2010 Seventh International Conference on Information Technology

# Service Provisioning Support for Non-Technical Service Clients

Luiz Olavo Bonino da Silva Santos, Luís Ferreira Pires, Marten van Sinderen
*Department of Computer Science*
*University of Twente*
*Enschede, The Netherlands*
*(l.o.bonino, l.ferreirapires, m.j.vansinderen)@ewi.utwente.nl*

***Abstract*—Recently paradigms such as Service-Oriented and Pervasive Computing are merging in scenarios where users are surrounded by a plethora of computing devices and available services. Dealing with this potentially large number of devices and services can become overwhelming to users without appropriate software support. Moreover, in the case of non-technical users, an additional difficulty is to express service requests using technical concepts such as data types, XML documents, etc. In this paper we present the architectural design of a software platform aiming at supporting the service provisioning for non-technical users. The platform also makes use of the surrounding computing devices to gather contextual information that helps in the tasks of service discovery, selection and composition.**

## I. Introduction

Service-Oriented Computing (SOC) is an emerging paradigm for distributed systems' architecture, design and deployment. The vision of SOC is that services represent distributed pieces of functionality that can be combined (composed, in SOC terms) to generate new (and more complex) functionality [1]. In an ideal scenario based on this vision, a human service requester expresses requirements to a software infrastructure, and the later then discovers, selects and invokes services without the need of further human interaction. Non-functional properties such as cost, trust, privacy, etc. should also be stated by the and resolved automatically by the infrastructure.

Although this vision is the ultimate goal of SOC, more work still has to be done to realize this vision. Scenarios with significant numbers of available services, service providers and service service requesters, may give rise to issues such as: *(i)* how to express service requests in a more intuitive way (suited for non-technical end-users); *(ii)* how to tackle semantic interoperability issues among services and service requests that use different conceptual models; and *(iii)* how to support the discovery, selection and invocation of services that fulfill the user's goals in the least disruptive and invasive manner.

The SOC vision also overlaps with some of the characteristics of Pervasive Computing. In his seminal paper about Pervasive Computing (formerly known as Ubiquitous Computing) [2] Weiser foresaw that computing, sensing and communication devices would be transparently embedded in our surrounding environment. These computer-enriched environments would grant access to information and services everywhere and anytime. Readily available information can contribute to the realization of the SOC vision specially by allowing a software infrastructure to gather information related to service execution without needing direct user interaction.

In this work we are particularly interested in scenarios where non-technical users are surrounded by computer-enabled devices and sensors, and a large number of services is available. In these scenarios, additional support should be provided to the end-users to help them deal with the (possibly) overwhelming amount of decisions and interactions regarding service provisioning steps, namely, service request specification, service discovery, selection, agreement, composition and invocation.

In this paper we present the Context-Aware Service Platform, which is a service platform aimed at supporting service provisioning to non-technical users. The main benefits of the platform are to allow users to express their service requests using concepts closer to their natural perception and to reduce the need of direct user interactions with the services. The paper also discusses the motivation and requirements for this platform and provides an example to illustrate how such a platform can be used.

This paper is further structured as follows. Section II presents use case scenarios for the proposed platform that we have used to identify the platform's stakeholders and its requirements. Section IV presents the stakeholders we have identified by analyzing the use case scenarios, and Section IV presents the requirements derived from this analysis. Section V presents the architecture of the proposed service platform and its functional components. Section VI gives conclusions and identifies topics for future work.

## II. Scenarios

Below we present use case scenarios aiming at identifying and characterizing usage patterns that our proposed service platform should support. By analyzing these scenarios we can identify the stakeholders that interact with the service platform and derive the platform's functional and non-functional requirements. The main objective of the service platform is to support service provisioning to end-users in Pervasive Computing environments, i.e., environments

978-0-7695-3984-3/10 $26.00 © 2010 IEEE
DOI 10.1109/ITNG.2010.210

containing a multitude of services and computing devices. Due to space constraints we selected only three use case scenarios that cover the issues we target in this paper.

**Use case scenario #1:** *Having customized ambient comfort*

John wants his ambient comfort settings applied whenever he arrives home or at his working environment. For instance, the lights are set to his preferred intensity and color, and the temperature adjusted according to his likings. Besides the city home, John also owns a beach house and a mountain cabin and he expects that his goal of having customized ambient comfort will be fulfilled regardless of the living and working environments he uses. Moreover, he prefers his comfort configuration is automatically applied in every location.

