User-Driven Visual Composition of Service-Based Interactive Spaces

Carmelo Ardito[1], Maria Francesca Costabile[1], Giuseppe Desolda[1]
Maristella Matera[2], Antonio Piccinno[1], Matteo Picozzi[2]

[1]Università di Bari Aldo Moro
Dipartimento di Informatica
via Orabona 4, 70125 Bari, Italy
{carmelo.ardito, maria.costabile, giuseppe.desolda, antonio.piccinno}@uniba.it

[2]Politecnico di Milano
Dipartimento di Elettronica e Informazione
Via Ponzio 34/5, 20133 Milano, Italy
{matera, picozzi}@elet.polimi.it

Abstract

Objective
The idea is to replace fixed, pre-packaged applications with flexible composition environments that, thanks to a separation among data, functions and presentations, make interactive environments "emerge" at run-time based on composition actions performed by end users.

Methods
This paper describes an approach for the lightweight construction of integrated, situational work-spaces pervasively accessible and sharable through a variety of devices. Specific emphasis is posed on the adoption of a composition paradigm that abstract from technical details and can be thus used by non-technical users. Thanks to the separation of concerns on which we ground the composition paradigm, the overall approach and its enabling platform are also amenable to customization with respect to the requirements of specific domains.

Results
A platform, based on service composition technologies, has been developed; it implements a meta-design approach to allow end users, not necessarily experts of technologies, to extract contents from heterogeneous sources and compose Personal Information Spaces (PISs) that satisfy their information needs and that can be pervasively executed on different devices.

Conclusion
We proposed an approach where the composition platform enables the extraction of content from heterogeneous services, and the integration of the extracted content into situational applications where content presentation is flexibly managed through different visualization templates.

Practice
The developed prototypes have been evaluated in studies in which real users (e.g. guides of an archaeological park) were observed in action.

Implications
The results provided hints for refining the prototypes, and pose the basis for future work related to the identification of design principles that can make composition technologies more useful and usable for the end users.

Keywords: End-User Development, Service Composition, Personal Information Space, Platform Independent Models

1. Introduction

Different work contexts and every-day life situations are nowadays characterized by activities where single users or groups of people, through different devices, browse heterogeneous content, capture, synthesize and annotate it to highlight insights, compose it in various ways in order to create new content and services. The huge amount of resources available on the Web provide a valuable source of content; but to enable an increasing numbers of people to make sense of such resources it is necessary to open up the construction of service-based software to non-programmers. This can be achieved by applying the lesson learnt in the last years on End-User Development (EUD) [1-3], to empower people, who may not have technical skills, with the possibility to compose interesting content, made available by services, to fulfill their situational needs.

Technologies for Web services composition have been proposed since the 90's in the context of the Service Oriented Architecture (SOA). More recently, we have been assisting to the proposal of platforms, based on *mashup technologies*, which claim to be more oriented towards end users. However, this claim revealed unrealistic because of the inadequacy, for end users, of the composition languages such platforms are based on [4-6]. Due to the strong need to empower people to compose their personal information spaces (PISs), which could improve their knowledge, productivity and collaboration behavior, service software construction has to be based on a composition paradigm suitable for end users. This implies that generality has to be relaxed to some extent while investigating how domain-dependent languages and tools can be effectively used by people to shape up software artifacts responding to the long tail of their varying needs. In order to identify adequate languages and tools, it is necessary to get a deep understanding of end users' social practices, considering all stakeholders' points of view and competences.

The overall goal of our current research is to investigate models, technologies and architectures for supporting people, who are not software developers and have diverse needs, to co-create personal information spaces (PISs) by integrating heterogeneous contents and artifacts. This is in line with the interest, growing both in academia and in industry, on elastic systems where applications can be flexibly shaped up at different layers (data, functions and presentation) at use time. Moreover, "*content, application, and device need to be decoupled as much as possible to allow users to focus on information without being confined to a particular pre-packaged application context*" [7].

The work presented in this paper addresses the above-mentioned issues of user-driven composition and proposes a platform providing visual paradigms for the lightweight construction of multi-device, integrated workspaces able to satisfy situational needs. Thanks to EUD and meta-design techniques [8, 9], the platform is able to support the different stakeholders involved in the creation of PISs. The work builds on the last years' experience in analyzing different paradigms for mashup composition and experimenting prototypes of a "generic" mashup platform [10, 11]. We are currently investigating how the platform can support information flows among the different stakeholders in collaborative practices, for example, in the context of e-health where PIS creation would facilitate communication and collaboration of different figures, e.g. physicians, caregivers, patients and their family, in the care of pathologies that last for years, like epilepsy or dementia. We also experienced the adoption of the platform to the Cultural Heritage domain. This paper in particular refers to a case study that shows how our platform can bring practical value to different stakeholders in the context of visits to sites of cultural interest, like archaeological parks.

The paper is organized as follows. Section 2 discusses the rationale of our work and Section 3 reports related work. Section 4 illustrates the meta-design approach to PIS composition. Section 5 describes the developed platform, addressing its main components and showing, with examples taken from the case study, the adopted visual interaction paradigms and how the platform is customizable to specific application domains. Section 6 reports the studies performed to validate the developed prototypes, and Section 7 concludes the paper.

2. Rationale

The emerging need of making software systems flexible, so that to increase their ability to support a large variety of tasks, is highlighted in recent works published in the literature. The idea is to replace fixed applications with elastic systems, where contents, functionality and access devices are totally decoupled from specific contexts of use and, as such, are able to accommodate multiple and variable contextual needs. New design principles are therefore emerging under the name of *transformative user experience*, to promote paradigms in which end users can access contents but also flexibly use such contents in several situations and across several applications [7]. In other words, content should be decoupled from containers and accessible independently of a specific application. The idea is to capitalize on a set of repeatable system behaviors and design principles to build interactive systems that can be fluidly shaped up by users to make them suitable with respect to the users' task.

This goes beyond current approaches for Web mashup development [11-14], for example those based on the synchronization at the presentation layer of Web APIs that embed content, functions and presentation. Alternative paradigms are instead needed to decouple contents, functions and devices, thus allowing users to focus on information without being confined to a particular prepackaged presentation, functionality or application context. In this new vision, which is gaining momentum in both research and industry contexts, the application logics for accessing and managing contents and functionality, together with the user interface, e.g. the different ways of presenting results, actually emerge at run time from the actions of users. This requires new principles on how data objects (content) interact with their contexts and how this flexibility can be leveraged to enable end users to compose and transform content as required from the current situations. It also demands new models and techniques for content extraction, integration and reuse, since content and functionality are not fixed ingredients as in traditionally designed application.

