# Creation and use of service-based Distributed Interactive Workspaces ☆

Carmelo Ardito [a,*], Paolo Bottoni [b], Maria Francesca Costabile [a], Giuseppe Desolda [a], Maristella Matera [c], Matteo Picozzi [c]

[a] Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro, Via Orabona, 4 70125 Bari, Italy
[b] Dipartimento di Informatica, Sapienza Università di Roma, Viale Regina Elena, 295 00161 Roma, Italy
[c] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano P.zza L. da Vinci, 32 201233 Milano, Italy

Received 20 September 2014
Received in revised form
11 October 2014
Accepted 13 October 2014
Available online 22 October 2014

## 1. Introduction

Web 2.0 has accelerated the evolution of the Web, becoming a driver for innovation. End users are now involved in the content creation process, a fact which has in turn amplified their will to become active creators of applications that simplify access to the huge quantity of data made available on the Web in heterogeneous formats and protocols.

The mashup phenomenon has been one of the results of this trend towards a "democratic" access to online resources. Mashups integrate heterogeneous services at different layers of the application stack, to provide unified views over integrated result sets fetched from different sources [1–3]. So far mashups have been especially conceived as personal information spaces, i.e., vertical applications solving situational needs, that end users assemble by merging ready-to-use resources [4]. Mashups, however, have a great potential to accommodate the sharing and co-creation of knowledge [5]. As highlighted by the field studies discussed in this paper, in several domains the involved stakeholders need to share, co-create and execute mashups in a distributed manner. Nevertheless, while collaboration mechanisms have been extensively investigated in different areas, the co-creation of interactive workspaces via mashup composition is still scarcely explored.

The work reported in this paper relates to the experience gained in the last few years in the analysis of different paradigms for mashup composition and in experiments with prototypes of a mashup platform [4,6]. The platform exploits End-User Development (EUD) principles and offers a visual, live programming paradigm to let users,

☆ This paper has been recommended for acceptance by S.-K. Chang.
* Corresponding author.
*E-mail addresses:* carmelo.ardito@uniba.it (C. Ardito), bottoni@di.uniroma1.it (P. Bottoni), maria.costabile@uniba.it (M.F. Costabile), giuseppe.desolda@uniba.it (G. Desolda), maristella.matera@polimi.it (M. Matera), matteo.picozzi@polimi.it (M. Picozzi).

not necessarily technology experts, create service-based, interactive Web applications. The novel contribution of this paper is to enable the collaborative creation and use of *Distributed Interactive Workspaces* (DIWs). DIWs are component-based interactive applications where content is produced by end users via the aggregation and manipulation of data fetched from distributed online resources, both local and third-party. DIWs can be deployed as personal applications through a client-side logic supporting the execution of the workspaces on multiple devices. DIWs can also exploit a centralized, server-side, execution logic to manage the sharing of workspaces among different users, the propagation of collaborative actions to active instances of a same workspace, and the distributed execution of a whole workspace, or of selected components, on different devices employed by different users. The peculiarity of the presented approach is that collaboration mechanisms can be applied to different elements of the mashup application: the mashup raw data are fetched from the basic services, through the integrated content resulting from the adopted data integration policy, to the integrated visualizations that in our approach guide the composition process.

As a further contribution, while recent works proposing some form of collaboration in mashup composition cover only specific, limited aspects (e.g., awareness in synchronous editing) [7], this paper shows how collaboration can be supported in different modalities. In particular, it discusses how collaboration for DIW co-creation can be described along two dimensions: (1) the time at which it takes place, and (2) the resources, of a physical or computational nature, used by participants. Concerning the temporal aspect, this paper discusses both synchronous and asynchronous collaboration, among all participants or selected subsets thereof. As for resources, the paper focuses on the different elements of a DIW, namely services, integrated data sets, and visualizations, and shows how they can be co-created during the collaboration process itself, or be produced or retrieved by individual participants working on their own. Although the two aspects are orthogonal, in the sense that systems and procedures can accommodate any combination of them, some problems may arise concerning the use of individual resources during collaborative work: How to smoothly integrate them in synchronous sessions? How to asynchronously communicate the availability of new resources to collaborators? How to protect resources intended only for personal usage, and how to change their status to public?

This paper shows how such questions have been addressed by integrating a general collaboration process in the creation and management of DIWs. The need for collaboration to co-create and share DIWs emerged from some formative studies in different application domains. Thus, the composition platform illustrated in [4] was extended to enable annotation and co-creation of service-based interactive workspaces. The paper describes how it is possible to augment the available resources with collaboration-oriented information, and how to make this information available to others without corrupting the original resources. Synchronous collaboration, mainly based on live editing, is also addressed whereby multiple users interactively modify (portions of) a shared DIW, being *aware* of the modifications operated by any participating collaborator.

The paper proceeds as follows. Section 2 summarizes the formative studies and the collaboration requirements that emerged Section 3 introduces the collaborative mechanisms that can be applied on DIWs, while Section 4 describes the platform architecture, with emphasis on the new modules supporting sharing, co-creation and distrib-uted execution of service-based interactive workspaces. Section 5 reports on related work and Section 6 concludes the paper.

