Context-Aware Leisure Service:

A Case-Study based on a SOA 2.0 Infrastructure

Guadalupe Ortiz, Juan Boubeta-Puig
UCASE Software Engineering Group
Department of Computer Science and Engineering
University of Cádiz
Cádiz, Spain
{guadalupe.ortiz, juan.boubeta}@uca.es

Adrián Brenes Ureba
Higher School of Engineering
University of Cádiz
Cádiz, Spain
adrian.brenesureba@alum.uca.es

Abstract— **Service-Oriented Architectures (SOAs) have settled as an efficient solution for the implementation of systems in which modularity, loose-coupling and communication among third parties are key factors. However, although there are excellent tools and frameworks for service development, their adaptation to context has not been properly focused on to date. In this paper, we have made use of a SOA 2.0, where the core element is an enterprise service bus, in order to improve context-awareness for services. The proposal is illustrated trough a real case-study scenario implementation, where the results show the benefits of using such an architecture for web service context-awareness.**

Keywords- Web Service; Context-Awareness; Service-Oriented Architecture; Enterprise Service Bus.

I. INTRODUCTION

In recent years, Service-Oriented Architectures (SOAs) have settled as an efficient solution for the implementation of systems in which modularity, loose-coupling and communication among third parties are key factors. This fact has led to the increasing development of distributed applications composed of reusable and sharable components (services). These components have well-defined platform-independent interfaces, which allow SOA-based systems to quickly and easily adapt to changing business conditions.

However, although there are excellent tools and frameworks for service development, their adaptation to context has not been properly focused on to date. Even though this is a field in which many industry and scientific community are starting to provide their proposals [1]–[5], there are no clear solutions in the scope of web services. To illustrate the need for adaptation, let us provide an example: for instance, we may have services that would be suitable for their adaptation to the invoking client's specific context- such as his location or the weather conditions in his location. This would imply that service answers should be adapted depending on these contextual situations. In the past, we proposed a method for adapting services to the invoking device [6], as well as to adapt them to the client-specific context in general [7]. These approaches are good for the specific type of context dealt with – adapting to device and client-specific context – but are not prepared to deal with the external context.

In this regard, adapting services to context and current conditions might require the analysis of context information very often. Nevertheless, SOAs are not suitable for environments where it is necessary to continuously analyze the information flowing through the system, a key factor for an appropriate context-aware service implementation. This limitation may be solved by the joint use of Complex Event Processing (CEP) [8] together with SOA, the so-called event-driven service-oriented architecture or SOA 2.0 [9]: an extension of SOA to respond to events that occur as a result of business processes. However, most approaches implementing context-aware services do not take advantage of the use of CEP and SOA 2.0, therefore having to continuously access a context manager [1]–[3], [10]. See Section III analysis on related work for further details.

We already envisaged an architecture for this purpose in [11], in which the key element is an Enterprise Service Bus (ESB), which currently is the core of SOA 2.0. The main contribution of this paper is the definition of the exact architecture required and its implementation through a real case-study scenario. To this end, we have chosen a service which provides leisure activities. Particularly, the provided activities will be based on the location and weather conditions of the user; weather conditions will also be used to send special offers to subscribers.

The rest of this paper is organized as follows. Section II provides some background on the paper main areas of interest: context-awareness and event-driven SOAs. Afterwards, Section III describes and compares more relevant related work to the one presented in this paper. Then Section IV addresses the implemented architecture, first of all including the case-study description, secondly the architecture definition and finally the flows required in the ESB for materializing the good use of the SOA 2.0 architecture. Following, Section V provides the final application overview from the point of view of the different user roles, specially focusing on how context-awareness is dealt with. The article ends with Section VI, which discusses the proposal and conclusions.

II. BACKGROUND

In this section, we will introduce the main concepts of context-awareness and event-driven SOA.

 ISBN: 978-1-61208-361-2

A. Context-Awarenes

Dey et al.'s context definition in [12] is specially well-known: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".

Context-awareness supports the fact that the context information provided by the client, or taken from the environment, is properly used by the system so as to improve its quality; that is, using such contextual information to customise system outputs to improve final user satisfaction. Therefore, a system is context-aware if it uses the context to provide relevant information or services to the user, adapting the system behavior to the user particular needs.