This trend is also reflected in the emergence of new research lines within HCI, which focus on *appropriation*, *End-User Development*, and *meta-design*. *Appropriation* is the capability of a system to be valid beyond a core set of use cases, being able to get adapted even to unexpected uses by the end users (e.g. see [15]). *End-User Development* (EUD) refers to the involvement of end users in the software development process, in order to modify, and even create, software artifacts [1-3]. EUD activities may go from simple parameter setting to integration of pre-packaged components, up to extending the system by developing new components. The design of systems that enable EUD activities requires a shift in the design paradigm, which must move from user-centered and participatory design to *meta-design*, which literally means "design for designers" [8, 9]. This new paradigm allows various stakeholders, including end users, to act as co-designers; thus, software engineers do not design the final application, as in traditional design, but they create software environments through which different stakeholders can contribute to the design of the final application. Meta-design is characterized by two main phases. The first phase consists in creating the design environments that allow system stakeholders to participate in the design (meta-design phase). The second phase consists in the design of the final applications, carried out as joint work of the various stakeholders, who collaborate through their design environments (design phase) [16]. Thus, professional designers (software engineers) face new challenges since they have to create software environments that can empower end users to shape the software they use, without obliging them to become programmers.

Our approach addresses all such lines of action, being it principally aimed at empowering people to create flexibly their PISs with tools that can let them actively and flexibly compose contents and functionalities. This is also in line with the so-called culture of participation [17, 18], that promotes a shift from consumer cultures, where produced artifacts are passively consumed, to participatory approaches that greatly exploit computational media to support collaboration and communication, providing users with the means to become co-creators of new ideas, knowledge, and products [19]. Thus, we are assisting to a redefinition (and also to a seamless fusion) of roles that go beyond the conventional user-designer dichotomy, in a context where system design and system execution are interwoven to let users create, immediately execute and iteratively evolve their own applications.

3. Related work

The term Personal Information Space (PIS) was used in [20] to define an informal space that consists of both artifacts and assigned meanings constructed, interpreted and manipulated by only one person. In the context of our research, we define PISs as personalized service-based interactive environments, where people, who are not software developers, *integrate* and *share* heterogeneous contents and artifacts coming from different sources available on the Web. With respect to the definition reported in [20], our research stresses the collaborative situations where PISs enable communication flows and information sharing among different stakeholders. Therefore, our definition of PIS is closer to the one of *Common Information Space* (CIS) introduced in [21] in relation to collaborative activities.

One aspect of our work is the deployment of PISs on different devices. Some works in the literature have addressed the problem of how to access information and services available on the Web through several kinds of devices, e.g. laptops, palmtops and cellular phones. Managing such a diversity of devices is not costless for service providers; they should develop many different user interfaces, one for each kind of device and context of use. An alternative approach is developing flexible user interfaces that can be adapted to provide the same information and services regardless the used device. Some proposals to the adaptation of Web applications, especially to mobile devices, are based on the use of ontology-based annotations [22] or model-based representations [23, 24]. The aim of model-based user interface development is to adopt declarative models to be used by model-based user interface design environments to build the final user interface [25]. This is usually pursued by specifying an abstract representation of a user interface, .i.e., a representation not linked to a specific interaction modality (visual, aural, tactile, etc.) neither to a specific deployment. The user interface is then adapted, in a completely or partially automated way, to the multiple devices and/or contexts of use (e.g. [26-28]). We will show in Section 5 that the platform we have developed provides service management experts, who work in collaboration with domain experts, with a visual composition environment, where they can select a visual template for visualizing the data gathered by the services, and associate attributes of a service into visual elements of the visual template. This environment automatically translates the user actions in a set of resource schemas codified in XML, which specify how the integrated data have to be displayed. The adaptation to the rendering capabilities of each single device is possible thanks to dedicated execution environments, while the schema generated through the composition environment is abstract and independent of any specific platform.

In the rest of this section, we contextualize our approach for PIS construction with respect to some works that address aspects related to the composition of service-based interactive spaces. Indeed, the last few years have seen an evolution in the way information-seeking applications are constructed and deliver responses to people. The ever-increasing availability of Web data sources, the huge amount of user-generated content (e.g. reviews, opinions, ratings, tags), the emergence of cloud computing, the new service composition paradigms and the spread of ubiquitous systems are reshaping the way information is created, accessed and consumed. The emerging trend is to let people create new value by integrating resources available on the Web to fit their contextual needs.

Integration technologies (workflows, service composition, data integration) have been around for the last two decades. More recently, Web mashup methods and composition technologies has been proposed for the creation of Web applications starting from reusable Web resources. Web mashups are indeed composite applications, where the “components” are as heterogeneous as SOAP/WSDL Web services, RESTful Web services, RSS/Atom feeds, JavaScript libraries, or simply content extracted (wrapped) from common HTML Web pages (and many more). More recently, W3C Widgets have also become the object of mashup composition [29]. Widgets are the result of the (still ongoing) W3C effort to standardize the development UI components, i.e. full-edged mini-applications offering functionality and data coming from third party resources, that can be exploited within a Web page.

Service composition has been traditionally covered by powerful standards and technologies (such as BPEL), which however can be mastered by information technology experts only [30]. What makes mashups different from plain Web service composition is their potential as tools through which end users, not necessarily skilled in computer science, are empowered to develop their own applications. Mashups are indeed emerging as an alternative solution that can help realize the dream of a programmable Web even by non-programmer users [31]. This is due to the emphasis that such composition technology poses on the integration at the presentation layer, focusing on the synchronization of service user interface to create rich interactive Web applications. This aspect indeed has not been adequately investigated so far in more traditional fields of service and application integration.

However, the mashup potential as technology targeting the end users is still rarely exploited. So far the research on mashups has focused on enabling technologies and standards, with little attention on easing the mashup development process - in many cases mashup creation still involves the manual programming of the service integration. Some recent user-centric studies also found that, although some prominent platforms (e.g. Yahoo!Pipes) simplify the mashup development, they are still difficult to use by non-technical users [5]. According to the End User Development (EUD) vision, enabling a larger class of users to create their own applications requires intuitive abstractions and easy development tools, and a high level of assistance. 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 [32, 33]. 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 [14, 34-37]. 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, as it happens for example for Intel Mash Maker [34], 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) [5].

There are also new composition practices and related execution environments that are emerging around the W3C Widgets. Dedicated execution platforms, the so-called *Widget portals* or *Widget containers* [38], allow users to group within a page miscellaneous pieces of information. Some works have also investigated the use of such technologies for the creation of PIS in specific domains, such as education [39, 40]. However, while these platforms make it easy the deployment and execution of widgets, they still do not enable the fusion of content and UIs within unified interactive spaces: the resulting Web pages jut includes collections of not coordinated contents or functions made available by each single widget, and the customization by the end users of such pages consists in the selection of the widgets they are more interested in. Few preliminary research works are addressing the integration aspects [41], but the literature still lacks convincing results.

