An Embeddable Fusion Framework to Manage Context Information in Mobile Devices

Ana M. Bernardos, Eva Madrazo, and José R. Casar

Telecommunications Engineering School, Technical University of Madrid,
Av. Complutense 30, 28040, Madrid, Spain
{abernardos,eva.madrazo,jramon}@grpss.ssr.upm.es

Abstract. Conveniently fused and combined with data from external sources, information from sensors embedded in a mobile device may offer a dynamic view of the user's situation, sufficient to build adaptive context-aware services. In order to shorten the development cycle of these applications, an embeddable framework to acquire, fuse and reason on context information is hereby described. 'CASanDRA Mobile' is designed to work autonomously in resource-constrained devices, offering to application developers a transparent management of context information. Based on a service-oriented architecture implemented in mobile OSGi, it offers a scalable infrastructure of bundles which decouple context acquisition and automate context inference from application development. 'CASanDRA Mobile' aims at providing the user with full control on his private context data, by using privacy policies suitable to handle P2P context sharing. To exemplify how to use the framework features, the design procedure for a context-aware wellness application is described.

Keywords: Context-aware system; data fusion; mobile reasoning; activity recognition; mobile framework.

1 Introduction

Since 1994, when Schilit and Theimer coined the term 'context-awareness' [1], a good number of architectures to handle context information have been described in literature [2]. Most of these proposals, which aim at decoupling the process of data acquisition from application development, base their performance on the existence of a centralized infrastructure-based module capable of fusing data to extract context information.

Mobile devices are increasingly equipped with better sensing, processing and storage capabilities, so it is possible to design a light infrastructure-independent middleware to infer context information. Light context-aware systems enabling distributed peer-to-peer context-sharing provides a important feature to implement the 'Internet of Things' concept, by which different types of objects and devices are able to opportunistically communicate among them. It is important to note that this autonomous approach of mobile computing is not opposed to the 'cloud computing' trend, but complements it (e.g. by exploiting short-distance data connections and device off-line autonomous capabilities).

Following we describe our embeddable framework to provide Context Acquisition Services anD Reasoning Algorithms, from now on called 'CASanDRA Mobile'. This middleware is the result of a learning process starting with the design and development of our infrastructure-based system for context-awareness, CASanDRA [3], which implements the fusion architecture described in [4]. Of course, the light version of the system present different challenges, but it is capable of offering a significant collection of features already considered in CASanDRA, while including domain-specific ones.

CASanDRA Mobile bundle a set of standard services, conceptually layered (from data acquisition to high-level fusion), but functionally sharing the same communication and inference resources. The developer can use context information at different levels of abstraction, as the middleware provides APIs to access all the available data. The middleware, developed by using an implementation of a Service Oriented Architecture [5], is prepared to handle P2P context-sharing and, in future, it will provide Quality of Context management in order to auto-regulate its performance.

Section II reviews previous approaches to mobile computing middleware. Section III lists CASanDRA Mobile's standard functionalities and justifies our choice for a SOA-based implementation. Section IV describes the middleware itself and Section V explains a practical implementation of a context-aware application using CASanDRA Mobile's activity inference capabilities. Finally, Section VI considers open issues which need to be addressed in further developments.

2 A Review of Light Architectures for Context Management

Numerous context-aware frameworks have been described in literature during the last decade; some of them are autonomous embeddable systems, designed to be installed in mobile resource-constrained devices. Following there is a chronological short review of some of these proposals; their features, architectural approaches and learned lessons have inspired the current design of CASanDRA Mobile.

Released in 2003 (when developments such as LIME, XMIDDLE, or Mobiware had already addressed general aspects of middleware development for mobile computing), MobiPADS [6] appears as one of the first designs considering context-awareness in mobile middleware. The platform *enables active service deployment and dynamic service composition depending on the context*, in order to optimize – in terms of resources – the overall operation of mobile applications. MobiPADS is composed by two parts: the system components, providing essential management services for deployment and configuration, and the service space, where a series of mobilets (MobiPADS' services) *can be chained* to provide the applications with aggregated functionalities. Mobilets access the components through the mobilet API, which also have interfaces to allow communication and configuration of the system's components.