**Use case scenario #2:** *Receive emergency assistance for epilepsy*

Maria suffers from epilepsy. However, she wants to carry on with her life as normally as possible. On weekdays, every morning she runs in the park near her house. Nowadays she owns a mobile device that can detect an imminent epileptic seizure based on body signals. On the (rare) occasions that her body signals are reaching a critical point, she is warned by the device. In this case she can stop running and try to put herself in a safe position. At the same time, a relative or friend which is closer to her location is informed about the situation of a potentially upcoming seizure so he can go to her location. Her on-duty caregiver also receives the information about her body signals, her location, the name of the relative or friend who has been informed and how far this person is from Maria's location. This information is used by the caregiver to decide whether to send an ambulance or to contact the relative or friend to provide further assistance instructions and gather extra assessment of the situation.

**Use case scenario #3:** *Control the use of medicine by home-bound elderly patients*

Peter and Sofia are an elderly couple living alone at their home. John suffers from high blood pressure and has to take some controlled medicine. His medicine varies in frequency and schedule. Sofia has a mild diabetes that can be normally controlled via her diet and only in exceptional cases she needs to take an insulin shot. Both of them are in the initial phase of senility, presenting occasional memory lapses. Therefore, a mechanism to remind them to take their medicine is in place so they do not need to have to move to a nursing home, which could degrade their quality of life.

## III. Stakeholders

Based on our use case scenarios, we can identify a set of stakeholders that would interact with a service platform that facilitates service provisioning:

- *Service Client*. Responsible for requesting the service provisioning. The Service Client also deals with possible negotiations over the service provisioning terms, e.g., a frequent traveler can negotiate with an airline for discounts on a bulk purchase of tickets, or a company can get a faster delivery of supplies after negotiating a transport service. In our work we distinguish a Service Client, which contracts a service, from a Service Beneficiary, which receives the benefit of the service delivery and may not be the same stakeholder. For example, a parent contracts the education services of a school for his child, while the direct beneficiary of the service is the child. However, for simplicity in this paper we consider that the Service Client is also the direct Service BSeneficiary.
- *Service Provider*. Responsible for the service provisioning, the Service Provider advertises its offered services and commits to performing certain activities once a service is contracted by the Service Client and set for delivery. Similarly to the differentiation between Service Client and Service Beneficiary, here we differentiate between Service Provider and Service Executor. The former is the actual responsible (and liable) for the service w.r.t. the Service Client. The later is the stakeholder responsible for actually performing the activities related to the service. For example, a cleaning company *X* offers corporate cleaning services and has been contract by bank *Y* to clean its headquarters. Due to internal personnel problems of company *X*, the cleaning of the bank's headquarters is being performed by freelancers temporarily hired by company *X*. The situation where the service provider and the service executor are different entities raises issues related to liability, trust, security, privacy, etc. Since these discussions fall outside the scope of this paper, hereafter we assume that service providers also execute their offered services, for the sake of simplicity.
- *Context Provider*. Responsible for supplying mechanisms that allow the supporting platform to request and transparently gather contextual information of users. The contextual information is used by the platform to reduce the need for direct user interaction. These mechanisms include information gathered from user's software-based data such as profiles, calendar events, appointments, travel bookings, etc., or from sensor devices such as location (from motion detectors, gps, etc.), blood pressure, heart rate and weight, among others.

## IV. Platform requirements

We analyzed the use case scenarios and the stakeholders to come up with a set of functional and non-functional requirements of the platform that are relevant for the purposes of this paper. The main requirements are briefly decribed as follows:

1) *Domain independence*. The use case scenarios presented in this paper relate to two different domains,

namely, ambient intelligence and home health care. Scenarios related to other domains are also being considered in our work. Therefore, the supporting service platform should be able to operate in different domains while keeping the same functional properties and benefits for its users.