Another relevant point to be considered is that all the composition platforms so far proposed tried to be generic (domain-independent), to increase their validity across different domains. This is however a lack more than a strength: limiting the possibility to customize both the platform functionality and the composition language becomes a barrier for the adoption of such platforms as tools for enhancing user productivity [6].

In light of the previous observations, our work tries to combine EUD principles with the potential of composition models and technologies. The aim is to support flexible and varying needs, by enabling end-users to construct their applications supporting their activities, through intuitive, easy-to-use composition mechanisms, which can be adapted to specific requirements of the addressed end user communities. We therefore start from a generic composition platform, which is flexible enough to be adopted in different contexts of use, and we adapt it with respect to the requirements identified within the specific communities of users.

4. Meta-design approach to PIS creation

Our proposal of visual languages and tools for the lightweight composition of PIS is contextualized within a meta-design approach based on the Software Shaping Workshop (SSW) model we have developed in the last years [9, 42]. SSW underlines the creation of software infrastructures that support EUD activities and knowledge co-creation among the different stakeholders involved in a design team. All the stakeholders of an interactive system, including end users, are "owners" of a part of the problem: software engineers know the technology, Human-Computer Interaction (HCI) experts know human factors, graphic designers know how to create an appealing graphical design, end users know the application domain. In order to contribute to system design by bringing their own expertise, all these figures need different software environments, specific to their culture and skills. The professional developers involved in traditional design constitute here the team of meta-designers, who creates software environments, which we call Software Shaping Workshops (in short SSWs or workshops) through which the other stakeholders, acting as designers, can be creative and can adapt the software to fit their specific needs. Through their own SSW, these stakeholders contribute to shape software artifacts, i.e. they create and modify elements (objects, functions, user interface widgets, etc.) of the system of interest, and exchange the results of their activities to converge to a common design. In a similar way, various communities of stakeholders are involved in the different phases of the PIS life cycle.

Figure 1. The meta-design approach to PIS creation.
The bottom layer outlines the environments for end users, the middle layer the environments for some experts, the top layer the environments for professional developers (meta-designers).

The meta-design approach depicted in Figure 1 refers to the creation of PISs with reference to the case study on the visits to archaeological parks. In the last year, we have performed systematic contextual enquiries [43], observing the work of professional guides conducting visits in the archaeological parks of the Apulia region, in Southern Italy. It emerged that guides could benefit from using personal information spaces, in which they could organize multimedia material to be shown in different phases of the visit through different devices. For example, large interactive displays to interact with the PIS could be available at the beginning of the visit in the hall of the museum associated to the park or in a room in order to support the introduction to the visit. A tablet would be used during the tour in the park for providing additional information, such as photos, videos, 3D reconstructions of ancient monuments, references to related sites, etc.

In this scenario, professional guides of the park are the main end users. They may create and access their PIS by using different software solutions running on different devices. Other end users could be the visitors, if the guides decide to share their PIS with them. As shown at the bottom level of Figure 1, our platform currently provides applications for desktop PC, tablet and large multi-touch display. Such applications have been customized to the end users' specific needs by other stakeholders, namely service management experts in collaboration with domain experts. As depicted in the middle level of Figure 1, their own software environment (or SSW) enables them to select the proper services among those available on the Web and register those one they find useful for the specific usage scenario required by the park guides. Sometimes it is necessary to integrate contents deriving from different services into unified views.

The environments for all the users of the platform have been created by the professional developers (meta-designers), who are also in charge of implementing and/or modifying some elements of the composition paradigm that require programming efforts (top level of Figure 1). For example, a basic element used by the composition paradigm is a *visual template* that, as we will better explain in the next section, represents a container that drives the integration of content, coming from any possible resource, into a PIS.

Figure 2. Architecture of the platform for PIS composition and use.

5. The platform for PIS composition and use

To support the service-based composition approach for the creation of pervasive PISs, we have developed a platform prototype based on a general-purpose mashup environment [10], in which the composition of interactive spaces exploits a "lightweight" paradigm for the integration of heterogeneous resources, mainly adopting visual mechanisms. By exploiting the same abstractions for mashup composition, we have equipped the platform with composition environments, customized to specific application domains, where the different stakeholders, depending on their role, create different types of service-based artifacts. The overall organization of the platform is illustrated in Figure 2. The resources used in the composition are based on heterogeneous services, which can be remote or locally managed within the platform. Such services constitute sources of contents that can be manipulated and integrated at different levels. To make this possible, an important component of the platform is the *Service Customization Environment*, devoted to service management experts, who, working with domain experts, select and package service-based resources that are adequate for specific composition scenarios in a certain application domain. Such resources embed the logic, defined by the experts, for querying services and visualizing the resulting data set. The descriptors of the so-created resources are made available to the *PIS Design Environment*, customized with respect to the characteristics of the target domain, through which end users are enabled to further manipulate the created resources and select the contents of interest to compose their

PIS. The PIS schema can be then deployed on different devices.

A feature of our approach is that the two environments, although offering different visual mechanisms for content and service composition, are based on a same conceptual model for resource integration, as well as on similar model-driven mechanisms for the creation of platform-independent schemas that can be executed on multiple devices. In the sequel, we illustrate the two composition environments, and highlight how the same abstractions guide the composition and the execution of service-based artifacts.

5.1. Service customization environment

The service customization environment offers support for querying REST services. The registration of such services is performed through visual forms. It produces *Service Descriptors* (see Figure 2), which specify basic properties and parameters to invoke the services, such as the service URI and the value of some search keys. The aim is to instrument a *Service Querying Module* with mechanisms to run service queries that retrieve an initial result set, which provides a basis for the service management experts to identify the type of data that can be retrieved through the service and its organization along different data attributes, i.e. the schema of the *Service Result Set*. The initial result set can then be refined to meet the needs of the end users. Data sets of different services can also be integrated. The result is the definition of schemas of *resources* that will available to end users for PIS composition.

Figure 3. The Service Customization Environment.

