Designing and Developing Context-Aware Mobile Mashups: The CAMUS Approach

Fabio Corvetta, Maristella Matera(✉), Riccardo Medana, Elisa Quintarelli, Vincenzo Rizzo, and Letizia Tanca

Dipartimento di Elettronica, Informazione e Bioingegneria,
Politecnico di Milano, Milan, Italy
{fabio.corvetta,vincenzo.rizzo}@mail.polimi.it,
{maristella.matera,elisa.quintarelli,letizia.tanca}@polimi.it,
r.medana@gmail.com

Abstract. CAMUS (Context-Aware Mobile mashUpS) is a framework for the design of mobile applications that dynamically collect and integrate heterogeneous resources (data sources and services) to offer integrated content and functions to mobile users in a context-aware fashion. CAMUS exploits a set of high-level abstractions for context and mashup modeling that hide the complexity resulting from service selection, invocation and integration. Generative techniques then enable the transformation of models into running code for mobile applications that flexibly respond to actual user needs as they vary in different situations of use.

Keywords: Mobile mashups · Context modeling · Context-aware mobile applications · End-user development

1 Introduction

Given the plethora of data and services today available online, it is often difficult to find on-the-fly the information or the applications that are appropriate to the current context of use [2]. This is even more true in the mobile scenario, where device resources (memory, computational power, transmission budget) are still limited. Given this evidence, our research focuses on the definition of methods and tools for the design and development of Context-Aware Mobile mashUpS (CAMUS). CAMUS apps dynamically collect and integrate data from documental, social and Web resources (accessed by means of Web APIs) and adapt the integrated content to the users' situational needs. They can offer multiple advantages thanks to their intrinsic capability of identifying pertinent data sources, selected on the basis of their adequateness with respect to the current users' needs, and pervasively presenting them to the final user in form of integrated visualizations deployed as mobile apps. This application paradigm overcomes the limits posed by pre-packaged apps and offers to users flexible and personalized applications whose structure and content may even emerge at runtime based on the actual user needs and situation of use.

P. Cimiano et al. (Eds.): ICWE 2015, LNCS 9114, pp. 651–654, 2015.
DOI: 10.1007/978-3-319-19890-3_49

The CAMUS framework does not only propose a new application paradigm; rather, as described in the following section, it paves the way to novel design methodologies and related tools for fast prototyping of mobile mashups, where context becomes a first-class modeling dimension improving *i)* the identification of the most adequate resources that can satisfy the users'information needs and *ii)* the consequent tailoring at runtime of the provided data and functions.

2 The CAMUS Framework

The CAMUS framework is characterized by design environments that, in line with recent approaches to visual programming of mashups, make intensive use of high-level visual abstractions [1]. Visual paradigms hide the complexity that is typical of service composition, data integration and mobile application programming, and assist CAMUS designers (even if non-experts in service composition) in the creation of personalized applications that can be run on multiple devices without the need of mastering different technologies. All the aspects that characterize the different contextual situations, i.e., the *dimensions* contributing to context, are modelled orthogonally with respect to the other instantaneous system inputs by means of the so-called *Context Dimension Model* [3,4], which provides the constructs to define at design-time the Universal Context Dimension Tree (*Universal CDT*), i.e., the set of possible contexts of use for a given domain of interest, expressed as a hierarchical structure consisting of *i)* context dimensions (black nodes), modeling the context variables, i.e., the different perspectives through which the user perceives the application domain (e.g., time, place, current company, interest topic), and *ii)* the allowed dimension values (white nodes), i.e., the variable values used to tailor the context-aware information (e.g., "evening", "New York", "with friends", "music"). Any subtree of the CDT with at most a value for each dimension represents a possible user context; variable values, detected at runtime through device sensors, then activate a given context. The adoption of a hierarchical structure allows us to employ different abstraction levels to specify and represent contexts.

Figure 1 represents the general organization of the CAMUS framework; it highlights its main architectural components and the flow of the different artifacts that enable the transition from high-level modeling notations to running code. The framework supports the activities of three main *personae*.

The *CAMUS administrator* is in charge of registering distributed resources (remote APIs or in-house services) into the platform, by creating descriptions of how the resources can be invoked and wrappers ensuring homogeneity of data formats. S/he also specifies the Universal CDT and the *mapping* between the identified context elements and some pertinent services among those previously registered. This mapping expresses the capability of services to return data and functions of interest with respect to the specified contexts. In other words, the Universal CDT expresses for each given context a virtual image of the relevant portion of the available resources.