## 2. Motivation for collaborative and distributed creation of interactive workspaces

The mashup platform described in [4] allows end users to create interactive workspaces by composing heterogeneous data sources, be they public (e.g., remote resources available in the form of Web services and APIs) or private (e.g., local content for personal use). The platform supports the integration of data into UI *components* [6,8], i.e., widgets that retrieve data from different services and display their integration into a unified user interface (UI). Given a set of distributed services registered into the platform, the user can define parametric, key-value queries through visual forms. The retrieved result sets can then be integrated within UI components. A number of *visual templates* supply possible UIs. Besides serving the rendering of the integrated data, visual templates also provide a unified schema for lightweight data mediation: a visual mapping process allows the user to select data items returned by the selected services and associate such items with visual elements playing the role of data collectors in a visual template. The association of data from multiple services with single UI data collectors determines the definition of *union* and *merge* operations on the involved data sets [6].

The created components can then be included within the workspace under definition, where they can be also synchronized according to an *event-driven*, *publish–subscribe* paradigm, which enables synchronization of *components'* behaviors. Each component exposes *events* and *operations* [1]. The coupling of components within a workspace is thus based on the subscription of *operations*, which become *listeners* for *events* exposed by other components. As a result, invoking an operation changes the state of the interested components.

With respect to other proposals for mashup composition, the peculiarity of this platform lies in its visual composition mechanisms, which make it adequate for end-user development [8]. Due to the extensive use of visual representations guiding the composition process, and the separation between such mechanisms and the logic for mashup composition and execution, the platform is easily customizable with respect to specific user requirements and characteristics. As discussed in [4],customization mainly requires the adoption of *visual templates* offering adequate visual metaphors to the users.

In the next two subsections we report on two field studies we performed in different application domains in order to verify the usefulness for end users of content made available by distributed data sources, as well as the overall validity of our composition approach. The studies were also useful to identify improvements and extensions of the approach, in particular for the collaborative and distributed creation of interactive workspaces. The early prototypes of the platform used in the formative studies performed in the field with actual end users offered limited possibilities for collaborative and distributed creation of workspaces, which in those studies we called "interactive workspaces" (IWs). For reasons of space, not all the details of the studies are reported in this paper.

### *2.1. Field study in the cultural heritage domain*

One of the studies was carried out in the context of visits to archeological parks [4]. Before the visit, two professional guides composed their IWs relative to the archeological park of Egnathia (in Southern Italy) using a desktop application, accessible through a PC placed in their office. A few days later they experimented the use and update of IWs with a large multi-touch display (46 in.) and a tablet device (7 in.) during two guided visits of the archaeological park, involving 28 visitors. Before starting the visit the professional guide interacted with the IW she created, in order to "enhance" her presentation of the history of the park. The IW was then deployed on a large multi-touch display available at the entrance of the park museum. Media contents, such as photos, videos, and wiki pages associated with park locations to be visited during the guided tour, were represented by an icon and a title placed on a Google map centered on the park. In this case, the map was the visual template adopted to guide the content selection and aggregation. By tapping on an icon, a pop-up window visualizes the corresponding media. For example, in Fig. 1a the guide has tapped on an icon on the map to show the picture of an earthenware jar that was used at Roman times. During the park tour, the guide accessed her IW on the tablet, in order to show photos, videos and other information when appropriate (see Fig. 1b).

The study showed a general appreciation of the use of IW in the context of the visit and many interesting insights emerged. Guides would like to *communicate and share information* with other stakeholders and would appreciate collaborating with colleagues, both synchronously and asynchronously, during IW composition. During the interview at the end of the composition phase the guides said they would welcome the possibility of *collaboratively composing* the IW before a visit, even when working at home. They would also like to be able to *ask advice* to colleagues about new services that can provide material they are not able to find through the services they have access to. They would like to *share their IWs* with visitors to allow them to view and possibly add contents. The guides might also need to communicate with software engineers managing the platform, to *ask for modifications* of the user interface structure, or for the introduction of new templates for information visualization.


**Fig. 1.** The guide is using her IW on: (a) a multi-touch display to illustrate the park history to the visitors; (b) a tablet during the park tour.

From interviews with the visitors a general appreciation of the experience emerged, but also two main limitations. First, when the guides were explaining and introducing the visit showing in-depth contents through the large display, they were covering the screen with their body because they needed to be next to the multi-touch screen to interact with. Hence, visitors were not able to see the whole screen. Second, when the guides were outdoor and were showing contents through the tablet, most visitors were not able to see the screen because of its limited dimensions with respect to the number of people in the visiting group. Multi-device collaboration mechanisms could be used to solve these issues by enabling remote control of the contents displayed on the large multi-touch screen or content delivery to visitors' mobile devices.

### *2.2. Field study in the Technology-Enhanced Learning domain*

Another field study, performed in a context of Technology-Enhanced Learning (TEL), allowed us to analyze the use of the platform in a situation in which students learn about a topic presented in class by their teacher, complementing the teacher's class by searching information on the Web [9]. The retrieved information can also be communicated and shared with the teacher and the

other students using interactive whiteboards, desktop PCs and personal devices (e.g., laptop, tablet and smartphone).

The study was carried out at the technical high school "Antonietta Cezzi De Castro" in Maglie, a city in Southern Italy. It was organized over 3 days, involving a class of 16 students (9 females, 19 year-old on average) and a teacher. During the first day, the teacher composed an IW relative to "Communication Networks". Two days later, the teacher gave a lesson supported by the IW visualized on an interactive whiteboard. At the end of the lesson, he divided the students into groups of 2–3; each group was assigned the task of creating an IW about a specific Communication Networks sub-topic, e.g., protocols, packet switching, and latency period. After a brief individual training session, all the groups accessed the laboratory to carry out their assignments. Fig. 2 shows a couple of students working with their IW, to which they are adding widgets visualized through a list-based visual template, to retrieve and integrate contents from Google, Slideshare and YouTube. At the end of this session, we simulated the sharing of their IWs with the teacher by manually integrating their components into a unique IW accessible by the teacher.