Figure 3 shows a screenshot of the service customization environment. Even though the experts using this environment are supposed to be familiar with service technologies, a "lightweight" paradigm, based on visual mechanisms, enables the resource packaging without any need to program or adopt complicated design notations. Figure 3 refers to a situation in which the expert has already defined the service descriptors of interest. The *data panel* on the left shows the initial result set retrieved by querying, for example, the Flickr service. Multiple services can be queried in parallel and their result sets are visualized in different tab windows. In the example of Figure 3, only one tab window is used to present the data set returned by querying the Flickr service using the keyword "Egnathia". It shows that the returned result set is composed of 465 items. For each result item some meta-data are also reported, such as an identification name (id), the owner id, the privacy settings, and the photo preview. The expert has now to define how the content returned by this service has to be visualized in a proper visualization container in the end user's PIS design environment. In our platform, a container is a *Visual Template*. In the example of Figure 3, the visual template chosen by the expert is a list-based template, i.e. a list of content items. Other visual templates currently available in our prototype exploit map-based and chart-based visualizations and address the visualization features of different devices. For example the list-based template in the example of Figure 3 is specialized for visualization on smart phones.

Using the selected visual template, the expert can *reduce* the service schema shown in the data panel by projecting only the attributes of interest. Attribute projection is expressed by moving attributes from the data panel onto some visual elements of the selected visual template, which we call *visual renderers* (*vr*s). The effect of this mapping is immediately shown in the visual panel: the visual template gets filled out with the content items returned by the service for the defined attribute projection. The expert can iteratively and interactively modify the attribute choice. In the example of Figure 3, the panel on the right shows the list-based template, in which each content item has to be displayed by means of an image and two text fields, one on top of the other. The expert is selecting, from the service schema, the attributes to be projected in this visual template. Beside the image, for the two text fields he selects "Title" and "Description" attributes. These are the visual renderers in this template. The projection is performed by dragging the attributes from the data panel onto the visual template panel, as indicated in Figure 3 by the arrow.

The effect of the visual mapping is the definition of a *binding* between a visual renderer and the projection query visually defined by the user (e.g. an XPath expression to navigate into the service result set). If the data associated to the same visual renderer are selected from multiple services, this visual mapping mechanism acts as a data mediation mechanism: the attribute selection defines the local schema of each involved service, while the visual template, and in particular the set of its visual renderers, provide the global schema the service reductions are mapped to. The result of this service customization session is a set of *resource schemas*, which specify how services have to be queried, how different reduced result sets have to be integrated, and how the integrated data have to be displayed in the design environment dedicated to end users. Such schemas are interpreted within the PIS design environment to instantiate the resource windows from which the end users select contents for PIS composition.

It is worth remarking that our platform is conceived for the integration of heterogeneous services, and it is not tied to any specific domain. Some recent studies on composition approaches show that too general platforms are not used with satisfaction by users, who instead prefer paradigms customized with respect to their specific needs [5, 6]. A strength of our platform is the possibility to be easily customized to a particular application domain and to the needs of specific communities of end users. This is achieved by means of i) the visual templates, defined at meta-design; ii) by the service customization environment, which enables service management experts and domain experts to select and register into the platform services and data sources (public or private) that can provide content able to fulfill relevant user information needs. As shown above, service customization is kept simple, being primarily based on direct manipulation of visual elements.

5.2. PIS design environment

The *PIS design environment* allows end users to compose the resources packaged by the experts; it offers visual mechanisms, through which end users synchronize contents with "container" visualizations.

Figure 4. PIS design environment running on a PC.

Figure 4 reports a screenshot of the PIS design environment customized for the Cultural Heritage scenario. The customization consisted, first of all, in packaging appropriate resources, which are here made available through some *resource windows*. Using such an environment, the guide of the archeological park (a female) can select relevant contents, e.g. 3D reconstructions from Google Sketchup, photos from FlickR, videos from Youtube, wiki pages from Wikipedia, maps from Google Maps. Since it is important for the guide to associate contents to specific locations in the park, the design environment adopts a map-based visual template as container visualization. The container visual template together with the service-based resources ad-hoc packaged by the service experts, thus constitute elements that address domain specificity.

By typing a keyword in a search box, the guide dynamically searches for contents in the available resource windows (i.e. within the content retrieved by the queries previously specified by the service experts when packaging the resources). In the example of Figure 4, she typed "Traiana" because she is interested in content related to the Trajan Way, an important Roman road in Egnathia, connecting Benevento to Brindisi. Within each resource window, content items are shown according to the visualization defined by the expert during the service customization. For example, on the left of Figure 4, the Wiki resource window shows title and short description of three content items: “La via Traiana”, “La via Traiana in Puglia” and “Via Egnazia”.The guide can then select and move contents of interest on specific position on the map in the main window. The call-out shown in the figure refers to a video about Egnathia that the guide is positioning at the top of the map.

This visual mapping action corresponds to the definition of a *binding* between an element of the visual template (e.g. a marker on the map characterized by geographical coordinates), and the identifier of a specific content instance to be visualized for that point. In other words, while in the service customization environment the visual mapping operate at the service schema level (intensional level), in the PIS design environment the visual mapping operates over specific instance (extensional level). In fact, as highlighted by some field studies in specific domains [11, 44], at this level it is important for the end users to dynamically query services, but it is also fundamental being able to “save” the instances included in the PIS, in order to retrieve the same content across different executions.

The visual composition actions that the end user performs through the design environment are captured and automatically translated into an XML-based, platform-independent model, which specifies the inclusion of and the bindings among the contents extracted by the different resources and the container visual template. The so-created composition model can be immediately interpreted and executed on the device where the composition is taking place. The PIS design environment indeed includes some modules for PIS execution in charge of querying services to retrieve the contents associated with the visual template and managing the visualization of such data. In this way, end users are able to interactively define and execute their PIS. It is worth noting that in all the domain-specific customizations of our generic mashup platform, the composition paradigm has always been characterized by the intermixing between design and execution of service-based artifacts: the users can define their compositions, immediately experience the effect of their composition actions, and iteratively and interactively refine the resulting applications [10][1].

When complete, the PIS schema can be saved in a remote *PIS schema repository*; end users can thus access their created models any time and deploy them on different devices. Indeed, the module for the

[1] In case of device limitations, or depending on the specific situation of use, the two phases can be of course detached. For example, for compositions to be executed on smart phones, it would be convenient to execute the composition design on a different device without screen limitations.

execution of PIS illustrated above can be run on different devices, e.g. as Web applications, or as native apps compliant with the technologies of the target devices. This is possible thanks to the model-driven logic behind our approach, which generates schemas that can be then interpreted and instantiated on different devices through lightweight execution engines complying with the device technology. Therefore, going back to our scenario, during the visit, the guide can interact with their PISs using an application on a multi-touch display, which supports the briefing phase before the tour. They can also use a tablet application during the tour through the remains to show multimedia contents. If during the visit the guide needs further contents, she queries the resources by typing a keyword in a search box. The retrieved contents can be accessed for further details and possibly added to the PIS, whose model is updated accordingly.