MobiPADS software claims to be reflective. *Reflection* [7] as design paradigm is also considered to build CARISMA [8]. This middleware offers customized services to applications by using policies, which depend on the context configuration. The middleware provides applications with an API (meta-interface) to inspect and alter the middleware behavior, as encoded in application profiles.

CORTEX [9] focuses on the configuration of a distributed network for context management. The system is composed of sentient objects, independent software components capable of acquiring context data and performing inference. *Information sharing* among neighbors and *dynamic resource discovery* are two of its important features. CASanDRA Mobile also bases is architecture on the composition of dynamic software

structures, adopting the 'reflection' paradigm: the middleware is able to manage and change its internal composition to provide the applications on top of it with the information they need, while minimizing its memory footprint and resource consumption.

The ReMMoC proposal [10] focuses on dealing with the *platform heterogeneity problem* by proposing a web services-based reflective middleware that allows mobile clients to be developed independently of both discovery and interaction mechanisms. In the same direction, the Obje infrastructure [11] uses abstract models common to every object/device in the network, which expose information about how a device can connect to another, provide metadata about itself, be controlled or provide references to other devices. CASanDRA Mobile extends these concepts to its components, which can be automatically discovered in the system. The middleware can then dynamically create and manage the life cycle of customized aggregation services, ready perform unplanned tasks for context-aware applications.

The ContextPhone [12] is a prototyping platform running on mobile phones using Symbian OS. It consists of four interconnected modules: Sensors, Communications, Customizable applications (which can seamlessly augment or replace built-in applications) and System Services. The architecture mirrors the widget approach of the well-known Context Toolkit, using a *publish-subscribe model* for its components. It is possible to add new data types for *context extension* at compile time. Citron [13] presents an alternative for internal data sharing. The framework, conceived to be fully operative in the mobile device, uses a blackboard approach to handle information tuples gathered from 'workers' (components which handle access to sensors). CASAnDRA Mobile opts by the publish-subscribe model, implemented by using the standard features of a SOA platform.

MADAM [14] employs component framework to design applications that can be adapted by reconfiguration; an application is assembled from a recursive structure of component frameworks. The middleware is based on extended goal policies expressed as utility functions, leaving the system to reason on the actions required to implement those policies. For MADAM, it is necessary to *describe the application structure*, the application's variability and distribution aspects, the properties of each variant and the utility functions for comparing variants.

MARKS [15] faces context management in ad-hoc networks. MARKS' authors claim to incorporate some unexplored attributes – such as 'knowledge usability', 'resource discovery' and 'self-healing' to a pervasive middleware for mobile devices, in order to optimize the use of physical resources but to ensure *security and privacy* too. MARKS is composed of core components and services: the former include the object request broker, resource discovery, trust management and universal service access unit.

CASanDRA Mobile brings concepts such as 'reflection', 'resource discovery', 'service chain', 'privacy policies' or 'event-based architectures' together. Additionally, it aims at managing Quality of Context by using bottom-up probabilistic methods. As far as we know, it is the first attempt to benefit from the use of a Service Oriented Architecture [5] based on mobile OSGi (mOSGi), highly modularizable and dynamically configurable in real time operation. mOSGi enable the design of a middleware capable of handling context asynchronous (event-based) communications and supporting off-line performance.

3 CASanDRA Mobile: Functionalities and Approach

CASanDRA Mobile aims at offering a set of standard off-the-shelf features for the development of context-aware applications, in order to accelerate the application design and development life cycle. The framework, thought to be easily and modularly scalable over the time, is component-based and infrastructure-independent. It is ready to maximize its functionalities even when no data connection with external infrastructures is available. For this reason, the middleware needs to offer an off-line functional alternative to services which may need infrastructure support (e.g. indoor positioning systems). CASanDRA Mobile can be used by both native and in-the-cloud applications; it is the developer who determines which kind of context information the application needs to handle.