After 2 days, teacher and students met again for a class on Communication Networks; this time the class was supported by the integrated IW running on the interactive whiteboard (see Fig. 3). The discussion on the retrieved information lasted for an hour and a half. At the end, teacher and students had 20 min to fill in a short questionnaire inquiring about platform pros and cons they perceived.

A significant part of this field study was a design workshop that was conducted afterwards, in order to better understand the need for collaborating with other people by means of IWs. The design workshop aimed at engaging students and teacher in: (1) elicitation of positive and negative aspects of the overall interaction experience with the platform; (2) active participation in the design of new solutions, primarily stressing the envisioned possibilities of collaborative composition of an IW. The latter objective was derived from the results of the field study in the Cultural Heritage domain indicating the willingness of professional guides to compose collaboratively the IW to be used during a visit. Four groups were formed, each involving four students, one interaction designer, one platform developer and one HCI researcher. One group also included the teacher. Stimulated by the researcher, participants elaborated their ideas about interaction possibilities. Then, they were asked to sketch such ideas (see Fig. 4). The design workshop lasted for an hour and a half. At the end, a plenary session of 30 min was held and the more promising ideas were illustrated and discussed. They were instrumental for the design of both functionality and visual interface of the collaboration mechanisms that, as discussed in Section 3, we next implemented in our platform.

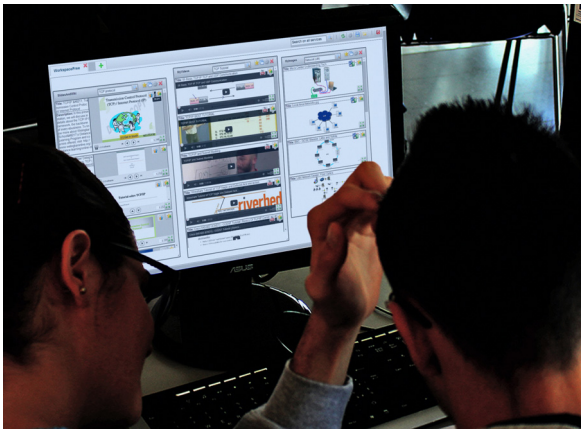

**Fig. 2.** Two students working with their IW on a desktop PC.

**Fig. 3.** A student discussing about Communicarion Networks by using the integrated IW on the interactive whiteboard.

**Fig. 4.** A group sketching interaction ideas during the design workshop.

The analysis of videos and notes taken during the workshop revealed that all groups were very active. In general it turned out that both students and teacher wish *more flexibility* in organizing the interactive workspace, and the need emerged for *visual containers* in which retrieved content can be arranged and classified according to unforeseen needs. They stressed that the platform should be improved to *support collaborative activities* and contributed in the design of possible features and usage scenarios. The teacher proposed a "peer-learning" workspace, in which both teachers and students can *share their contents*, *offer comments* or *create a discussion thread*, and *express their appreciation* in a Facebook or YouTube style.

The students envisaged the possibility of a *distributed collaborative creation* of a workspace, which could be asynchronous in case of a homework assignment or synchronous if carried out in class during a lesson.

### 2.3. Summary of collaboration requirements

In both studies, the availability of live collaboration mechanisms and of annotations at different levels was identified as a key feature of interaction, composition and update of the DIW. Live editing was in particular singled out as a mechanism to show and share, in real time, personal contributions with other stakeholders that could enrich/improve a workspace. Annotations could then be used as personal memos, e.g.: remembering – by highlighting significant parts of a DIW; as expressions of thinking – by adding one's own ideas, critical remarks, questions; and as clarifying elements – by reshaping the information in the DIW into one's own verbal representations. The need for storing items in a frozen form from the dynamic content displayed in the workspace was also stated as a special kind of asynchronous collaboration. Therefore annotations were in general deemed useful for sharing information and communication among DIW stakeholders, as they can support discussions among users having access to a same workspace.

## 3. Collaborative interventions on DIWs

Given the collaboration requirements illustrated in the previous section we now discuss what covering such requirements entails, if a mashup composition paradigm is adopted for the creation of the interactive workspaces. An *interactive workspace* (IW) can be defined as an interactive document corresponding to any instance of a schema that is specified along three main dimensions:

- a *composition model* (CmM), describing the organization of the UI components in the IW and the way they synchronize by means of event-driven publish-subscribe couplings;
- a *content model* (CnM), describing the actual content dynamically fetched by the different services and the way it is integrated into each UI components starting from the retrieved result sets. Given the dynamic nature of IWs, content is specified by means of queries on the involved services, which are in turn expressed according to some service-specific schema;
- a *visual template model* (VTM), describing the presentation aspect of the integrated data sets forming the IW through the association of queries to elements of the adopted visual template.