2) *Reduced user interaction*. The platform should achieve a reduction on user's interaction. Since we are considering Pervasive Computing and Services environments, constant requests for user interaction when devices and services need some information would lead to undesirable disruptions of the users' routine. The supporting platform should make use of context-aware mechanisms to gather necessary information aiming at reducing user interaction.
3) *Abstract service request*. By targeting our platform's support on non-technical end-users, we impose a restriction on how the users request service provisioning. We claim that non-technical end-users would have difficulties specifying service requests using current computer-based service technologies such as WSDL [3] and WSMO/WSMX [4], [5]. These difficulties relate to mandatory use of computer-related technicalities such as data types, XML formatting, URLs, URIs, ports, among others, to specify a service request and interact with the discovered services [6]. Therefore, to be able to appropriately support non-technical end-users, a service platform should provide an intuitive way of requesting services.
4) *Semantics*. To allow inference and reduce issues related to semantic interoperability, the interactions between the supporting service platform and its users should be semantically enriched. Furthermore, the internal operation of the platform is expected to benefit from the provided semantics. For instance, when searching for a service using a set of parameters, the platform can find candidate services whose parameters are not exact matches but are close enough, by applying subsumption [7].
5) *User support*. The platform should support all its users with interfaces, APIs and tooling according to each user's objectives. The service client should be supported according to its technical expertise and based on the domain's needs. For instance, a service client in the home health care domain such as Peter and Sofia from use-case scenario 3, could interact with the supporting platform through their TV set, facilitating the visualization of the interface items. In contrast, supposing Maria (from use-case scenario 2) is technologically savvy, she interacts with the supporting service platform through a web interface on her computer as well as through her smartphone.
   Service Providers require tools to support them in tasks such as service description publishing and maintenance (insertion, update and deletion) and semantic annotation of the service descriptions. Moreover, the supporting platform can provide feedback on service usage, and information about services that have been requested by users but have not been offered by any Service Provider.
   Context Providers require tooling support to manage the registration of their provided contextual information as well as to provide semantical annotations for the provided information.
6) *Modularity*. The platform should be designed to be modular in order to allow the substitution of particular elements according to evolution of the requirements, changes in the available technologies and specificities of applications domains. The change of Service Client's interfaces described on requirement 5O is an example of the possibilities supported by a modular design.

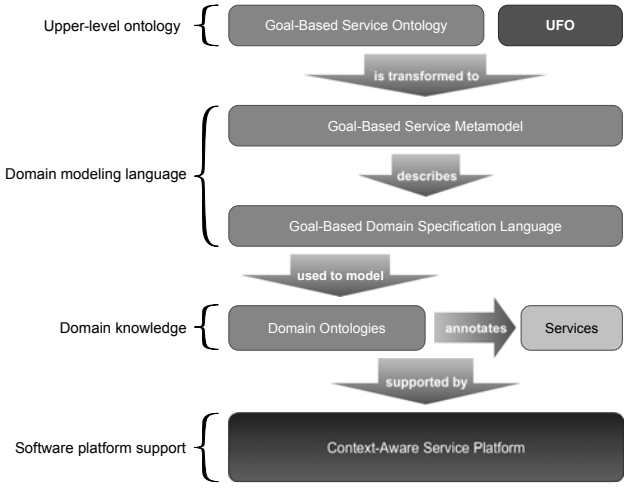


Figure 1. Framework for service provisioning support

## V. Platform Architecture Design

Considering our requirements, in particular requirement 4, we have identified a new stakeholder, namely the Domain Specialist. In our platform, semantics are provided by specifying domains in terms of ontologies. Therefore, the Domain Specialist is responsible for defining Domain Ontologies and submitting them to the platform . The supporting service platform should provide facilities to help Domain Specialists to accomplish the tasks of defining and managing their Domain Ontologies.

Figure 1 shows the Goal-Based Service Framework (GSF) [8] in which our platform is embedded. The main elements of the GSF are briefly introduced as follows:

- *Goal-based Service Ontology (GSO)*. This ontology defines domain-independent concepts such as service,