The middleware includes *P2P capabilities*, mainly driven to make possible context information sharing among different devices equipped with CASanDRA Mobile. Context data will be labeled depending on *privacy policies* controlled by the user; as a consequence, the P2P sharing functionality will handle and expose context data taking into account the privacy restrictions.

The framework will have internal procedures for handling the *quality of context* it manages, in order to optimize the acquisition and processing procedures while considering the computational and resource costs. The applications will be aware of the quality of the information they receive, being able to adapt their behavior depending on the accuracy, up-to-dateness and other QoC features. This implies that uncertainty needs to be controlled over all the context composition lifecycle in a coordinated and reliable way.

The framework will deliver access-to-sensor and fusion features, but also reasoning tools for automatic processing of complex context information. This means that an *inference engine* will be available inside the framework. Initially, this inference engine will be assimilated to a rule-base engine, but the final objective for CASanDRA Mobile is to expand its capabilities and manage elaborated data models (e.g. light ontologies).

CASanDRA Mobile aims at *adapting its components' behavior at runtime*, starting, stopping and hibernating components when needed. Real-time scalability (dynamic discovery of context sources) and easy maintenance are also key features. This means that, for example, *hot installation and remote update of components* (to be done without restarting the framework) will be possible. Finally, components are to be *weakly integrated*, allowing modular and independent software development.

Considering these requirements, we have opted to implement the architecture of CASanDRA Mobile over a SOA implementation (mOSGi). Service Oriented Architectures handle 'services' as software units and use them to implement the key concepts of 'visibility', 'interaction' and 'effect'. As defined in the standard [5], a 'service' needs to be able to perform work for another, specifying the work offered for another and also offering to another to perform the work. For this reason, services need to have interfaces to be externally invoked, and to publish their functionalities for applications to use them. In SOA architectures, modularization improves the reusability of software components and makes parallel development and testing

simpler (each service may be independently developed and mock-uped when not available). With respect to its core functionalities, mobile OSGi enables automatic service registration and component management, and allows hot deployment of new services, not requiring a stop-start procedure when updating the service offering.

4 Description of CASanDRA Mobile Middleware

4.1 Introduction to CASanDRA Mobile's Design

The architecture of CASanDRA Mobile is composed by three building blocks - Acquisition Layer, Context Inference Layer and Core System. The Acquisition Layer decouples the access to embedded and external sensors from upper processing levels by using software 'Sensors', which deal with low-level hardware information retrieval. The Context Inference Layer gathers a number of 'Enablers' - modules that process data coming from 'Sensors', fuse them, and infer complex context parameters. Finally, the Core System provides several features to integrate these components in the middleware, such as discovery and registry management of new elements and some common utility libraries. Both 'sensors' and 'enablers' publish their output data in the middleware through an event manager. The main difference between these two types of components is that the formers only act as data providers, not being ready to consume data from other components. Applications run on top of CASanDRA Mobile middleware, consuming context information provided by Enablers and Sensors and using its standard features.

Regarding information retrieval, CASanDRA Core offers a set of APIs to handle 'subscription-based' and 'on-demand' information queries. In the first case, a component (or an application) can subscribe to a set of context parameters: this method makes possible to receive periodic updates for the selected context data via asynchronous communications (events). Consumer elements are able to configure context patterns in order to combine and filter notification events.

CASanDRA Mobile middleware is implemented in Java; it runs on a mobile OSGi platform based on J9 inside a Windows Mobile (WM) device. OSGi is a Dynamic Module System for Java, handling modules called 'bundles', cohesive, self-contained units, which explicitly define their dependencies to other modules/services and their external service API. OSGi improves encapsulation and reusability, simplifies the implementation of a modular system, and provides a very useful diversity of standardized optional services including the logging service or the *eventAdmin* service to manage events. Additionally, several implementations of mOSGi frameworks are available for mobile operative systems such as Symbian or Android, so CASanDRA Mobile concept can be flavored to cover other type of devices beyond WM.

4.2 CASanDRA Mobile's Components

Fig. 1 shows the general architecture of CASanDRA Mobile and its APIs.