A context-classification can be found in [13]; in this work, we will focus on dealing with the environmental context: sensors and/or specific services are currently used in order to provide such kind of information as location, temperature, precipitations, wind, etcetera. This type of context will imply filtering the information sent to the client; for instance, if location is taken into account when looking for leisure activities, the result would be restricted to those available within a limited distance; when taking into account weather conditions, the results might be narrowed and only activities suitable for these conditions might be provided.

B. Event-Driven SOA (SOA 2.0)

Event-driven architectures promote the detection of events and the subsequent intelligent reaction to them [14]. These architectures rely on complex event processing, a technology that provides a set of techniques to help discover complex events by analyzing and correlating other basic and complex events [8]. Therefore, CEP allows detecting complex and meaningful events in a particular context and inferring valuable knowledge for end user interests. Let us suppose again that we are looking for leisure activities for today; kite-surfing is fine when is windy; however it is not the same that it is windy when you are going to kite-surf, than that it is windy and raining more than 10 cm^3/h. This is an example on how complex events may help make decisions on the information to be provided to the user.

Currently, the integration of Event-Driven Architecture and SOA is known as Event-Driven SOA or SOA 2.0 [9]. SOA 2.0 will ensure that services do not only exchange messages between them, but also publish events and receive event notifications from others. For this purpose, an Enterprise Service Bus (ESB) will be vastly helpful to process, enrich and route messages between services of different applications. Further information on the integration of CEP with SOA in other scenarios can be found in [15].

III. RELATED WORK

In this section, we will focus on the main research for CEP and SOA integration and context-aware service implementations.

Several works on CEP and SOA integration in different domains can be found in the literature; for instance, Taher et al. [16] develop an architecture that integrates a CEP engine and input/output adapters for SOAP messages in order to adapt Web service messages between incompatible interfaces: input adapters receive messages sent by Web services, transform them into the appropriate representation to be manipulated by the CEP engine and send them to the latter. Accordingly, output adapters receive events from the engine, transform them into SOAP messages and send them to the corresponding to Web services.

There are some approaches which use CEP for monitoring such as the one from Xu et al. [17], where CEP is used to detect events in an Ambient Assisted Living (AAL) domain so that proper palliative actions can be taken in real time. The paper from Li et al. [18] is also worth a special mention. They provide an adaptive approach to context provisioning and automatic generation of actions. The latter definitely bears similarities with our proposal; however we focus on non-intrusive service result adaptation rather than action taking.

Most of the work found in the context adaptation area specially focuses only on websites [19] or in general on client side adaptation. We can mention, for instance, the paper from Laakko and Hiltunen [1] where content adaptation is done through a proxy; we can also mention the one by Mohomed et al. [2] where the system can learn about context through the interaction with the user. Both are interesting works, but they overhead the client computation; opposite to our proposal which deals with all the heavy tasks in the server side. Another example is the proposal from Keidl et al. [3], which consists of an approach for services to deal with client contextual information through a context framework. In their case, the context is always included in the client SOAP header, as well as in service messages. This implies that not only services, but also clients have to process the context included in the header; however they do not explore how the client can deal with the received context, and again they are overheading client communications.

Bucchiarone et al. [4] focus on the role of context in adaptation activities and describe a life-cycle for designing and developing adaptable service-based applications. They consider necessary to build contextual monitors and adaption mechanisms to detect context changes and trigger the subsequent actions. Furthermore, they propose rule engines as possible candidates for this purpose. However, implementations using rule engines are slower and less efficient in handling and receiving notifications, compared to those using CEP engines [20].

Sheng et al. [5] proposes *ContextUML*: a modeling language for context-aware model-driven web services. Several years later they improved their proposal supplying a platform for developing context-aware web services [21]. This platform, named *ContextServ*, is based on *ContextUML* and provides an integrated environment where developers can specify and deploy context-aware services, as well as generating Business Process Execution Language code. The main drawback of this proposal is the high learning curve required for their modelling methodology; in addition, it does not take any advantage of the use of the ESB and CEP,

 ISBN: 978-1-61208-361-2

which leverages the context-aware system scalability, usability and maintenance.

There are also several approaches which only consider location and personal preferences [22], [23], but no other environmental contexts; those are not relevant in this scope.