5.3. Composition Model

As already mentioned in the previous sections, in both the service customization and the PIS design environment the composition paradigm exploits *visual templates*, i.e. schemas of the final user interface of services or of PIS, that act as *containers* that the users can fill in with data extracted from services or resources [45]. Currently, our platform offers visual templates based on lists, maps and charts. In our case study, the services offered to the end users have been customized using a list-based template (see Section 4.1), while the environment offered to the guide uses a *map template*, which exploits the visualization offered by the Google map service to geo-localize data extracted from other resources. In fact, visual templates can be supported by Web services (e.g. maps and charts are supplied by the Google public APIs), or can be developed ad-hoc, as it happens for our list-based template.

Figure 5. An excerpt of an XML-based composition schema generated by the Service Customization Environment.

Figure 5 reports a fragment of the XML-based composition schema that is automatically created when the expert interacts with the Service Customization Environment. The schema fragment refers to a query to the YouTube API, whose results are displayed in a list. The way the API is queried and the retrieved contents are visualized is illustrated by means of the elements that we describe in the following.

For each data source involved into the composition, the parameters used to query the source at runtime are specified. The visual elements that compose the application user interface (UI) are then expressed**.** Such elements are abstract in that they correspond to visualization elements that are independent from any device and any visualization technology. We have called these elements *visual renderers* (*vr*), since their role is to render some data in the "concrete" UI. At execution time, visual renderers are sources of events (e.g. the selection of a data item), which then trigger the execution of queries on the underlying service and the display of the retrieved data through the visual render itself.

For each visual render, a data binding specifies the content to be visualized. The binding is expressed by the pair <*data source, xpath*>, representing the source from which data are extracted and the XPath query to extract the data from the source result set. Such bindings can also associate to a same visual element the queries issued to different services.

5.4. Execution Engine

The execution of the created composition schema requires a dedicated *execution engine*, i.e. a set of modules able to interpret the composition schema and to dynamically instantiate the corresponding application on the target device. The execution logic is the same for the instantiation of resources within the PIS Design Environment and for the execution of entire PISs on a certain device.

As reported in Figure 2, a *Schema Interpreter* is in charge of parsing the schema of a PIS. It then invokes the *UI Controller* that, based on the adopted visual template, dynamically generates the user interface. The UI controller also invokes the *Data Manager* module, which in turn queries the involved services based on the specification in the model of the user-defined bindings. The service responses (e.g. represented in JSON) are cached locally on the device. The UI controller finally manages the display of the retrieved data through the visual elements also specified in the bindings.

So far, we have developed execution environments for Web browsers, for a multi-touch platform and for Android mobile devices.

6. Validation of the developed platform

In this section, we describe the evaluations performed to validate the developed platform. In particular, Section 6.1 reports the evaluation of the service customization environment used by the experts for selecting, registering and composing the services, which will provide the contents to be included in the PIS created and used by end users by means of their platform environments. Section 6.2 summarizes the results of a formative evaluation that involved guides of archaeological parks. A field study was also carried observing how guides created a PIS and used it during a real guided tour of the archaeological park of Egnathia, in Southern Italy.

6.1. User evaluation of the service composition environment

The service customization environment addresses service management and domain experts who intend to prepare components for the end users. The user study specifically focused on the effectiveness and intuitiveness of the composition paradigm for component creation, trying to measure such factors in terms of user performance, ease of use and user satisfaction.

Participants. The study involved 10 participants (age 26-29) among the students of the Computer Engineering curriculum at the Politecnico di Milano. All the users had experience in programming ad some exposure to service composition.

Procedure. The participants were tested one at a time in a quiet research laboratory of the Politecnico di Milano. The experiment was composed of a training phase and a test session. During the training phase, one of the two involved researchers gave a 10 minutes demonstration, to introduce the participants to the service creation environment and its basic composition mechanisms. A pretest questionnaire was also administered, to gather data on knowledge about computers, mobile devices, services and mashups.

In the test session, each participant was given two test scenarios (Scenario 1 and Scenario 2), communicated by written instructions. Scenario 1 was simpler and required the participant to perform tasks very similar to those demonstrated by the researcher during the tutorial. In Scenario 2, the participant had to perform a greater number of steps, making more service manipulations. After the completion of each scenario, the participants were asked to use the PIS they had just composed.

Scenario 1
In order to organize your free time, you want to keep an eye on the events available in the city of Milano. By exploiting the information available on the Web thanks to the services "Eventful" and "Upcoming", create a map of events. Every event should be accompanied by title, description and address where it will be held.

Scenario 2
You are organizing a trip to Milan. You decide to create a map that displays hotel and metro stations. For each hotel, you would see name and category. For each metro station, you need name and routes passing by that station. The map also shows the position and some photos of the Cathedral.

At the end of the test session, they filled in a questionnaire composed of eighteen questions aiming at evaluating the user satisfaction and the perceived ease of use of the service composition environment.

Results. All the participants were able to complete both scenarios without particular difficulties. The efficiency was measured by analyzing the scenario completion time. The average time for Scenario 1 was about 3m36s, while for Scenario 2 it was about 4m46s. Although such times denote a satisfying ease of use of the system, some users complained about the difficulty of interpreting the service result set shown in the data panel. Most of the time was indeed spent to interpret the service results and especially to identify the relevant attributes. However, they all remarked that the visual representation makes service querying easier with respect to manual programming, and it especially facilitates interpreting the result sets which would instead be represented for example in XML or JSON and would need to be parsed through ad-hoc programming scripts.

The *ease of use* was measured by the data collected through four questions in the post-questionnaire, asking the participants to judge whether they found it easy to identify and include services in the composition and to perform the visual mapping for defining the service query and the visualization of data. The participants also scored the general ease of use of the tool on a 7-point Likert-scale (1 = very negative ÷ 7 = very positive). On average, they gave the ease of use a mark of 5.6 (min = 4.1, max = 6.9, reliability $\alpha = 0.83$).

The *user satisfaction* with the composition paradigm was assessed by using a semantic differential scale that required users to judge the method on 12 items. The participants could modulate their evaluation on a (1 = very negative ÷ 7 = very positive). A user-satisfaction index was computed as the mean value of the score across all the 12 items: mean = 5.2, meanS.E. = 0.94, reliability $\alpha = 0.75$.