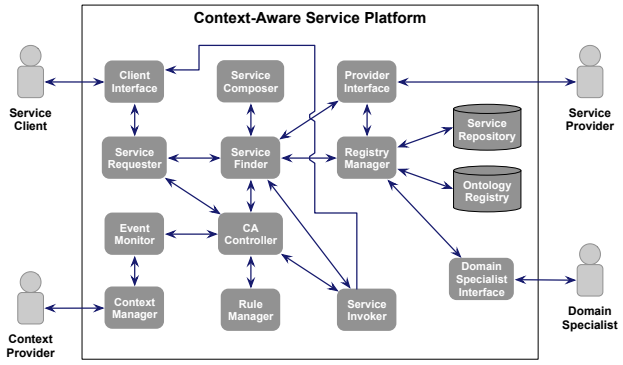


Figure 2. Architectural design of CASP

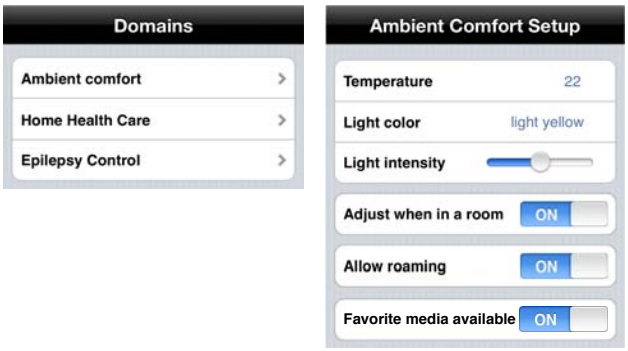


Figure 3. Service Client's GUI application screenshots

stakeholder, organization, goal and task, and their relations. These definitions are further used and specialized in the domain ontologies. GSO extends the Unified Foundational Ontology (UFO) [9] by adding concepts related to SOC and by relating goals, tasks and services.

- *Domain Ontologies*. Based on the goal-based service ontology concepts, domain ontologies are defined to provide shared knowledge about particular domains. The domain ontologies define domain-specific concepts, the relations among these concepts, the valid goals for that domain, valid tasks in that domain and how they relate to the domain goals.
- *Context-Aware Service platform (CASP)*, which supports interactions between service providers and service clients. From the service provider's perspective, the platform supports the publication of service descriptions. From the service client's perspective, the platform provides mechanisms for service discovery, composition, invocation and monitoring, among others.

Based on the requirements presented and discussed in Section IV, we propose the architecture design of the CASP depicted in Figure 2. We have separated the platform's components in three main areas and describe them as follows.

### A. Stakeholders' Interface Components

Figure 2 shows that the CASP supports the interactions of its stakeholders by providing a set of interface components, namely, Client Interface (for Service Clients), Provider Interface (for Service Providers), Domain Specialist Interface (for Domain Specialists) and Context Manager (for Context Providers). These interface components provide APIs that allow GUI applications to interact with the platform. Stakeholders interact with the platform by using either the GUI applications or directly through the APIs.

The Provider Interface component's API offers methods to retrieve available domain ontologies (to be used to semantically annotate the service descriptions), manage the service descriptions' registration (add, update and delete) and to manage the registration of the Service Provider to the platform. The Service Provider's registration is used to control the ownership of the published services and to give feedback to the providers when a client requests a service that cannot be found in the service registry. In this way the Service Providers can have the opportunity to supply a pending demand for services.

The Domain Specialist Interface component's API offers methods to manage the registration of Domain Specialists and of domain ontologies (add, update and delete). Domain ontologies are modeled using the Goal-Based Domain Specification Language (GDSL) which appears in Figure 1. Moreover, the CASP includes a GDSL editor to help Domain Specialists define domain ontologies.

The Context Manager component's API offers methods to manage the registration of the Context Providers, to manage the registration of the contextual information they provide and to retrieve available domain ontologies (which should be used to semantically annotate the contextual information descriptions).

The Client Interface component's API offers methods that allow Service Clients to submit their service requests, to receive the results of the service execution and to enter information required by the services but could not be gathered by the platform as contextual information. In GSF we use the concept of Goal as an abstraction to represent the client's service request. Goal is the propositional content of a service client's intention, i.e., a service client not only wishes something to be accomplished but is committed to its fulfillment [10]. In our approach, we represent a goal by a description of a particular state of affairs that satisfies the goal. For instance, the goal of having your house's ambient comfort set can be satisfied by specifying the room's temperature and lighting settings. Figure 3 depicts our prototype Service Client's GUI application for the Ambient Comfort domain.