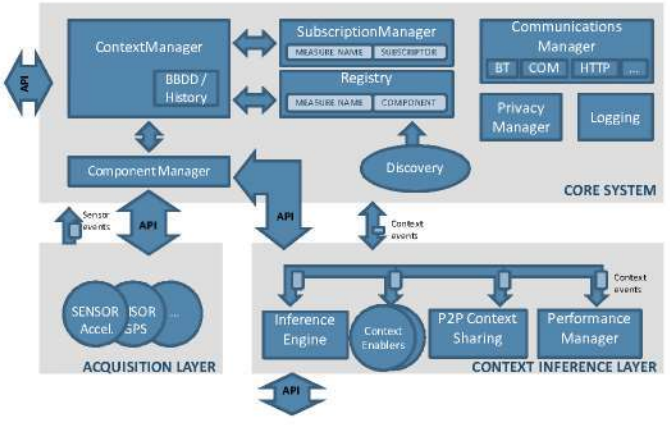

Fig. 1. Software components

CASanDRA Core System. CASanDRA Mobile main modules are encapsulated together in the Core System bundle, which controls the components' integration and their lifecycle and facilitates. In brief, the Core System is composed by:

1. *The Context Manager*, the main module in the Core bundle, stores and manages the publication of context parameters. It controls when a subscription is done or removed and consequently asks the Component Manager to initiate or stop the needed components. Components are started and stopped in a lazy manner, that's to say that the middleware only starts a component when necessary and stops it when it is not needed by any of the components in the middleware. The objective is to adapt the structure of the framework to the consumer application needs, keeping the component deployment as simple as possible, in order to improve the middleware performance. This module provides an API allowing the access to stored measures/context parameters and another to require on demand measures. Context information is stored in name-value tuples aggregated to compose *measure* objects (for instance, the three axis acceleration values compose an acceleration 'measure'). This facilitates the management and the storage of measures in an object-oriented database (db4o).
2. *The Component Manager* manages the components life cycle according to the needs of active applications. It is able to start, stop or configure components under the supervision of the Context Manager.
3. *The Subscription Manager* allows components and applications to subscribe to a measure/context parameter. It is aware of every active subscription (having information about the subscribed component/application), and provides an API to retrieve subscription data.
4. *The Registry* gathers all the available context measure names, together with the component that publishes each measure. The Registry API allows components and applications to ask for available measures/context parameters at any moment.
5. *The Discovery* module listens to new components registration queries and adds these components to the Registry. Components must use some special parameters

when registering, so the discovery module can effectively make them available in the middleware.

6. *The Communications Manager* centralizes the access to available communication interfaces in the mobile device: every component needing a Bluetooth connection, COM ports, or HTTP connections, will use this library. It also includes additional features; for instance, the Bluetooth Manager performs the periodic search for new near devices, and provides an API used by the P2P Context Sharing for this purpose.
7. *The Logging Module* includes some basic logging facilities.
8. *The Privacy Manager* controls the access to the user's private data. It stores credentials and privacy profiles, and manages application authorizations. This is especially useful in order to share context parameters with other devices via P2P context sharing.

CASanDRA Core is packaged in a single OSGi bundle. This core API is sufficient to develop sensor components, enablers and applications, and to make them work together.

CASanDRA Mobile Sensors and Enablers. 'Sensors' and 'Enablers' are CASanDRA's components. A component is a bundle implementing the *ComponentInterface* interface, and offering this interface as a service by registering itself in the middleware. When a component is started, it subscribes to all the components providing the needed data to compute its output; automatically, the middleware starts every component in the middleware that generate those data. This chain reaction ends with the startup of the sensors that directly acquire raw data from hardware.

CASanDRA Mobile aims at providing a complete set of 'sensors', software pieces accessing in a customizable way the sensing resources embedded in (or attached to) the mobile device, but also managing connections to retrieve data from external sensors which may be connected through Bluetooth or ZigBee interfaces. For example, CASanDRA Mobile offers sensing modules to retrieve a) acceleration data from an embedded inertial system, b) the received signal strength indicator of a WiFi connection (in order to enable localization algorithms) or c) biometric data from an external BT oxymeter.