A question also asked the participants to globally score the method on a 10-point scale (1 = very negative ÷ 10 = very positive). The global satisfaction was high (mean = 7.7, meanS.E. = 0.7). The last two questions then asked the participants to judge their performance as mashup developers and to indicate the percentage of requirements expressed in the scenarios they believed to have satisfied with their composition, based on the observation of the PIS they were able to generate. This metric can be considered as a proxy of confidence [46]. On average, the participants stated to be able to cover the 94% of requirements specified by the two experimental scenarios (min = 70%, max = 100%, meanS.E. = 10.7%). They also felt very satisfied about their performance as composers (mean = 3.2, meanS.E. = 0.6; on a 4-point scale 1 = very negative ÷ 4 = very positive). This also indicates they found effective the PIS they created.

6.2. Evaluations of PIS creation and use in the CH domain

In the early phases of design and development of the case study, which addresses the creation of PISs to enrich the visit experience of cultural heritage sites, a participatory design team was set up. It also included archaeological guides with a long experience in conducting visits to archaeological parks and museums. They regularly create multimedia presentations for the lectures that they give in schools and cultural clubs. The guides were observed while guiding several groups of visitors and interviewed for iteratively define and refine user requirements [44].

Running prototypes of the platform environments for PIS creation and use were iteratively evaluated, also involving professional guides. In the next sections, the evaluation in the lab and the field study we performed are briefly described.

6.2.1. Evaluation in the lab

Beside several inspections of the different prototypes, formative evaluations were carried out, also including an informal test in the university laboratory. The main goals were to evaluate if the user requirements had been correctly identified and if the proposed composition paradigm and prototypes actually met them. The evaluation consisted in observing in two separate sessions two guides interacting with the prototype for creating a PIS for the visit of a specific archaeological park. They had to collect from some services the multimedia contents specified on a paper sheet, and to compose them with the map of the archaeological park. The test was especially useful to identify some problems of our prototype. Details of this test are in [44]. The main problem experienced by both guides was caused by the feature that permitted to lock the state of the content displayed in the resource windows and already included in the PIS before going on with a new search. In fact, both of them appeared very confused when they performed a new search and the contents displayed in each research window and on the map changed. Even if they were been told about the use of the lock mechanism through the padlock icon displayed at the top right corner of resource windows and map, they were not able to use it in the way designers implemented it. The test observer prompted the tester that he could lock the current content of the resource windows by means of the window padlock. However, none of them was able to exploit the hint, because they did not grasp the reason for doing that. Thus, the observer had to spend some minutes for illustrating the dynamicity of the contents retrieved through the composition. The padlock has been eliminated in the final platform environment for PIS creation, and end users implicitly locks specific contents when they include them in PIS, e.g. they put on the map; they also have the possibility to remove each content from the PIS or to update resource windows and PIS by using specific icons.

6.2.2. *Field study*

During November 2012, a field study was conducted in two different phases to assess composition and use of PISs in real conditions. The first phase aimed at assessing the effects of PIS composition with two professional tourist guides, who composed their PISs relative to the archaeological park of Egnathia using the desktop application, accessible through a PC placed in their office. The second phase was performed a few days later. It consisted in observing use and update of PISs with a large multi-touch display (46-inch) and a tablet device (7-inch) in a real context at the archaeological park of Egnathia during two guided visits, involving 28 visitors randomly divided into two groups of 14 persons. The groups were heterogeneous with respect to age (from 21 to 50 year-old, plus an 8-year old child), gender and cultural background.

Figure 6. The guides interacting with their PIS (a) during the briefing phase using the multi-touch display; (b) during the tour using the tablet.

Each guide first interacted with his/her PIS on the multi-touch display during a briefing phase carried out at the entrance of museum associated to the park to introduce the history of Egnathia and the ruins that they would see later in the park (Figure 6a). During the tour of the park, the guides used tablet devices to show visitors the contents of their PISs regarding what they were looking at (Figure 6b). The guides could also seek for new contents, if and when the opportunity arose, by dynamically formulating new queries to the services integrated in their PISs.

Six HCI experts followed every phase of the field study, taking notes of the most important episodes. At the end of the visit, three experts conducted a focus group with each group of visitors; the discussion addressed the overall visit experience. A third focus group was conducted with the two guides to discuss in more details their experience in using the systems, highlighting pros and cons. The study is described in another paper, together with the detailed analysis of its results [47]. In the following, we briefly discuss some results, which highlight some problems that occurred.

During the PIS composition performed in the office, the few usability problems experienced by the guides were not so serious to stuck them; they were indeed able to continue the PIS composition without the help of the two HCI experts observing them. Both guides appeared disoriented by the few contents returned by some of the searches they had performed; they tried to refine the search by typing different keywords and, finally, added the most appropriate multimedia material available on the Web. This is a problem common to all service-based applications, which have to rely either on content made available by third-party or on user-generated contents. To limit this problem, more sensible services should be added into the platform; they can also be local/ad-hoc created collections of contents, maintained by domain experts and even fed by the end users themselves by adding self-produced material. However, at the end the guides were rather satisfied by the PIS they had created and they were confident that it would be a valuable support during the visit.

During the interaction with the PIS on the multi-touch display, some interaction difficulties were due to some technology limits: 1) a temporary loss of internet connection; 2) in a situation, a guide was not able to close the pop-up window by touching the “X” icon, which was located near the display border, because the employed multi-touch device was not very sensitive along its borders; 3) in the few cases a guide had to use the virtual keyboard displayed on the screen, she experienced the too low precision of the device in correctly detecting the pressed key.

The tablet was not used as frequently as we expected. In the focus group we conducted after the visit, both the guides said that they would have liked to use the tablet more, since they appreciated its support in making available useful material. However, they had used the PIS so little during the tour because they realized that many of the PIS images were available on the panels in the park. Thus, they had preferred to show such images on the panels since the tablet was too small for a group of 14 persons. Tablet size emerged as a problem also in the focus groups with visitors, who said that they preferred to look at images on the panels rather than flocking together around the guide to see them on the tablet. They also complained about the brightness of the tablet screen, compromised by external factors, such as sunlight.

Summarizing, the study results showed a general appreciation of use of PIS in the context of the visit. Many important aspects deserve more attention in future work, such as the opportunity to share some contents between the guide’s tablet and the visitors’ smartphones. We are also interested in investigating user appropriation of the proposed tools [15], i.e. if a long-term use of PISs could have a positive impact on guides’ working activities.