The organization of an IW along these dimensions is specified in schemas expressed in an XML-based language; each workspace can thus be represented as a tree, where nodes define *composition elements* and leaves present the actual *content* users can interact with, and the visual template adopted for its visualization. In the definition reported above, distribution refers only to the component-based nature of the IWs, retrieving data from distributed resources. However we also show how introducing collaborative mechanisms leads to distributed execution of IWs along different application instances, each instantiating the whole schema or only portions of it, depending on the users' access rights, the collaboration needs and the workspace sharing settings.

The collaboration dimension complements the previous IW definition leading to the notion of *Distributed Interactive Workspaces* (DIWs). Hence, a collaborative process for the creation of DIWs first of all requires that users share a same application schema, each according to specific access rights. Based on this, collaboration then consists in the production of additional information, associated with some node or leaf in this tree, by users performing collaborative interventions.

A *collaborative intervention* on a document representing a DIW consists in the creation of a collection of additional elements, together with a mapping describing the relation of each such element to the original document. The additional elements can refer to the original document as a whole, or to any subtree or leaf in its composition. The structure of each additional element is defined by a schema supporting at least the following data [10]:

- *author*, as identified during the interaction;
- *timestamp*, indicating when the new element is committed to the DIW repository;
- *source*, as identified by the interactive selection of the part of the document to which the element refers;
- actual *info* added by the author;
- *visibility*, either private, public, or group-based.

An interactive process can exploit specific tools to identify the source, typically text selection or sketching. In general arbitrary fragments of the original workspace, or sets of fragments across the tree structure, can be selected and associated with an additional element. For example one can draw a shape to identify an area of a picture, or select several areas and collectively refer to them [11].

While interacting with the workspace, users can perform three types of interventions: *annotation, live editing*, *freezing & aggregation*, distinguished according to the nature of the info descriptor and to their usage within the platform.

With annotation, the *info* descriptor is of an arbitrary nature and can consist of any kind of digital data. The info added in an annotation can be used to generate an independent document, without corrupting the original workspace. The new document can be annotated in turn, thus supporting forms of *asynchronous collaboration*, for example by constructing annotation threads. Moreover annotations can be used to request modifications of parts of the workspace itself, delegating the realization of the request to authorized users, i.e., users with visibility on the annotation. In particular every user can access his or her private annotations and all public annotations, while users belonging to some group can access annotations posted to that group. In general, the annotation process enriches the

DOM of the loaded document with specific tags in correspondence of user selections. These tags are then saved with reference to that DOM. When a document on which some annotation is performed is loaded again, all the tags for the corresponding DOM are also loaded and used to create and render its enriched version.

With live editing, the *info* descriptor specifies a set of modifications to some structural element included in the workspace schema (e.g., a component or its underlying services in CmM, an integration query in CnM, the visual template). Such modifications are not only immediately activated on the instance in use by their author but also reflected the aspects of workspace composition and behavior shared with other users, defining a form of *synchronous collaboration*. Examples of live editing interventions are addition of a new component, or deletion of an existing one, or modification of the component query, resulting in an update of the content displayed in the component.

Finally *freezing & aggregation* refers to a process by which, during a specific user's interaction, snapshots of (fractions of) actual contents are captured and added to a list (an *aggregator*) of contents. The content of the *info* descriptor is the selected fraction of the content associated with the source at the time of the intervention. Users can build any number of aggregators and add any set of *frozen data* to each of them, by indicating the target aggregator at the time of freezing. The default visual template for aggregators iterates on the aggregator list to present frozen data, each according to its original format. All of these processes are enabled by special mechanisms through which actions on any instance of the DIW are captured and propagated, if needed, to the other active instances. As described in Section 4, this is made possible via an interaction between the client-side modules managing the composition and execution of each DIW instance and a server-side module managing persistency and evolution of the DIW schema.

As an example of remote collaboration by end users in the creation of a DIW, let us consider the following scenario. Mario is a high school student; his teacher assigns him and two of his classmates a homework for which they have to collect documents and multimedia resources. Mario opens the platform and, by clicking on the *share* button (see the right side of label 1 in Fig. 5), invites his friends Giuseppe and Alessandro to collaborate with him. After some minutes, Mario sees that his two friends are online. He types a message in the chat tool (4 in Fig. 5) to start collaborating. Both Giuseppe and Alessandro add a UI component to access a service. The live editing mechanisms allow Mario to understand what Giuseppe and Alessandro are doing by highlighting the border around the component: for example, an orange border highlights interventions from Giuseppe (he added a UI component for the Slideshare service), while a pink border indicates those from Alessandro (he added the UI component for the Vimeo service). Furthermore, the annotation feature of the platform allows each collaborator to attach a note to the DIW by clicking on the *note* button (at the right of label 1 in Fig. 5). For example Alessandro attaches a note to the Vimeo component (the balloon labeled with 2 in Fig. 5) to indicate that he would like to perform the union operation of Vimeo and Youtube, in order to retrieve more videos. A notification (the envelope indicated by 1 in Fig. 5) makes Mario aware of this new element. Mario can minimize the note, by clicking the minimize icon at the top-right corner of the balloon, or he can reply to it by clicking the reply icon. Mario and his friends go on adding other components to the DIW, and use them to retrieve content and multimedia resources. To save a specific result as "frozen content", Mario clicks on the *star* icon available at the top right corner of every visualized content. All these contents will be shown in a specific aggregation container to be shared with the teacher.