To summarize, our proposal mainly differs from others in benefiting from the advantages of the use of CEP and an ESB to adapt services to context information in a decoupled and scalable way, where the context can be automatically detected through real time events.

IV. CASE-STUDY DESCRIPTION

In this section, we are going to describe the case-study requirements and the proposed architecture to implement it.

A. *Description*

The goal is having a service-based application which offers leisure activities, as well as special offers depending on the weather conditions in a particular location.

The *application manager* (web master) will be the one in charge of defining a set of categories corresponding to weather conditions; for instance, wind speed above 40km/h means it is a windy day; otherwise it is not.

The *activity providers* will define on the one hand which activities are suitable under specific weather conditions (for instance a camera obscura is not a place to recommend when it is already dark whilst it is really nice in a sunny day). On the other hand, special offers which might be triggered under specific weather categories: the camera obscura provider might recommend it as long as there is day light, but since many people would not go there when suddenly raining or cloudy, he might be interested in sending a special price offer in such conditions (to attract some additional visitors).

The prospective *application users* are several:

First of all, a visitor might check what he can do now/today in the visited city. The result should take into account both weather-based recommended options and any available offer. This kind of user should not be required to get registered in the platform, since most of prospective users would not register for a short visit.

Secondly, a local user might be interested in receiving suggestions and offers in his city continuously; he should register for this purpose. He also has the chance to indicate under which weather conditions he is interested in doing leisure activities, to therefore receive customised alerts in his email account.

Finally, another kind of user would be future visitors who might check what is possible to visit in a location before they go there. For these visitors, the activities might be based on the weather forecast or on the weather historic data when no forecast information is available.

We would like to highlight that the system should be readily accessible both from a computer and a mobile device.

B. *Architecture*

If we start from the top-left of Figure 1, we can see that weather information is constantly arriving to the system (1); this information might be provided by sensors or any other event producer element; we have used web suppliers.

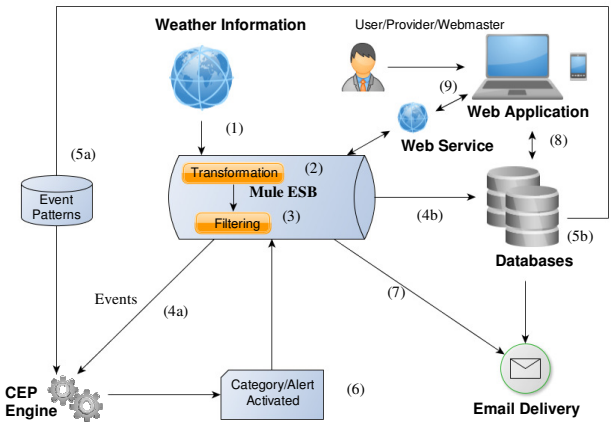


Figure 1. System Architecture

Therefore, in this system, the events reaching the system consist of the weather information. This information is transformed in the required objects of the system (2) and filtered to keep only the information/events of interest (3). Then these events are automatically redirected to the Esper CEP engine [24] (4a) in order to detect the predefined complex events pattern, as well as to a non-SQL database (4b) in order to keep a history table. Such patterns (5a) and their correspondence to the system weather categories have been designed by the system manager and stored permanently in the database (5b); besides they can be updated at any time.

If the patterns of interest are detected then two things happen: on the one hand categories are activated for their use in the web site (6); on the other the corresponding alert emails are sent to the subscribed users (7).

The provider has previously established the conditions for the alerts to be triggered or the activities to be offered at special prices (8). Activities and their associated conditions are stored in a SQL database.