7. Conclusion and future work

This paper has illustrated how service composition technologies can support the definition of a meta-design approach allowing end users, not necessarily experts of technologies, to compose PISs that satisfy their situational information needs and that can be pervasively executed on different devices. Starting from a general-purpose platform for mashup composition, we have identified how some modeling abstractions, which guide the integration of contents within visualization containers, enable different stakeholders to co-create interactive information spaces through intuitive visual paradigms. In line with the current trend of increasing system flexibility by decoupling content, applications, and devices [7], we propose an approach where the composition platform enables the extraction of content from heterogeneous services, and the integration of the extracted content into situational applications where content presentation is flexibly managed through different visualization templates. Thanks to the adoption of platform-independent modeling abstractions, the structure of the composed applications is specified in automatically generated schemas that can be deployed on multiple devices, thus promoting the pervasive fruition of the created PISs, also facilitating their sharing. [7]. Such a separation of concerns, together with the possibility to extend the platform with ad-hoc visualization templates, and the ease of packaging ad-hoc content resources, facilitate the customization of our meta-design approach to specific application domains.

We are currently working on refining our meta-design approach and revising the implementation of our platform accordingly. In particular, we are designing an environment for supporting the WYSIWYG creation of visual templates (which currently have to be manually coded in HTML and JavaScript). We are also refining the techniques used to link the abstract representation of visual templates, as a set of generic visual renderers, to the concrete visualizations in the final applications. This can be achieved through the definition of an intermediate modeling layer highlighting which concrete visual renderers are exploited within the concrete visualizations. This mapping is now hard coded within the execution environment. Moving such a specification to the model level enhances the decoupling between content, presentation and device our approach aims to.

We are also investigating to which extent a collaboration paradigm would improve the usefulness of PISs for supporting the cooperation among different stakeholders. The need for collaboration to co-create and share PISs emerged as a whished feature in all the user-based evaluation sessions we have performed so far. Based on these new requirements, we already defined some extensions of our composition platform to enable PIS annotation, and PIS co-creation [48].

8. Acknowledgments

This work was partially supported by the Italian Ministry of University and Research (MIUR) under grant VINCENTE (A **V**irtual collective **IN**telligen**C**e **EN**vironment to develop sustainable **T**echnology **E**ntrepreneurship ecosystems) and by Italian Ministry of Economic Development (MISE) under grant LOGIN (**LOG**istica **IN**tegrata). We are extremely grateful to the director and the personnel of the Egnathia Archeological Park, and to the professional guides Achille Chillà and Conny Blasi who took part in our field study. We also thank organizers and participants of the EUD4Services 2012 workshop for the interesting discussions from which we got useful suggestions, and all the students that helped us to implement the prototypes.

9. References

[1] A. Sutcliffe, N. Mehandjiev, End-user development (Introduction to Special Issue), Commun. ACM, 47 (2004) 31-32.

[2] H. Lieberman, F. Paternò, V. Wulf (Eds.), End User Development, 9, Springer, Dordrecht, The Netherlands, 2006.

[3] M.F. Costabile, Y. Dittrich, G. Fischer, A. Piccinno (Eds.), End-User Development 2011, Proceedings of the 3rd International Symposium on EUD (IS-EUD 2011), Torre Canne, Italy, June 7-10, Springer, Heidelberg, 2011.

[4] N. Zang, M.B. Rosson, What's in a mashup? And why? Studying the perceptions of web-active end users, in proc. of: IEEE Symposium on Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008., Herrsching am Ammersee, Germany, 2008, pp. 31-38.

[5] A. Namoun, T. Nestler, A. De Angeli, Conceptual and Usability Issues in the Composable Web of Software Services, in: F. Daniel, F.M. Facca (Eds.) Current Trends in Web Engineering, LNCS 6385, Springer, Berlin / Heidelberg, 2010, pp. 396-407.

[6] F. Casati, How End-User Development Will Save Composition Technologies from Their Continuing Failures, in: M.F. Costabile, Y. Dittrich, G. Fischer, A. Piccinno (Eds.) End-User Development, LNCS 6654, Springer, Berlin / Heidelberg, 2011, pp. 4-6.

[7] M. Latzina, J. Beringer, Transformative user experience: beyond packaged design, Interactions, 19 (2012) 30-33.

[8] G. Fischer, E. Giaccardi, Y. Ye, A. Sutcliffe, N. Mehandjiev, Meta-design: a manifesto for end-user development, Commun. ACM, 47 (2004) 33-37.

[9] M.F. Costabile, D. Fogli, P. Mussio, A. Piccinno, Visual Interactive Systems for End-User Development: a Model-based Design Methodology, IEEE T. Syst. Man Cy. A, 37 (2007) 1029-1046.

[10] C. Cappiello, F. Daniel, M. Matera, M. Picozzi, M. Weiss, Enabling End User Development through Mashups: Requirements, Abstractions and Innovation Toolkits, in: M.F. Costabile, Y. Dittrich, G. Fischer, A. Piccinno (Eds.) End-User Development, LNCS 6654, Springer, Berlin / Heidelberg, 2011, pp. 9-24.

[11] C. Cappiello, M. Matera, M. Picozzi, G. Sprega, D. Barbagallo, C. Francalanci, DashMash: A Mashup Environment for End User Development, in: S. Auer, O. Díaz, G. Papadopoulos (Eds.) Web Engineering, LNCS 6757, Springer, Berlin Heidelberg, 2011, pp. 152-166.

[12] F. Daniel, M. Matera, M. Weiss, Next in Mashup Development: User-Created Apps on the Web, 2011, pp. 22-29.

[13] Y. Jin, B. Benatallah, F. Casati, F. Daniel, Understanding Mashup Development, Internet Computing, IEEE, 12 (2008) 44-52.

[14] F. Daniel, F. Casati, B. Benatallah, M.-C. Shan, Hosted Universal Composition: Models, Languages and Infrastructure in mashArt, in: A.F. Laender, S. Castano, U. Dayal, F. Casati, J. Oliveira (Eds.) Conceptual Modeling - ER 2009, LNCS 5829, Springer, Berlin / Heidelberg, 2009, pp. 428-443.

[15] A. Dix, Designing for appropriation, in proc. of: 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it - Volume 2, British Computer Society, University of Lancaster, United Kingdom, 2007, pp. 27-30.

[16] M.F. Costabile, P. Mussio, L. Parasiliti Provenza, A. Piccinno, Supporting End Users to Be Co-designers of Their Tools, in: V. Pipek, M.B. Rosson, B. de Ruyter, V. Wulf (Eds.) End-User Development, LNCS 5435, Springer, Berlin / Heidelberg, Germany, 2009, pp. 70-85.

[17] G. Fischer, Understanding, fostering, and supporting cultures of participation, Interactions, 18 (2011) 42-53.

[18] H. Jenkins, Confronting the Challenges of Participatory Culture: Media Education for the 21st Century, MIT Press, Cambridge, MA, USA, 2009.

[19] J. Porter, Designing for the Social Web, New Riders Press, Thousand Oaks, CA, USA, 2008.