It is worth noticing that the combination of live editing and annotation poses some specific issues, concerning the problem of orphan annotations, i.e., annotations referring to content items, which are no longer present in the document. While in principle some mechanism could be devised to retrieve content simply moved to a different position [12] we adopt here a conservative stance,

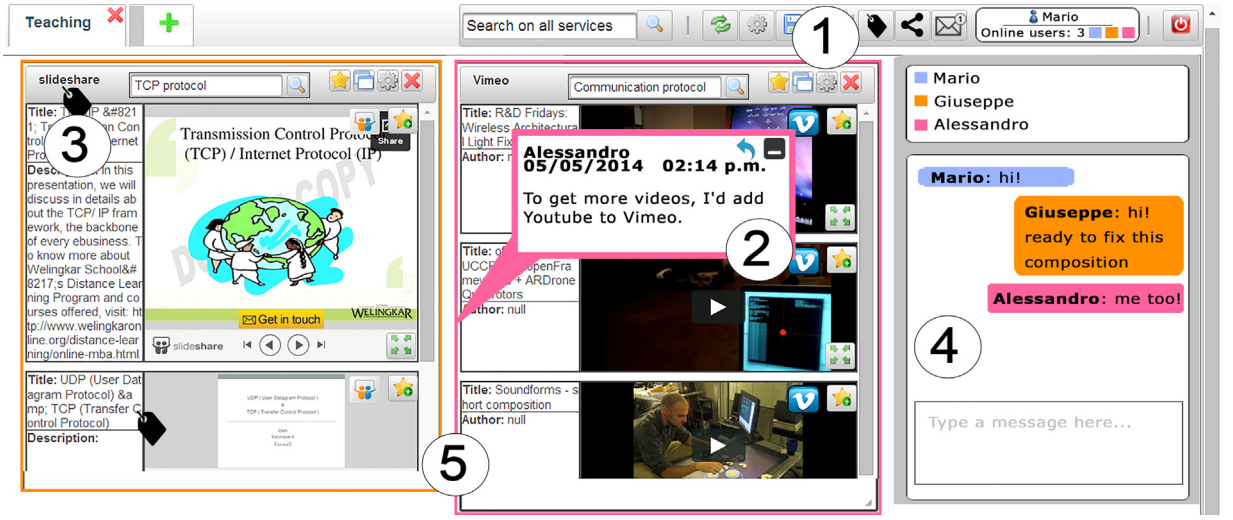


**Fig. 5.** Example of DIW, highlighting features for asynchronous and synchronous collaboration.

removing any annotation referring to a node in a subtree eliminated by a live editing process. This choice concerns annotations on service composition and visual template models, while those on content provided by a service can be retrieved if the same service offers the same content at some subsequent request, or if the content has been frozen to some aggregator.

## 4. Architecture of the DIW platform

Fig. 6 illustrates the organization of the platform supporting the composition paradigm and the collaboration interventions illustrated in the previous sections. The definition of a DIW is performed through the *Workspace Composition Environment*, an HTML/JavaScript Web application where end users can execute the composition actions and immediately see the result, i.e., a running application, thanks to the adoption of a live programming paradigm. In particular, a *Workspace Manager* on the client-side intercepts the visual mapping and synchronization actions performed by an end user. Through its *Schema Manager* module such actions are automatically translated into elements of a *Workspace schema*, expressed in an XML-based domain specific language [6], which describes the service queries, the association of the query results with specific visual templates, and possible synchronizations among different visual templates.

In order to support the live programming paradigm, as soon as new elements are added into the XML schema the workspace is immediately updated to show the changes, thus making the interactive definition and execution of DIWs possible. Therefore, the workspace manager is in charge of:

- querying services dynamically (through the *Service Manager* module), according to the queries defined in the workspace schema. Pre-defined service URIs and query parameters are specified in service descriptors stored in proper repositories;
- supporting visual refinement of service queries, as end users can define new selection and projection queries over the result set from a UI component service, as well as new union and join queries to integrate data of additional services;
- displaying dynamically the retrieved result set in a visual format, according to the visual mapping defined by the users and expressed in the *Workspace schema*.

Execution of a DIW can occur in the very device where the composition is created, as well as in *Execution Environments* running on different devices. As presented in Fig. 6 executing a DIW indeed simply requires an Execution Engine (which can be coded according to any Web or device-native technology), able to interpret the Workspace schema (*Schema Interpreter*) and instantiate the adopted visual templates (*UI controller*) by rendering the corresponding UI and filling the visual elements with data requested to the involved services (*Service Querying* module).

The architecture of the original composition platform hosts all the modules for DIW composition and execution at the client side. In order to preserve such a "lightweight" approach, in the extended platform the pure composition logic still resides at the client side and the creation of workspace schemas is still operated on the clients. However, as highlighted in Fig. 6, a server-side *Collaboration Layer*, together with additional client-side modules, takes care of persistence management and of co-evolution of schemas. The server also hosts the required repositories (for *Registered Services*, *Workspaces* and *Visual Templates*) and the database to store the data items required to manage synchronous and asynchronous communication.

Specific modules take care of the collaboration aspects. The *Sharing Server* manages resource sharing, by handling users' access rights and versioning of the released resources. This module also enables the distribution of portions of a same DIW on multiple devices. Based on the adopted sharing policies, the local execution engines instantiate and show only the part of the DIW a user or a device is authorized to use.