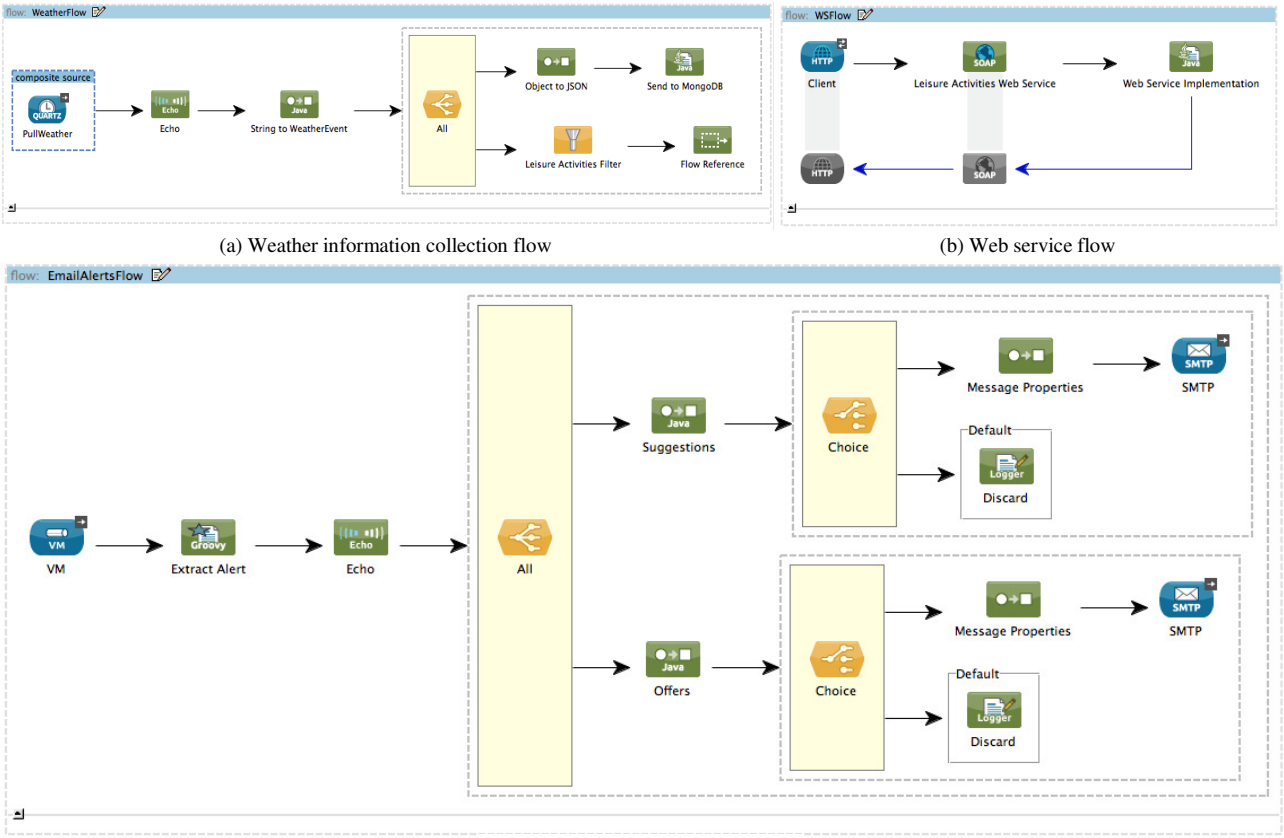
The web service is used as the intermediary between the user and the system. The user, when looking for leisure activities in the web site is transparently invoking the web service which provides this information (9), the latest is already adapted in real time – thanks to our architecture – to the current weather conditions.

C. *Flows in the Enterprise Service Bus*

Most of the business logic of the system has been implemented inside the enterprise service bus, specifically as Mule flows. This provides us with a twofold benefit: on the one hand, Mule facilitates the interoperation of multiple inputs/outputs formats; on the other, all the core functionality will be located together in the ESB. In Figure 2, we have included the most relevant flows in the bus.

Particularly, we have a specific flow for weather information collection/detection, which is shown in Figure 2(a). In this flow, information from several locations is obtained every minute; then it is immediately processed and transformed into the required format and immediately after sent to the non-SQL database, as well as to the complex

 ISBN: 978-1-61208-361-2



(a) Weather information collection flow

(b) Web service flow

(c) Notification flow

Figure 2. Flows in the enterprise service bus

event processing engine. We have filtered the information sent to the engine, so that only relevant information related to current leisure activities in the system reach the CEP engine. Please also bear in mind that additional sources could be added at any time should it be necessary.

Figure 2(b) shows the flow corresponding to the leisure web service. The client (in our case the web site) can invoke the service; the implementation of the service in Mule already takes into account the patterns detected in the engine, adapting the results to the weather conditions.

Finally, Figure 2(c) shows how suggestions and offers are sent based on the alerts triggered by the complex event processing patterns: first of all, we extract the alerts detected by the CEP engine according to the weather patterns defined in the system and to the weather events entering into the latest. Then, based on this information, and on the client interests stored in the system, the corresponding suggestions and offers are sent to their email accounts. The emails sent to the users are by default limited to one per day.