[20] C. Tang, S. Carpendale, An observational study on information flow during nurses' shift change, in proc. of: SIGCHI Conference on Human Factors in Computing Systems (CHI), ACM, San Jose, California, USA, 2007, pp. 219-228.

[21] L. Bannon, S. Bødker, Constructing common information spaces, in proc. of: fifth conference on European Conference on Computer-Supported Cooperative Work (CSCW), Kluwer Academic Publishers, Lancaster, UK, 1997, pp. 81-96.

[22] Y. Yesilada, R. Stevens, S. Harper, C. Goble, Evaluating DANTE: Semantic transcoding for visually disabled users, ACM Trans. Comput.-Hum. Interact., 14 (2007) 14.

[23] J.M.C. Fonseca, W3C Model-Based UI XG Final Report, W3C Incubator Group, 2010, http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/ accessed on February 25, 2013.

[24] F. Paternò, C. Santoro, L.D. Spano, MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments, ACM Trans. Comput.-Hum. Interact., 16 (2009) 1-30.

[25] P.P. da Silva, User Interface Declarative Models and Development Environments: A Survey, in: P. Palanque, F. Paternò (Eds.) Interactive Systems Design, Specification, and Verification, LNCS 1946, Springer, Berlin Heidelberg, 2001, pp. 207-226.

[26] S. Trewin, G. Zimmermann, G. Vanderheiden, Abstract representations as a basis for usable user interfaces, Interacting with Computers, 16 (2004) 477-506.

[27] F. Paternò, C. Santoro, L.D. Spano, Model-Based Design of Multi-device Interactive Applications Based on Web Services, 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I, Springer-Verlag Berlin, Heidelberg, 2009, pp. 892-905.

[28] M.F. Costabile, D. Fogli, A. Marcante, P. Mussio, L. Parasiliti Provenza, A. Piccinno, Designing Customized and Tailorable Visual Interactive Systems, Int J Softw Eng Know, 18 (2008) 305-325.

[29] M. Cáceres, Packaged Web Apps (Widgets) - Packaging and XML Configuration (Second Edition), W3C Recommendation, 2012, http://www.w3.org/TR/widgets/, accessed on February 15, 2013.

[30] A. Ro, L.S.-Y. Xia, H.-Y. Paik, C.H. Chon, Bill Organiser Portal: A Case Study on End-User Composition, in proc. of: International Workshops on Web Information Systems Engineering, Springer-Verlag, Auckland, New Zealand, 2008, pp. 152-161.

[31] E.M. Maximilien, H. Wilkinson, N. Desai, S. Tai, A Domain-Specific Language for Web APIs and Services Mashups, in: B. Krämer, K.-J. Lin, P. Narasimhan (Eds.) Service-Oriented Computing – ICSOC 2007, LNCS 4749, Springer, Berlin / Heidelberg, 2007, pp. 13-26.

[32] R. Krummenacher, B. Norton, E. Simperl, C. Pedrinaci, SOA4All: Enabling Web-scale Service Economies, in proc. of: 2009 IEEE International Conference on Semantic Computing, IEEE Computer Society, Berkeley, CA, USA, 2009, pp. 535-542.

[33] J. Spillner, M. Feldmann, I. Braun, T. Springer, A. Schill, Ad-Hoc Usage of Web Services with Dynvoker, in proc. of: 1st European Conference on Towards a Service-Based Internet, Springer-Verlag, Madrid, Spain, 2008, pp. 208-219.

[34] R. Ennals, E. Brewer, M. Garofalakis, M. Shadle, P. Gandhi, Intel Mash Maker: join the web, SIGMOD Rec., 36 (2007) 27-33.

[35] T. Copeland, Presenting archaeology to the public, in: T. Merriman (Ed.) Public Archaeology, Routledge, London, UK, 2004, pp. 132-144.

[36] Yahoo! Inc., YahooPipes, 2007, http://pipes.yahoo.com/pipes/, accessed on February 22, 2013.

[37] J. Wong, J.I. Hong, Making mashups with marmite: towards end-user programming for the web, in proc. of: SIGCHI

Conference on Human Factors in Computing Systems, ACM, San Jose, California, USA, 2007, pp. 1435-1444.

[38] D. Griffiths, M.W. Johnson, K. Popat, P. Sharples, S. Wilson, The Wookie Widget Server: a Case Study of Piecemeal Integration of Tools and Services, J UNIVERS COMPUT SCI, 18 (2012) 1432-1453.

[39] D. Griffiths, M.W. Johnson, K. Popat, P. Sharples, S. Wilson, The Educational Affordances of Widgets and Application Stores, J UNIVERS COMPUT SCI, 18 (2012) 2252-2273.

[40] A. Soylu, F. Mödritscher, F. Wild, P. De Causmaecker, P. Desmet, Mashups by orchestration and widget-based personal environments: Key challenges, solution strategies, and an application, Program: electronic library and information systems, 46 (2012) 383 - 428.

[41] S. Wilson, F. Daniel, U. Jugel, S. Soi, Orchestrated User Interface Mashups Using W3C Widgets, in: A. Harth, N. Koch (Eds.) Current Trends in Web Engineering, LNCS 7059, Springer Berlin Heidelberg, 2012, pp. 49-61.

[42] M.F. Costabile, D. Fogli, P. Mussio, A. Piccinno, End-User Development: the Software Shaping Workshop Approach, in: H. Lieberman, F. Paternò, V. Wulf (Eds.) End User Development, 9, Springer, Dordrecht, The Netherlands, 2006, pp. 183-205.

[43] H. Beyer, K. Holtzblatt, Contextual design: defining customer-centered systems, Morgan Kaufmann, San Francisco, 1998.

[44] C. Ardito, M.F. Costabile, G. Desolda, M. Matera, A. Piccinno, M. Picozzi, Composition of situational interactive spaces by end users: a case for cultural heritage, in proc. of: 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design (NordiCHI), ACM, Copenhagen, Denmark, 2012, pp. 79-88.

[45] C. Cappiello, M. Matera, M. Picozzi, End User Development of Mobile Mashups, in proc. of: HCI International, Springer, Berlin / Heidelberg, in print.

[46] K. Hornbæk, Current practice in measuring usability: Challenges to usability studies and research, INT J HUM-COMPUT ST, 64 (2006) 79-102.

[47] C. Ardito, M.F. Costabile, G. Desolda, R. Lanzilotti, M. Matera, M. Picozzi, Personal Information Spaces for Enhancing Visits to Archaeological Parks, submitted for publication.

[48] 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, End-User Develpment, LNCS, Springer, Berlin / Heidelberg, in print.