Integration of Heterogeneous Context Resources in Ubiquitous Environments

Daniel Romero, Romain Rouvoy, Lionel Seinturier, Frédéric Loiret

Integration of Heterogeneous Context Resources in Ubiquitous Environments

Daniel Romero, Romain Rouvoy, Lionel Seinturier, and Frédéric Loiret
INRIA Lille – Nord Europe, ADAM Project-team,
University of Lille 1, LIFL CNRS UMR 8022,
59650 Villeneuve d'Ascq, France
firstname.lastname@inria.fr

***Abstract*—Ubiquitous environments provide families of context-aware applications that are capable of exploiting the user mobility as well as the device variability. Typically, these applications retrieve context information from local and remote providers and react accordingly to the detected variations. However, this must be done by considering the heterogeneity of devices and protocols found in ubiquitous environments. Unfortunately, although the context integration represents a keystone of context-aware systems, existing approaches in the literature fail to integrate the diversity of context sources in a standard and flexible way. Therefore, in this paper, we overcome this challenge by introducing resource-oriented bindings into the SCA (*Service Component Architecture*) model. This new kind of bindings follows the *REpresentational State Transfer* (REST) principles and leverages the provision of context as RESTful resources. A smart home scenario that highlights challenges in terms of integration in ubiquitous environments motivates the use of our approach.**

***Keywords*-context-awareness; SCA; middleware; integration; ubiquitous computing; heterogeneity; REST**

I. Introduction

In ubiquitous environments, applications use context information to adapt their behaviour according to changing conditions. For that, the context providers of this information have to be discovered and then accessed by client applications. These issues are related to two key concepts in pervasive computing: *context dissemination* [1] and *spontaneous interoperability* [2]. Both of them should be addressed by considering the heterogeneity in terms of interaction and discovery protocols, devices, and context providers that rules in these environments. Regrettably, the existing approaches fail to integrate the diversity of context sources in a standard and flexible way regarding this heterogeneity. Hence, a simple mechanism that allows the context integration in ubiquitous environments is necessary.

In [3], we have proposed an approach to deal with discovery issues in ubiquitous environments. In this paper, we focus on the challenge of integrating heterogeneous context providers. To do this, we extend the SCA component model [4], which combines the *Component-Based Software Engineering* (CBSE) principles and SOA to build distributed applications. In particular, we introduce a new kind of binding, the *resource-oriented bindings*, that enables the exchange of context information by following the *REpresentational State Transfer* (REST) principles. These bindings support context dissemination by exposing context information as resources that can be accessed via standard and simple protocols (*e.g.*, HTTP, FTP). In this way, our solution leverages on the notion of *context as a resource*. We claim that the combination of REST and SCA fosters low coupling between the interacting entities and makes it easier to deal with context distribution.

This paper is organized as follows. We start by illustrating the importance of context integration in ubiquitous environments (cf. section II) using a smart home scenario and then we introduce SCA, the foundation of our proposal (cf. section III). In section IV, we present the *resource-oriented bindings*, our solution to deal with context integration. Next, we present the evaluation of our implementation (cf. section V) and some works associated with context distribution (cf. section VI). We finish with some conclusions and perspectives (cf. section VII).

II. Motivations and Challenges

To highlight the challenges for supporting context mediation in ubiquitous environments, we propose a smart home scenario that is inspired by the DIGIHOME platform [5]. In this scenario, Alice's house is equipped with several sensor nodes to monitor the current temperature, occupancy (movement detection), noise level, and light states. Rooms also include *actuators* to physically control lights, air conditioning, blinds, and TV.

In order to customize the configuration of the rooms, the mobile devices of Alice and her family store their preferences for each room. These preferences describe the temperature, noise, and light levels accepted by each family member. When several people meet in the same room, the decision is based on merged preferences. In case of conflict, the decision prioritizes the first person who enters the room. Below we describe a concrete situation of the scenario.

Alice's daughter, Jenny, listens to music in the living room. The temperature, light, and noise levels conform to her preferences. When Alice enters the living room, the controller detects Alice's device and retrieves the valid preferences. We assume that there is no conflict with the temperature level. However, the noise level is too high for Alice's tolerance. The noise range specified by Jenny

includes the level accepted by Alice. Hence, the system decides to reduce the volume of the audio system according to Alice's preferences. The light preferences are also conflictual, but the system decides to keep the daughter's configuration as she arrived first.