V. APPLICATION OVERVIEW

In this section, we describe the relevant functionality of the resulting application from the point of view of the system manager, the activity provider and the final user.

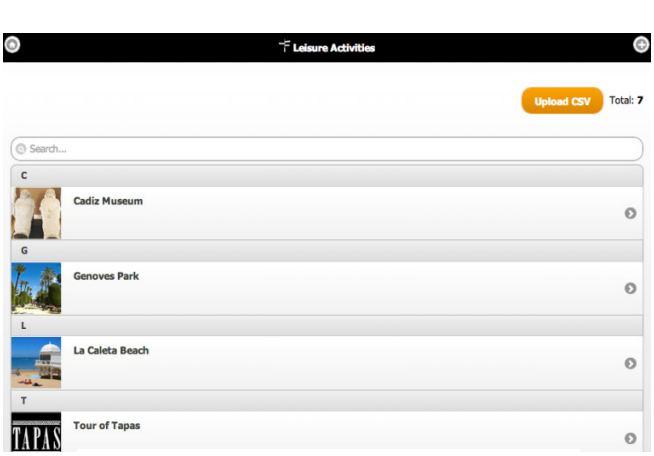
A. *System Manager Role*

The manager (web master) will be the one in charge of administrating categories. Even though the system already includes common categories related to weather situation; the manager will be the one in charge of including new categories – should it be necessary – and the patterns matching the named category. Those patterns have to be defined using EPL of the chosen CEP engine (Esper). The selection of this language was not only based on the efficiency of the CEP engine, but also on its close syntax to the well-known SQL, as well as its native support for multiple event format types. To give an example, if we want to include the category “windy”, an example of pattern for a windy day using Esper EPL would be the following:

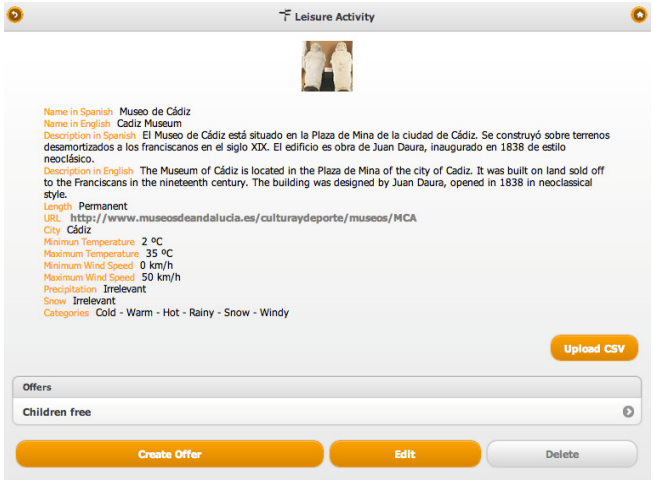
```
@Name("windy")
insert into windy
select 'wind' as alertName, a.windSpeed as
   windSpeed
from pattern [every a =
   WeatherEvent(windSpeed > 50)]
```

The remaining tasks of the system manager/web master would be usual web sites maintenance tasks.

 ISBN: 978-1-61208-361-2



(a) Activities uploaded by the provider



(b) Activity configuration by the provider

Figure 3. Application functionality from the point of view of the activity provider

B. *Provider Role*

The providers will include in the system the different activities. They can fill in all the information manually per each activity or can do it uploading a CSV file (see Figure 3 (a)). The relevant issue here is that they will indicate the weather conditions which will trigger the offers for each activity and the activities which will be provided under particular weather conditions (see Figure 3 (b)).

C. *User Role*

Non-registered user will enter the system and will be able to see the activities to be done now in his location, as well as those which have a special offer for today (see Figure 4).

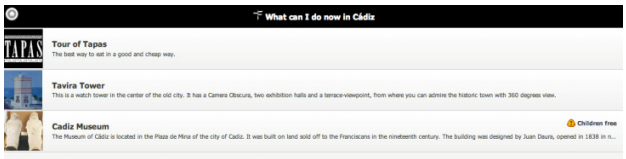


Figure 4. User activity search result

When the user registers he can predefine for which weather categories he wants to receive suggestions and offers (Figure 5).