For live editing, every relevant modification on a DIW composition and execution is propagated to any other running instance of the same DIW. In particular the *Live Editing Client* (LEC) captures the elements describing the modification to the DIW structure and propagates them to

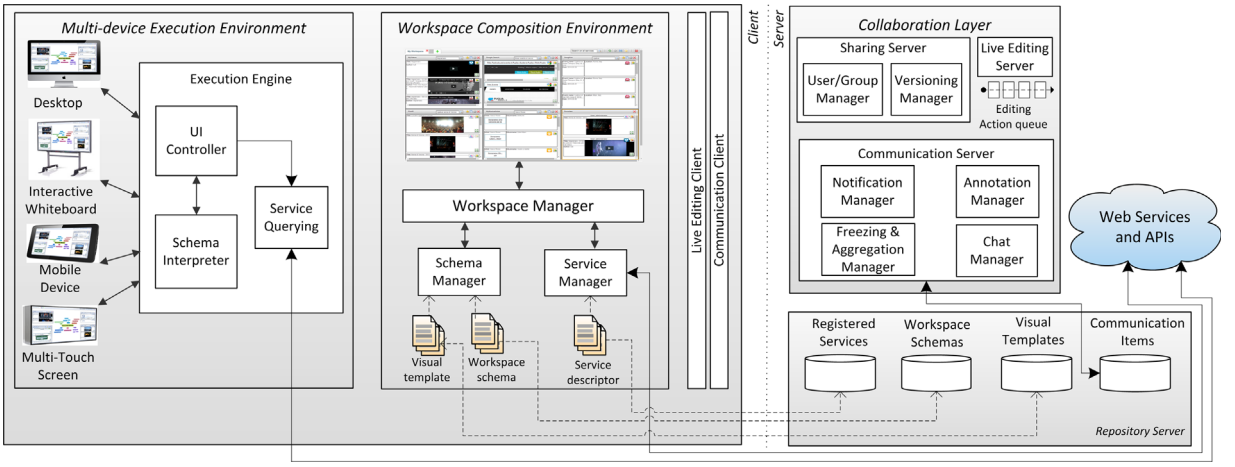


**Fig. 6.** Architecture of the platform for the collaborative creation and use of DIWs.

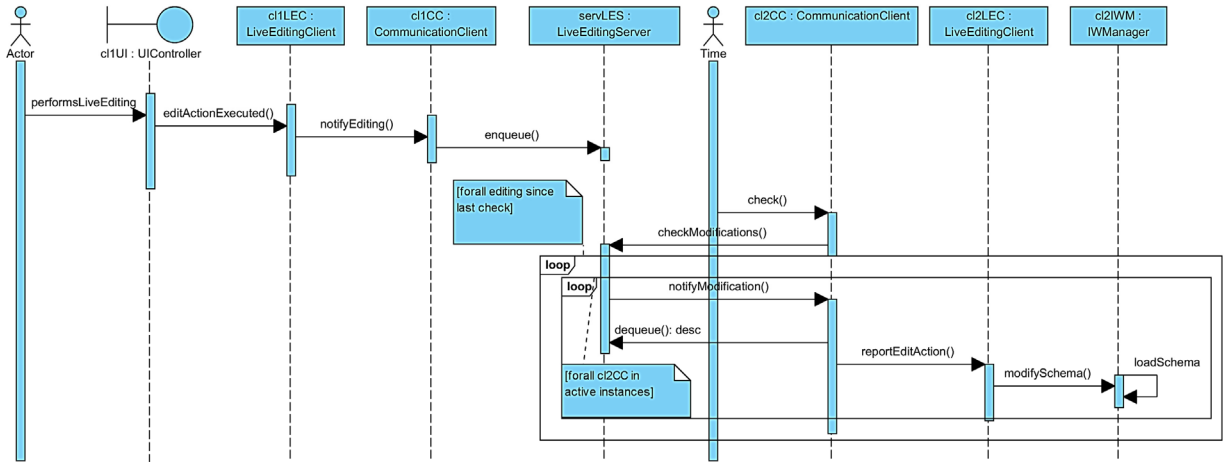


**Fig. 7.** Communication between instances of Live Editing Client and Live Editing Server for publishing (client cl1) and retrieving (client cl2) edit actions.

the *Live Editing Server* (LES), which takes care of the DIW schema evolution by maintaining a representation of the distributed editing actions. Every editing session on a given DIW has an associated *Editing Action queue*. Any active instance of the DIW periodically queries the LES to know whether new actions, generated by other DIW executions, are available in the queue.

Any live editing action propagated to the clients is represented as a pair `<modifiedObject, notification>`. The first element represents the argument of the action (e.g., a component, a service binding, an inter-component coupling, a query parameter) and all its properties. The second represents some metadata (e.g., the ID of the user who performed the action), needed for notifying the change. The *modifiedObject* properties have effect on the composition schema; the *notification* element is used by the LEC to visually highlight the action (e.g., highlighting the border of a UI Component as illustrated in Fig. 5). The LEC thus interprets the received actions and triggers events to let the *Workspace Manager* modify the composition schema accordingly and reload it. Changes are highlighted in the DIW based on the *notification* meta-data. To reflect DIW changes with minimal delay the LEC periodically checks the composition status representation through the LES, to verify whether some actions must be loaded and rendered within the DIW instance. The sequence diagram in Fig. 7 illustrates communication between LEC and LES for publishing (by client `cl1`) and retrieving (by client cl2) edit actions.