Challenges: According to our scenario, we can identify that the integration of context providers and context-aware applications in ubiquitous environments imposes the following challenges: *i)* how to *deal with the heterogeneity* in terms of devices, services (actuators, sensors, and other context sources) and context information, *ii)* how to *tackle consumer/producer mobility* (dynamism) and, *iii)* how to *deal with context quality*, such as accuracy, precision, trustworthiness, up-to-dateness, and completeness [6]. In section IV, we present our approach to deal with the first issue. The solution to face the mobility and provider selection based on quality of context information can be consulted in [3].

III. Service Component Architecture

The *Service Component Architecture* [4] (SCA) is a set of specifications for building distributed application based on SOA and *Component-Based Software Engineering* (CBSE) principles. In SCA, the basic construction blocks are *software components*, which have *services* (or provided interfaces), *references* (or required interfaces) and expose properties. The references and services are connected by means of *wires*. SCA specifies a hierarchical component model. This means that components can be implemented either by primitive language entities or by subcomponents. In the latter case the components are called *composites*.

SCA is designed to be independent from programming languages, *Interface Definition Languages* (IDL), communication protocols and non-functional properties. In particular, to support interaction via different communication protocols, SCA provides the notion of *binding*. For SCA references, *bindings* describe the access mechanism used to invoke a service. In the case of services, the bindings describe the access mechanism that clients use to execute the service.

IV. Resource-oriented Bindings

As already mentioned, one of the challenges for context-aware applications is the heterogeneity. In this paper, we tackle this issue by defining a new type of binding for the SCA component model: *Resource-oriented Bindings*. These bindings provide a *simple but efficient* context mediation solution that allows the dissemination of context information. The simplicity and efficiency are leveraged on the REST principles [7]. Typically, REST defines the principles for encoding (*content types*), addressing (*nouns*), and accessing (*verbs*) resources using Internet standards (*e.g.*, URIs, HTTP, XML, and mime-types). The simplicity, lightness, reusability, extensibility, and flexibility properties that characterize REST make it a suitable option for context integration in ubiquitous environments.

Following the REST principles, we define the nouns, verbs, and context types for the new bindings. The nouns are *context identifiers* conforming to the URI format. Therefore, context identifiers include a *communication scheme*, a *server address*, a *context path*, and *request parameters*:

scheme://context-server/context-path?request-parameters

In order to provide access to context information, we use the REST verbs (*i.e.*, `GET`, `PUT`, `POST`, `DELETE`) to encode the operations typically offered by context providers: *retrieve* using `GET`, *notify* with `PUT`, *subscribe* using `POST` and, *unsubscribe* with `DELETE`.

For example, the HTTP request below is produced automatically by our resource-oriented binding to retrieve the context information *light level preference* of the living room from a mobile device:

```
GET /preferences/living_room/light_level HTTP/1.1
Host : device.inria.fr:8080
Accept : application/xml, application/json
...
```

The previous example shows that we use the request field `Accept` to specify the accepted representations for the retrieved context information. As it can be seen, in our resource-oriented bindings, we use the MIME media types classification to describe the supported representations.

Service (server-side)

```
<composite name="living-room-node.composite">
  <service name="living-room-light-level-preference"
    promote="light-level-node/living-room-light-level-preference"/>
  <component name="light-level-node">
    <implementation.java
            class="smarthome.mobile.LightLevelOperator"/>
    <service name="living-room-light-level-preference">
      <resource.xml schema="light-level-preference.xsd"/>
      <resource.json schema="light-level-preference.json"/>
      <binding.http uri="/preferences/living_room/light_level"/>
      ...
    </service>
      ...
  </component>
  <component name="temperatute-level-node">
    ...
  </component>
    ...
</composite>
```

Reference (client-side)

```
<composite name="smart-home-node.composite">
  <reference name="living-room-light-level-preference"
        promote="living-room-light-configuration-node/
                 living-room-light-level-preference">
  <component name="living-room-light-configuration-node">
    <implementation.java
        class="smarthome.controller.LRLightConfigurationOperator"/>
    <reference name="living-room-light-level-preference">
      <resource.xml schema="light-level-preference.xsd">
      <binding.http uri="http://device.inria.fr:8080/preferences
                         /living_room/light_level"/>
      ...
    </reference>
  </component>
  <component name="configuration-stabilization-node">
    ...
  </component>
    ...
</composite>
```

Figure 1. Example of an SCA Definition for Resource-oriented Bindings

Figure 1 depicts a resource-oriented binding definition example for services (upper part) and references (lower

part). Unlike the SCA typical bindings, the services and references using resource-oriented bindings do not need to expose interfaces. Instead, the services (resp. references) specify the provided (resp. required) resources. To do this, we add a new element, `<resource.representation>`, that expresses the required or provided information including its representation and type (defined with a schema).