VI. DISCUSSION

We have presented an application which satisfies both the provider and the consumer: the first one may trigger offers to profit from a larger number of clients when weather conditions are not suitable for his offered activities. The consumer not only receives information about the more suitable activities for current weather conditions but also might benefit from special offers.

It could be thought that limiting your plausible clients might not be good for your business. However, imagine you are happily visiting Granada in winter time and you are about to decide what to do today; the system offers you the option of visiting the Alhambra or going to ski. You go for the second, you rent all the equipment before you go up to Sierra Nevada and once there you discover that all the ski tracks are closed due to the blizzard. What would you do next time? Would it have not been better that the system had not suggested skiing for such a day?

Regarding the limitations of our proposal, we are aware of the difficulty for the system manager to create new event patterns in the system. This is why (1) we provided a large set of categories predefined in the system and (2) we pretend to integrate this architecture with other results of our research: the user-friendly editor for complex event pattern which will generate and deploy automatically the code in the CEP engine for the patterns to be detected [25].

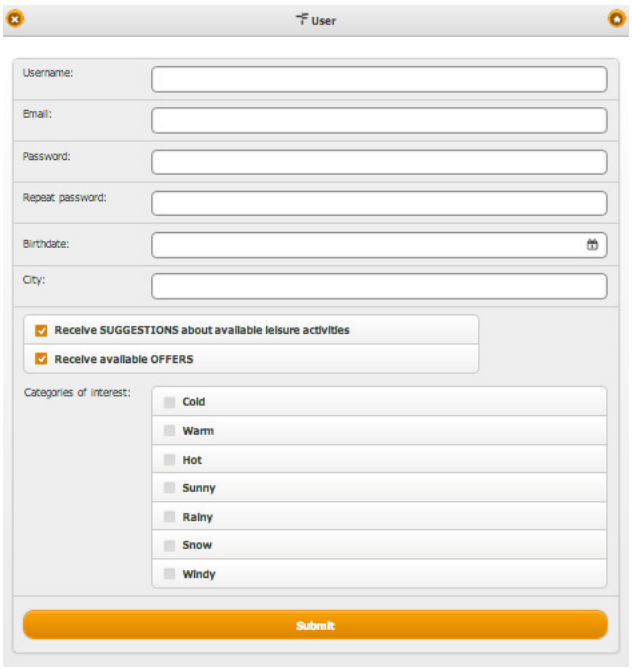


Figure 5. User preferences configuration

 ISBN: 978-1-61208-361-2

VII. CONCLUSION AND FUTURE WORK

In this paper, we have presented an event-driven service-oriented architecture and case-study implementation for providing a context-aware leisure service. Thanks to the use of an ESB to connect the different inputs and outputs of the system and to the use of a CEP engine, we can provide the activities adapted to the weather conditions in real time. Even more, activity providers benefit from a system which may send offers and suggestions to prospective clients based on weather condition real time changes, therefore improving their revenues, as well as the users satisfaction.

In our future work, we plan to extend the architecture with additional features which facilitate different contexts dealing and adaptation mechanism. As we mentioned before, we also pretend to integrate this architecture with a user-friendly editor for event pattern definition.

ACKNOWLEDGMENT

G. Ortiz and J. Boubeta-Puig acknowledge the support from the Spanish Ministry of Science and Innovation under the National Program for Research, Development and Innovation, project MoD-SOA (TIN2011-27242).

REFERENCES

[1] T. Laakko and T. Hiltunen, "Adapting Web Content to Mobile User Agents," *IEEE Internet Comput.*, vol. 9, no. 2, pp. 46–53, Mar. 2005.

[2] I. Mohomed, J. C. Cai, S. Chavoshi, and E. de Lara, "Context-aware interactive content adaptation," in *Proceedings of the 4th international conference on Mobile systems, applications and services - MobiSys 2006*, Uppsala, Sweden, 2006, p. 42.

[3] M. Keidl and A. Kemper, "Towards context-aware adaptable web services," presented at the 13th international World Wide Web conference on Alternate track papers & posters, New York, NY, USA, 2004, pp. 55–65.