This form of synchronization of all the active DIW instances implies a "distributed" representation of the DIW schema, maintained at each client. Server-side management of the action queue ensures synchronized evolution of all the active DIW instances. Differently from the composition model in the original platform, the DIW is now *stateful*. The schema is enriched with state meta-data (e.g., parameter values to query single components, items selected in a data set) to synchronize each DIW instance not only on the composition structure (components and bindings), but also with respect to the displayed data set. Hence, composition is now *long-lasting*, maintaining structure and state across different sessions.

Interactions between client and server modules and schema persistence and evolution are also needed for enabling both synchronous and asynchronous communication among stakeholders in the form of annotations, frozen data, and messages exchanged via chat sessions. With respect to live editing modules the communication server and the communication client manage the addition of collaboration-oriented information to the original workspace, which we call *Communication Items*. Communication items are persisted by different modules (the *Annotation Manager*, the *Freezing & Aggregation Manager*, the *Chat Manager*), and are retrievable by authorized users when uploading the original document.

Such communication items can also be presented to users in a live form. To this end, an instance of the DIW can inquire with the server if there are annotations for some of the nodes currently present in the composition instance whose timestamp is more recent than the last check for annotations. The way each communication item is displayed then depends on the nature of the item: annotations are displayed in popup windows attached to the workspace elements they refer to; frozen data are presented in an aggregation component in charge of grouping all such items collected by the different users sharing the workspace; chat messages are displayed in an ad-hoc viewer for chats.

## 5. Related work and discussion

While collaboration is a mature research field in some communities such as the one working on Computer Supported Cooperative Work, it is not still quite explored in the creation of Web artifacts, especially Web mashups. Tools have been proposed to ease mashup development for unskilled users unable to program the component integration logic [13,14]. By offering intuitive visual notations to substitute programming, these tools reached the goal of enabling end user development of mashups. However, most of them offer paradigms for the creation of single-user applications, while they do not support the co-creation of shared information workspaces.

In the context of Web-based collaborative learning, Web Space Configuration is introduced in [15] as a basic container for instantiating W3C Widgets. Composition of widgets is independent of the runtime environment. Independence is exploited to support portability of the created applications, and sharing via broadcasting and co-editing. These are achieved by establishing a long-lasting connection by the owner of a Web space, which invites other users to join and see the Web space (broadcasting) and to apply changes (co-editing). Our approach also exploits independence from the runtime environment to enhance portability of the created interactive workspaces on different devices. The collaboration paradigm we propose also covers sharing and live co-editing. While Web space co-editing only focuses on the presence awareness aspect, we also support action awareness, by propagating and notifying in real time any change applied by one of the collaborating users on a shared information space. In addition, we support synchronous and asynchronous communication through chat, annotation, and frozen data mechanisms. These dimensions are not covered by the Web Space Configuration approach. We also believe that aggregator components for freezing data items are original in the mashup world. In this field, indeed, applications are fully dynamic, meaning that they retrieve data via instant queries to the involved services. The field studies revealed that storing single specific data items is a recurrent need of real users.

In [16], the authors propose a crowdsourcing paradigm where user participation is adopted as a solution to responsive design in Web application development, trying to collectively solve problems related to the adaptation of Web applications to different device screens. System developers provide an interface where adaptive features can evolve at runtime with the help of users, who can refine the adaptations to better match peculiar usage context. This paradigm opens Web development towards the social dimension. However, its aim is limited to letting users customize the presentation of their Web applications to best fit their current device, while it neglects collaboration and coordination of different stakeholders. Moreover, it focuses on presentation adaptation, not covering modification of the application content at all. The approach presented in this paper is instead specifically targeted towards handling content. In [17] the same authors then discuss how to distribute the execution of mashups along different devices. However, they do not deal at all with synchronous and asynchronous collaboration.

In [7] a generic awareness infrastructure is proposed for providing basic awareness services reusable throughout different platforms. Awareness support is anchored at a standardized layer to provide an application agnostic solution. Different collaborating clients include a component, the *generic awareness adapter* that embeds awareness widgets and is devoted to propagating contextual information (from the client to the server and vice–versa). The approach is especially interesting because of its portability across different platforms and its intrinsic extensibility, being based on the integration of widgets managing the different collaboration aspects. However, it only manages awareness in live-editing sessions, while our approach also covers asynchronous communication, shifting the focus from applications with duration limited to single sessions to long-lasting applications that support knowledge evolution and consolidation. Although the collaborative paradigm proposed in this paper is based on ad-hoc extensions implemented in the platform for interactive workspace composition, the architecture extensions have been designed as components, detached by the composition editor, with an adaptable event-driven logic. The collaboration components can be thus easily plugged (or unplugged), and adapted to other collaboration environments by customizing the events exchanged between client and server.

## 6. Conclusions

As users become increasingly familiar with Web 2.0 mechanisms to exchange ideas and instantly communicate with peers, collaboration becomes a fundamental feature of modern Web-based applications. However, service-based Web composition environments offering this feature are still lacking. This paper tries to fill this gap by showing how services, service-based resources, and composition models can be considered objects of collaboration. The presented collaborative features emerged from a series of studies where real users expressed their desiderata on possible collaborative aspects of a mashup platform. Besides operations such as Create, Read, Update, Delete applied to UI components and workspaces, already offered by the previous version of the platform, the collaboration extensions now support resource publishing and versioning on the common platform repository, and the definition of associated access rights for the other stakeholders. By proper setting of sharing policies, users can choose to distribute the entire DIW or part of it on multiple devices and among different peers. This feature is particularly useful in scenarios where each collaborating user contributes with single components to the creation of a shared workspace. Each single actor can manage an arbitrary complex workspace, but share with the others only some specific components or also some specific content items feeding shared aggregator components. We were also able to identify new forms of communication, for example through freezing ad aggregation of single content items. This is a novel characteristic, peculiar for mashups but at the same time scarcely investigated in the mashup world, which could be easily extended to different classes of content-intensive collaborative systems.

