel, M., Markl, V., Padmanabhan, S., and Singh, A. Damia: data mashups for intranet applications. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ACM (2008), 1171–1182.
22. Spillner, J., Feldmann, M., Braun, I., Springer, T., and Schill, A. Ad-hoc usage of web services with dynvoker. In *Towards a Service-Based Internet*, P. Mähönen, K. Pohl, and T. Priol, Eds., vol. 5377 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, 208–219.

Proc. of CHItaly 2013 Doctoral Consortium, Trento (Italy), September 16th 2013 (published at http://ceur-ws.org).