In the service-side, the `<resource.representation>` element describes the different formats used to offer the information. For its part, the `<binding.protocol>` element specifies the protocol supported to exchange the information and the path to access it (via the `uri` attribute). In the reference-side, the `<resource.representation>` element defines the required information and the `<binding.protocol>` expresses how the device can retrieve it—*i.e.*, the supported communication protocol. The `uri` attribute in the `<binding.protocol>` element is optional (because it can be defined at runtime, for example, if discovery protocols are activated [3]) and contains the address for retrieving the information. In this way, by encapsulating the resource retrieval as SCA bindings, we enable the development of applications following a resource-oriented approach and the integration of not only context information, but also any kind of data (*e.g.*, streams and events).

V. Empirical Validation

Implementation Details: We have integrated our resource-oriented bindings into the FraSCAti [8] platform. We used the COSMOS framework [9] to implement the context policies. Both, FraSCAti and COSMOS, are based on the Fractal component model and use the Julia[1] reference implementation.

Performance Results: To evaluate our approach, we implemented the scenario introduced in section II as a COSMOS context policy.

Table I summarizes the results we obtained when executing several physical configurations for the same context policy using HTTP and Twitter[2] as communication protocols. The values reported correspond to the observation delay for deciding the room configuration.

The message exchanged between the mobile device and the controller aggregates all the user's preferences. Thus, considering that the context processing is locally performed in $135.5ms$ (configuration *a*), we observe that the overhead introduced by the resource-oriented bindings for retrieving context information costs around $150ms$ per observation (compared to configuration *b*, where we distributed the context policy using the two laptops).

We also appreciate that the deployment of the distributed context policy on a mobile device (configuration *d*) introduced an additional overhead compared to the deployment on two laptops (configuration *c*). This overhead is mostly due to the limited processing and networking capacities of the mobile device. Furthermore, in this experimentation, the deployment of the XML version of the distributed context policy was not possible due to a limitation of the Java Virtual Machines used on the mobile device (CACAOVM & JamVM). Nevertheless, this is not a problem for our approach since several representations are available for the same context information.

Table I
Performances of Resource-oriented Bindings in *ms*

Providers Configuration	Provider	Resource Representation				
		Object	JSON	XML	JSON	XML
a) No Provider	N/A	135.5				
b) 1 Local Provider	N/A	281.7	293.8	300.8	965.2	971.44
c) 1 External Provider	MacBook Pro	209.8	221	228	948.2	951.55
d) 1 External Provider	N800	475.1	511.25	N/A	1191.4	N/A
e) 2 External Providers	MacBook Pro & N800	519.7	604	N/A	1738.4	N/A
f) 2 External Providers	N800 A & B	839.5	883.1	N/A	1772.9	N/A
	Interaction	HTTP			Twitter	

In the microblogging case, the values are considerably higher because of the Internet indirection introduced by Twitter. The use of this service always requires to access the server that hosts the tweets. However, despite of this overhead, the simplicity of this interaction mechanism makes it still a suitable alternative to share the context information in an asynchronous way.

VI. Related Work

In literature, it is possible to find two kinds of solutions to deal with context integration: *centralized* and *decentralized*. In the centralized category we can find middleware, such as PACE [10] and *Context Distribution and Reasoning* (ConDoR) [11]. PACE proposes a centralized context management system based on repositories. The context-aware clients can interact with the repositories using protocols, such as Java RMI or HTTP. For its part, ConDoR takes advantage of the object-oriented model and ontology-based models to deal with context distribution and context reasoning, respectively. In ConDoR, the integration of the information using different protocols is not considered as an important issue. The problem with this kind of approach is the introduction of a single point of failure into the architecture, which limits its applicability to ubiquitous computing environments.

On the other hand, in decentralized approaches, we find solutions like CORTEX [12] and MUSIC [13]. CORTEX defines sentient objects as autonomous entities that have the capacity of retrieving, processing, and sharing context information using HTTP and SOAP. MUSIC middleware is another decentralized solution that proposes a peer-to-peer infrastructure dealing with context mediation. The decentralized approaches address the problem of fault tolerance

[1] Julia: http://fractal.ow2.org/julia
[2] Twitter Microblogging Service: http://twitter.com

by distributing the information across several machines. However, as well as some centralized solutions, the lack of flexibility in terms of the communication protocols remains a key limitation for these approaches. In addition to that, peer-to-peer approaches have performance and security problems.

As already said, the context integration by means of SCA bindings promotes the use of diverse interaction mechanism. Furthermore, the inclusion of the mediation mechanisms as SCA bindings fosters their reuse for resource exchange in different kinds of systems, not only context-aware applications. On the other hand, the combination of REST principles with SCA makes easier the integration of legacy systems and reduces the coupling between the different consumers and providers by focusing in the exchanged data.