The platform prototype described in Section 3 was created on the basis of the design workshop involving end users. Formative evaluation of this prototype, performed with a thinking aloud test in laboratory, indicated that the users appreciated it. Future work aims at further validating the extended composition approach, to assess in the field the effectiveness of the intermixing of service-based interactive composition and collaboration features. Therefore, new user-based studies have been planned in the same usage contexts where the field studies reported in this paper were conducted. Efforts will be also devoted to design and develop new versions of the composition

environment that can be run not only in the Web browser but also as native mobile applications.

## Acknowledgment

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”. We are grateful to all people involved in the reported studies.

## References

[1] J. Yu, B. Benatallah, R. Saint-Paul, F. Casati, F. Daniel, M. Matera, A framework for rapid integration of presentation components, in: Proceedings of WWW '07, 2007, ACM, pp. 923–932.

[2] J. Yu, B. Benatallah, F. Casati, F. Daniel, Understanding mashup development, IEEE Internet Compost 12 (5) (2008) 44–52.

[3] N. Zang, M.B. Rosson, What's in a mashup? And why? Studying the perceptions of web-active end users, in: Proceedings of VLHCC '08, 2008, IEEE Computer Society, pp. 31–38.

[4] C. Ardito, M.F. Costabile, G. Desolda, R. Lanzilotti, M. Matera, A. Piccinno, M. Picozzi, User-driven visual composition of service-based interactive spaces, J. Vis. Languages Comput. 25 (4) (2014) 278–296.

[5] C. Ardito, P. Bottoni, M.F. Costabile, G. Desolda, M. Matera, A. Piccinno, M. Picozzi, Enabling end users to create, annotate and share personal information spaces, Springer, Berlin/Heidelberg, 2013, 40–55 (In: End-User Development-Is-EUD 2013, LNCS 7897).

[6] C. Cappiello, M. Matera, M. Picozzi, End-User Development of Mobile Mashups, Springer, Berlin/Heidelberg, 2013, 641–650 (Design, User Experience, and Usability. in: Web, Mobile, and Product Design – HCII '13, LNCS 8015).

[7] M. Heinrich, F. Grüneberger, T. Springer, M. Gaedke, Reusable awareness widgets for collaborative web applications – a non-invasive approach, Springer, Berlin/Heidelberg, 2012, 1–15 (in: Web Engineering – ICWE ‘12, LNCS 7387).

[8] C. Cappiello, F. Daniel, M. Matera, M. Picozzi, M. Weiss, Enabling end user development through mashups: requirements, abstractions and innovation toolkits, Springer, Berlin/Heidelberg, 2011, 9–24 (in: End-User Development – Is-EUD 2011, LNCS 6654).

[9] C. Ardito, M.F. Costabile, G. Desolda, R. Lanzilotti and M. Matera, 2014, Creating Flexible Interactive Workspaces through Data Source Composition, IVU Lab-Technical report-No.02-2014, avaliable at ⟨http://ivu.di.uniba.it/papers/2014/IVU_TR_2-2014.pdf⟩.

[10] P. Bottoni, S. Levialdi, N. Pambuffetti, E. Panizzi, R. Trinchese, Storing and retrieving multimedia web notes, Int. J. Comput. Sci. Eng. 2 (no. 5/6) (2006) 341–358.

[11] M. Addisu, D. Avola, P. Bianchi, P. Bottoni, S. Levialdi, E. Panizzi, Annotating Significant Relations on Multimedia Web Documents, John Wiley & Sons, Inc., 2012, 401–417 (Multimedia Information Extraction).

[12] P. Bottoni, A. Cotroneo, M. Cuomo, S. Levialdi, E. Panizzi, M. Passavanti, R. Trinchese, Facilitating interaction and retrieval for annotated documents, Int. J. Comput. Sci. Eng. 5 (no. 3/4) (2010) 197–206.

[13] S. Aghaee, M. Nowak, C. Pautasso, Reusable decision space for mashup tool design, ACM, 2012, 211–220 (In: Proceedings of EICS '12).

[14] A. Namoun, T. Nestler, A. De Angeli, Service Composition for Non-programmers: prospects, Problems, and Design Recommendations, IEEE Computer Society, 2010, 123–130 (In: Proceedings of ECOWS '10).

[15] S. Sire, E. Bogdanov, M. Palmér and D. Gillet, Towards Collaborative Portable Web Spaces, In: Proceedings of *MUPPLE '09*, 2009.

[16] M. Nebeling, S. Leone, M. Norrie, Crowdsourced Web Engineering and Design, Springer, Berlin/Heidelberg, 2012, 31–45 (In: Web Engineering-ICWE '12, LNCS 7387).

[17] M. Husmann, M. Nebeling, M. Norrie, MultiMasher: a Visual Tool for Multi-device Mashups, Springer, Switzerland, 2013, 27–38 (In: Proceedings of Current Trends in Web Engineering-ICWE '13 Workshops, LNCS 8295).