[4] A. Bucchiarone, R. Kazhamiakin, C. Cappiello, E. di Nitto, and V. Mazza, "A context-driven adaptation process for service-based applications," presented at the 2nd International Workshop on Principles of Engineering Service-Oriented Systems, New York, NY, USA, 2010, pp. 50–56.

[5] Q. Z. Sheng and B. Benatallah, "ContextUML: a UML-based modeling language for model-driven development of context-aware web services," in *International Conference On Mobile Business*, 2005, pp. 206–212.

[6] G. Ortiz and A. García De Prado, "Improving device-aware Web services and their mobile clients through an aspect-oriented, model-driven approach," *Inf. Softw. Technol.*, vol. 52, no. 10, pp. 1080–1093, Oct. 2010.

[7] G. Ortiz and A. García de Prado, "Web Service Adaptation: A Unified Approach versus Multiple Methodologies for Different Scenarios," presented at the Fifth International Conference on Internet and Web Applications and Services (ICIW), 2010, pp. 569 –572.

[8] D. C. Luckham, *The power of events: an introduction to complex event processing in distributed enterprise systems*. Addison-Wesley, 2002.

[9] B. Sosinsky, *Cloud Computing Bible*. John Wiley & Sons, 2011.

[10] Q. Z. Sheng, S. Pohlenz, J. Yu, H. S. Wong, A. H. H. Ngu, and Z. Maamar, "ContextServ: A platform for rapid and flexible development of context-aware Web services," in *International Conference on Software Engineering*, 2009, pp. 619–622.

[11] G. Ortiz, J. Boubeta-Puig, A. García de Prado, and I. Medina-Bulo, "Towards Event-Driven Context-Aware Web Services," in *Adaptive Web Services for Modular and reusable Software Development: Tactics and Solutions*, IGI Global, 2012, pp. 148–159. DOI: 10.4018/978–1–4666–2089–6.ch005.

[12] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," London, UK, 1999, pp. 304–307.

[13] A. García de Prado and G. Ortiz, "Context-Aware Services: A Survey on Current Proposals," in *The Third International Conferences on Advanced Service Computing*, Rome, Italy, 2011, pp. 104–109.

[14] H. Taylor, Ed., *Event-driven architecture: how SOA enables the real-time enterprise*. Upper Saddle River, NJ: Addison-Wesley, 2009.

[15] J. Boubeta-Puig, G. Ortiz, and I. Medina-Bulo, "An Approach of Early Disease Detection using CEP and SOA," in *Service Computation 2011, The Third International Conferences on Advanced Service Computing*, 2011, pp. 143–148.

[16] Y. Taher, M.-C. Fauvet, M. Dumas, and D. Benslimane, "Using CEP technology to adapt messages exchanged by web services," New York, NY, USA, 2008, pp. 1231–1232.

[17] Y. Xu, P. Wolf, N. Stojanovic, and H.-J. Happel, "Semantic-based Complex Event Processing in the AAL Domain.," in *ISWC Posters&Demos*, 2010, vol. 658.

[18] F. Li, S. Sehic, and S. Dustdar, "COPAL: An adaptive approach to context provisioning," 2010, pp. 286 –293.

[19] D. Carlson and L. Ruge, "Towards Augmenting Legacy Websites with Context-awareness," presented at the 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services,, Tokyo, Japan, 2013.

[20] K. M. Chandy and W. R. Schulte, *Event Processing: Designing IT Systems for Agile Companies*. USA: McGraw-Hill, 2010.

[21] Q. Z. Sheng, J. Yu, A. Segev, and K. Liao, "Techniques on developing context-aware web services," *Int. J. Web Inf. Syst.*, vol. 6, no. 3, pp. 185–202, 2010.

[22] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, "Developing a context-aware electronic tourist guide: some issues and experiences," 2000, pp. 17–24.

[23] R. A. Abbaspour and F. Samadzadegam, "Building a context-aware mobile tourist guide system based on a service oriented architecture," *Int Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XXXVII, no. B4, pp. 871–874, 2008.

[24] E. Inc, *Esper - Reference Documentation*. 2014.

[25] J. Boubeta-Puig, G. Ortiz, and I. Medina-Bulo, "A model-driven approach for facilitating user-friendly design of complex event patterns," *Expert Syst. Appl.*, vol. 41, no. 2, pp. 445–456, Feb. 2014.

 ISBN: 978-1-61208-361-2