VII. Conclusions and Perpectives

In order to face the heterogeneity in terms of communication and representation in ubiquitous environments, we introduce a context mediation middleware to leverage on the notion of context as RESTful resources. This solution benefits from the SCA component model and the REST principles in order to define *resource-oriented bindings* encapsulating the context distribution concern and enabling RESTful communications in the SCA-based applications. By benefiting from SCA extensibility and its clear separation of concerns, we integrate a resource-centric communication in a transparent way. Thus, the originality of our solution rests on its simplicity and efficiency achieved by the combination of well defined and accepted standards and protocols.

In order to show the suitability of our approach, we have performed our evaluations on a smart home scenario. The empirical evaluation of resource-oriented bindings shows that the introduced overhead is negligible compared to the benefits of considering context as RESTful resources.

Future work includes further tests using different kinds of mobile devices, protocols, and context managers. In particular, we plan to include a support for others standard protocols, such as the *Extensible Messaging and Presence Protocol* (XMPP)[3]. We also plan to benefit of the FraSCAti advantages in terms of runtime reconfiguration of providers and consumers to build ubiquitous feedback control loops [3] in the context of the CAPPUCINO project[4].

References

[1] A. Buchner, "Context Mediation among Knowledge Discovery Components," Ph.D. dissertation, University of Ulster, UK, 2004.

[2] F. Zhu, M. W. Mutka, and L. M. Ni, "Service Discovery in Pervasive Computing Environments," *IEEE Pervasive Computing*, vol. 4, no. 4, pp. 81–90, 2005.

[3] D. Romero, R. Rouvoy, and L. Seinturier, "Service Discovery in Ubiquitous Feedback Control Loops," in *10th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'10)*, ser. LNCS, vol. 6115. Springer, Jun. 2010, pp. 113–126.

[4] Beisiegel, M. et al, "Service Component Architecture," Nov. 2007.

[5] D. Romero, G. Hermosillo, A. Taherkordi, R. Nzekwa, R. Rouvoy, and F. Eliassen, "RESTful Integration of Heterogeneous Devices in Pervasive Environments," in *10th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'10)*, ser. LNCS, vol. 6115. Springer, Jun. 2010, pp. 1–14.

[6] A. Manzoor, H.-L. Truong, and S. Dustdar, "On the evaluation of quality of context," in *EuroSSC '08: Proceedings of the 3rd European Conference on Smart Sensing and Context*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 140–153.

[7] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[8] L. Seinturier, P. Merle, D. Fournier, N. Dolet, V. Schiavoni, and J.-B. Stefani, "Reconfigurable SCA Applications with the FraSCAti Platform," in *IEEE International Conference on Services Computing (SCC'09)*. IEEE Computer Society, 2009, pp. 268–275.

[9] R. Rouvoy, D. Conan, and L. Seinturier, "Software Architecture Patterns for a Context-Processing Middleware Framework," *IEEE Distributed Systems Online*, vol. 9, no. 6, Jun. 2008.

[10] K. Henricksen, J. Indulska, and T. Mcfadden, "Middleware for Distributed Context-Aware Systems," in *International Symposium on Distributed Objects and Applications (DOA'05)*. Springer, Nov. 2005, pp. 846–863.

[11] F. Paganelli, G. Bianchi, and D. Giuli, "A Context Model for Context-Aware System Design Towards the Ambient Intelligence Vision: Experiences in the eTourism Domain," in *Universal Access in Ambient Intelligence Environments*, 2006, pp. 173–191.

[12] C.-F. Sorensen, M. Wu, T. Sivaharan, G. S. Blair, P. Okanda, A. Friday, and H. Duran-Limon, "A context-aware middleware for applications in mobile Ad Hoc environments," in *2nd Workshop on Middleware for Pervasive and Ad-hoc Computing (MPAC'04)*. ACM, Oct. 2004, pp. 107–110.

[13] M. Kirsch-Pinheiro, Y. Vanrompay, K. Victor, Y. Berbers, M. Valla, C. Frà, A. Mamelli, P. Barone, X. Hu, A. Devlic, and G. Panagiotou, "Context Grouping Mechanism for Context Distribution in Ubiquitous Environments," in *International Conferences on Distributed Objects and Applications (DOA'08)*, ser. LNCS. Springer, Nov. 2008, pp. 571–588.

[3] XMPP Standards Foundation: http://xmpp.org/

[4] CAPPUCINO project: http://www.cappucino.fr