Figure 2 also shows that the CASP has two main repositories, namely the Service Registry for storing service descriptions and the Ontology Repository for storing domain ontologies. These two repositories are accessible through the Registry Manager, which provides methods to other

components for retrieving, updating and deleting service descriptions and ontologies, while abstracting from the technical details of the repositories' implementations. In our prototype the domain ontologies are represented using OWL [11] and the service descriptions in SQSDL [12]. The choice for these languages is justified by the availability of tools. In our work we used the Fusion Semantic Repository [13] as service registry, WSMO Editor [14] for semantic annotation of service descriptions by service providers and OWLIM [15] as ontology repository.

### B. Service Provisioning Components

After receiving the Service Client's goal (represented by the specification of a state of affairs that satisfies the goal), the Client Interface component forwards it to the Service Requester component. The Service Requester is responsible for generating a service request using the CASP's internal format which contains not only the goal to be fulfilled but also the provided inputs and pre-conditions, and expected effects and post-conditions. We assume that a service fulfills a service client's goal if the state of affairs resulted from the outcome of its execution (i.e., its effects) matches the state of affairs defined by the service client representing its goal.

To generate the service request the Service requester component queries the Context-Aware Controller for the available contextual information that could be used as services' inputs. After being generated, the service request is submitted to the Service Finder which proceeds to discover candidate service(s). In case no single service could fully comply with the service request, a composition is requested to the Service Composer component. To compose the services, the Service Composer uses information present in the domain ontology regarding the processes acceptable in that domain that fulfill user's goals. The process information can be structured in a hierarchy of processes/sub-processes, giving the Service Composer a template for service composition. For instance, if no service could be discovered to fully book a trip in a travel domain, the Service Composer gather the process information from the domain ontology defining that the book a trip process is composed of the "book a flight", "book a hotel" and "book a car" sub-processes. Therefore, the Service Composer can continue to search for services that provide the functionality of these sub-processes.

Once the service's pre-conditions have been met, the service invocation is performed by the Service Invoker. Before invoking the service, the Service Invoker checks whether the contextual information is still valid for the pre-selected services. For instance, a client could have requested services providing information about nearby restaurants before meal times and the platform pre-selected a service giving this information for the region of the client's residence. However, it may be the case that the client had a business trip to another city not covered by the original service. The Service Invoker detects this situation change (using contextual information) and order a new service discovery to the Service Finder. After service invocation, the Service Invoker submits the service's outputs to the Client Interface to properly inform to the service client. The Service Finder, Service Composer and Service Invoker components are extensions of the DynamiCOS semantic service discovery and composition platform discussed in [16].

### C. Context-Aware Components

The context-aware components are responsible for gathering contextual information and providing it to the other platform's components. The platform uses contextual information to *(i)* increase the accuracy and suitability of the selected services, and *(ii)* provide service's input information. The Context-Aware Controller receives a list of requested pieces of information (e.g., John's location or room's temperature). The contextual information can be requested for single use, or can be subscribed for if the platform needs constant update of that information.

When the requested contextual information is received, the Context-Aware Controller forwards it to the Event Monitor component that queries the Context Manager for the availability of this information. Context Providers register their supplied contextual information through the Context Manager. Registration details contain the provided information, update frequency, etc. In the case subscribed contextual information is requested, the Context-Aware Controller generates an Event-Control-Action (ECA) rule containing the requested information and when and how frequent this information is required. This rule is managed by the Rule Manager component. These context-aware components that have been built in our prototype are extensions of the Context Management Service and Awareness and Notification Service [17]

## VI. Conclusions and future work

In this paper we presented the Context-Aware Service Platform, which embodies our approach to support service provisioning for non-technical users. Moreover, we identified the platform's stakeholders and discussed their requirements. The main contributions of CASP are the support to service provisioning for non-technical users and the use of contextual information. The former is achieved by allowing users to express their service requests by means of their goals. This brings the concepts related to service request closer to the user's conceptualization, instead of forcing users to use technical concepts such as data types, XML-based documents, etc. as in the traditional approaches. The use of contextual information is used to both increase the accuracy and suitability of the selected services, and reduce the need of direct user interaction by gathering information that is used as input to services.

In the current stage, the service provisioning and the context-aware components of the CASP have been implemented in the scope of the DynamiCOS framework [16] and the Amigo's ANS [17]. The integration of these two sets of components are underway together with the deployment of the Service Registry, Ontology Repository, Registry Management and Stakeholders' Interface components. Further work include the validation of the CASP in a realistic setting.