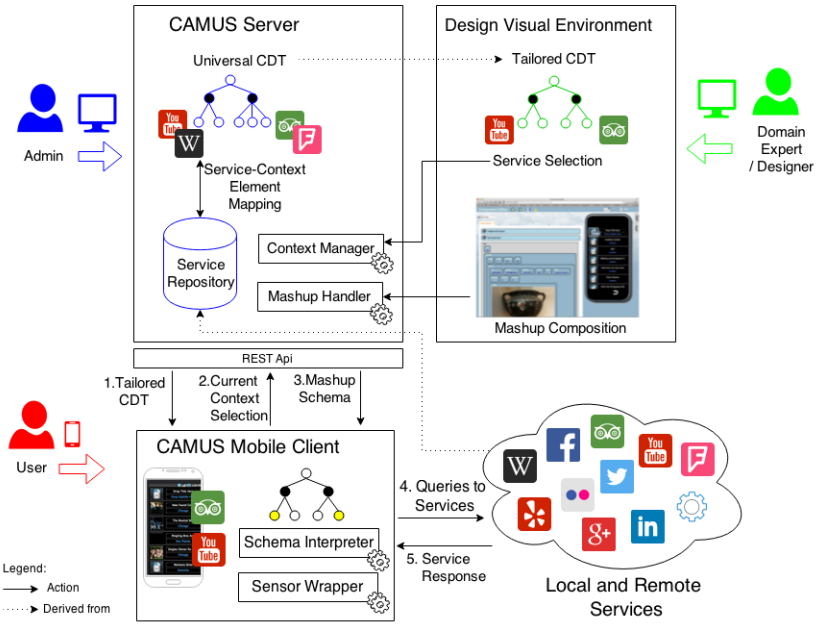

Fig. 1. Architecture of the CAMUS framework

The *CAMUS (MashUp) designer* starts from the image on the available resources represented by the universal CDT and, using a *Design Visual Environment*, defines the *Tailored CDT* by further refining the selection of possible contexts based on the needs and preferences of specific users or users' groups. The designer also defines how to mash up the identified services, i.e., how to integrate their result sets and synchronize their functions, and how to visualize the resulting information through unified views deployed as mobile apps. The result of this activity is an an XML-based *mashup schema*, automatically generated by the design environment based on a Domain Specific Language [5], which includes rules that at runtime guide the execution of the resulting app.

The *CAMUS (app) users* are the final recipients of the mobile app that offers a different bouquet of content and functions in each different situation of use. When the app is executed, the context parameter values that characterize the current situation, identified by means of a client-side *Sensor Wrapper*, are communicated to a server-side *Context Manager*. Based on the tailored CDT, this module computes the context-dependent parameters to instantiate queries to and invoke functions from the integrated services. The mashup schema created by the designer is interpreted locally (by means of a *schema interpreter*) to send parametric requests to the involved services and populate with the returned data the integrated views provided by the mobile app.

A dedicated module of the design environment, the *Mashup Handler*, translates the designers' visual composition actions into XML-based schemas that

express rules for data integration and event-driven synchronization of service functions [5]. The platform indeed exploits generative techniques that comply with Model-Driven Engineering methods: modeling abstractions guide the design of the final applications while generative layers mediate between high-level visual models and low-level technical engines that execute the final mashups. Execution engines, created as native applications for different mobile devices, then make it possible the interpretation and pervasive execution of schemas.

It is worth noting that, in comparison to other approaches to mashup design [6], the composition activity and more specifically the selection of services is not exclusively driven by the functional characteristics of the available services or by the compatibility of their input and output parameters. Rather, the initial specification of context requirements enables the progressive filtering of services first and then the tailoring of service data to support the final situations of use.

3 Conclusions and Acknowledgments

This poster illustrates the CAMUS framework whose aim is to empower developers to create context-aware, mobile or Web-based apps by integrating multiple and heterogeneous APIs acting on situational needs. CAMUS can offer advantages in several application domains, as for instance tourism and enterprise, two domains for which our previous work already highlighted the need of methods and tools for the creation of flexible, situational applications [5]. Our current work is devoted to refining the implementation of the platform, and in particular to improve the integration of a visual mashup environment [5] with the CAMUS back-end. We thank the large group of students of Politecnico di Milano who enthusiastically contributed to the design and implementation of the first CAMUS prototype through which we assessed the feasibility of revising mashup composition practices through the introduction of context modeling concepts.

References

1. Ardito, C., Costabile, M.F., Desolda, G., Lanzilotti, R., Matera, M., Piccinno, A., Picozzi, M.: User-driven visual composition of service-based interactive spaces. J. Vis. Lang. Comput. **25**(4), 278–296 (2014)
2. Bianchini, D., Castano, S., De Antonellis, V., Ferrara, A., Quintarelli, E., Tanca, L.: RUBIK: Proactive, Entity-Centric and Personalized Situational Web Application Design. T. Large-Scale Data-and Knowledge-Cent. Syst. **13**, 123–157 (2014)
3. Bolchini, C., Curino, C., Orsi, G., Quintarelli, E., Rossato, R., Schreiber, F.A., Tanca, L.: And what can context do for data? CACM **52**(11), 136–140 (2009)
4. Bolchini, C., Orsi, G., Quintarelli, E., Schreiber, F.A., Tanca, L.: Context modeling and context awareness: steps forward in the context-addict project. IEEE Data Eng. Bull. **34**(2), 47–54 (2011)
5. Cappiello, C., Matera, M., Picozzi, M.: A UI-centric approach for the End-User Development of multi-device mashups. ACM Trans. on Web (to appear, 2015)
6. Daniel, F., Matera, M.: Mashups - Concepts, Models and Architectures. Data-Centric Systems and Applications. Springer (2014)