## REFERENCES


[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing research roadmap," European Union Information Society Technologies (IST), Directorate D, Tech. Rep., 2006. [Online]. Available: http://infolab.uvt.nl/pub/papazogloump-2006-96.pdf

[2] M. Weiser, "The computer for the 21st century," *SIGMOBILE Mobile Computing and Communications Review*, vol. 3, no. 3, pp. 3–11, 1999.

[3] "Web services description language (wsdl) version 2.0 part 0: Primer." [Online]. Available: http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/

[4] J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, M. Kifer, B. König-Ries, J. Kopecky, R. Lara, E. Oren, A. Polleres, J. Scicluna, and M. Stollberg, "Web service modeling ontology (wsmo)," http://www.wsmo.org/TR/d2/v1.3/20061021/, October 2006.

[5] T. Haselwanter, P. Kotinurmi, M. Moran, T. Vitvar, and M. Zaremba, "Wsmx: A semantic service oriented middleware for b2b integration," in *ICSOC*, ser. Lecture Notes in Computer Science, A. Dan and W. Lamersdorf, Eds., vol. 4294. Springer, 2006, pp. 477–483.

[6] C. Rolland, R. S. Kaabi, and N. Kraïem, "On isoa: Intentional services oriented architecture," in *CAiSE*, ser. Lecture Notes in Computer Science, J. Krogstie, A. L. Opdahl, and G. Sindre, Eds., vol. 4495. Springer, 2007, pp. 158–172.

[7] D. L. McGuinness and A. Borgida, "Explaining subsumption in description logics," in *IJCAI (1)*, 1995, pp. 816–821.

[8] L. O. Bonino da Silva Santos, E. Gonçalves da Silva, L. Ferreira Pires, and M. van Sinderen, "Towards a goal-based service framework for dynamic service discovery and composition," in *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, 2009.

[9] G. Guizzardi, "Ontological foundations for structural conceptual models," Ph.D. dissertation, University of Twente, 2005.

[10] L. O. Bonino da Silva Santos, G. Guizzardi, R. Silva Souza Guizzardi, E. Gonçalves da Silva, L. Ferreira Pires, and M. J. van Sinderen, "Gso: Designing a well-founded service ontology to support dynamic service discovery and composition," in *2nd International Workshop on Dynamic and Declarative Business Process (DDBP 2009)*, September 2009.

[11] "Owl 2 web ontology language primer," October 2009. [Online]. Available: http://www.w3.org/TR/2009/REC-owl2-primer-20091027/

[12] J. Farrell and H. Lausen, "Semantic annotations for wsdl and xml schema," August 2007. [Online]. Available: http://www.w3.org/2002/ws/sawsdl/

[13] D. Kourtesis and I. Paraskakis, "Web service discovery in the FUSION semantic registry," in *BIS 2008 - 11th International Conference on Business Information*, ser. Lecture Notes in Business Information Processing, W. Abramowicz and D. Fensel, Eds., vol. 7. Springer, 2008, pp. 285–296.

[14] M. Dimitrov, A. Simov, V. Momtchev, and M. Konstantinov, "WSMO Studio — a semantic web services modelling environment for WSMO," in *ESWC '07: Proceedings of the 4th European conference on The Semantic Web*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 749–758.

[15] A. Kiryakov, D. Ognyanov, and D. Manov, "OWLIM - a pragmatic semantic repository for OWL," in *WISE 2005 - Web Information Systems Engineering International Workshops*, ser. Lecture Notes in Computer Science, M. Dean, Y. Guo, W. Jun, R. Kaschek, S. Krishnaswamy, Z. Pan, and Q. Z. Sheng, Eds., vol. 3807. Springer, November 2005, pp. 182–192.

[16] E. Gonçalves da Silva, L. Ferreira Pires, and M. J. van Sinderen, "Supporting dynamic service composition at runtime based on end-user requirements," in *User Generated Services Workshop at the International Conference on Service Oriented Computing (ICSOC 2009)*, November 2009.

[17] L. O. Bonino da Silva Santos, R. Poortinga-van Wijnen, and P. Vink, "A service-oriented middleware for context-aware applications," in *5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2007)*, November 2007.