'Enablers' performs data fusion at different levels, from signal to situation and impact fusion. The middleware currently includes a *P2P Context Sharing Enabler*, which listens to other devices executing CASanDRA middleware, in order to share with them context parameters (P2P context sharing makes possible to enhance the context image locally treated in each device) and a group of general *Context Enablers*, which e.g. includes a *Location Broker* - to fuse position information coming from different Location Enablers in order to provide seamless position estimation- or an *Activity Enabler* – working on acceleration data to infer activity estimates.

Two additional enablers offer horizontal functionalities. On one hand an *Inference Engine* based on the rule engine 3aplm offers an API to configure rules to be executed when needed, providing with reasoning capabilities to external components. Applications and internal components may subscribe to receive events from the Inference Engine. On the other hand, the *Performance Manager* watches variables such as the available memory or the number of working threads, and generates context parameters in relation to the system state.

Every sensor or enabler registers the measure or context parameter it provides when initialized by using a 'name'. These names are encouraged to follow a coherent taxonomy for improving the use of patterns in subscriptions. For instance, all the components publishing a location context parameter should name it 'location.X' so that any component can subscribe to 'location.*' and receive every location context parameter available (e.g. location.gps.internal, location.wifi, location.bluetooth).

5 Deployment of an Application on Top of CASanDRA Mobile

5.1 Service Scenario: A Wellness Application to Control Sedentary Behavior

The application to develop on top of CASanDRA Mobile is a native context-aware wellness application, which aims at persuading the user to increase his physical activity in order to avoid or minimize sedentary behaviors. Each hour, the application evaluates the user's activity level, and makes a positive or negative verdict. This balance is visually communicated to the user: the scenery serving as wallpaper for the application turns into a dark landscape each day at midnight, which progressively evolves to a greener scene if the user's activity level is adequate. Additionally, the application triggers context-aware notifications in order to help the user to increase its activity when low levels are detected.

The application needs to be aware of the user's daily activity, assumed to be a combination of 'atomic activity estimates' (at rest-walking-running), covered walking distance and time. The user is not wearing any external device but his personal mobile phone, ideally located in a chest-pocket in the user's shirt. This simplified scenario makes possible to estimate the user's movement through the accelerometer in the mobile device. Activity information is combined with GPS data when outdoors and with a WiFi-based location estimates when indoors.

5.2 Development Methodology and Application's Design

CASanDRA Mobile can be easily improved at the same time that applications are built, if some general guides are taken into account when designing and developing the application's building blocks. An important recommendation is to develop every enabler/sensor/application in a separate bundle: this allows specialization, division of labor, component reusability and parallel design and testing life-cycles. Therefore, when building an application, the developer has to define the bundles –sensors, enablers and application bundles- needed to have a full-featured application, and match his needs with the services available in CASanDRA Mobile.

For the proposed scenario, three sensors are needed: 'accelerometerSensor', 'wifirssSensor' and 'GPSSensor'. Then, it is necessary to process sensor data in order to infer the context feature 'activity'. The 'activityEnabler' will be able to discriminate among a set of simple activities such as 'at rest', 'running' or 'walking', by computing the variance of the acceleration signal in each axis (x, y, z). The 'activityEnabler' will be in charge of publishing the activity estimation, in order to be used by the consumer components. The 'wifirssSensor' retrieve signal strength data from the WiFi network, for the 'wifiLocationEnabler' to process them. The 'locationBrokerEnabler' gathers both GPS data and WiFi estimates to provide seamless location information.

The application will be able to use the Inference Engine's API to configure the specific behavioral patterns' checking, in order to make the activity evaluation easier. For example, the application may be interested in getting a notification from the Inference

Engine if no activity and no position change is detected during two hours in a working day. The logic on how to deal with this situation still remains in the application side, which may decide to send an email to the user with some healthy advices in order to a) check that the user is carrying his mobile phone with him and b) foster the user's physical activity. In future, the application may deal with learning methods to cope with uncertainty and minimize notifications.

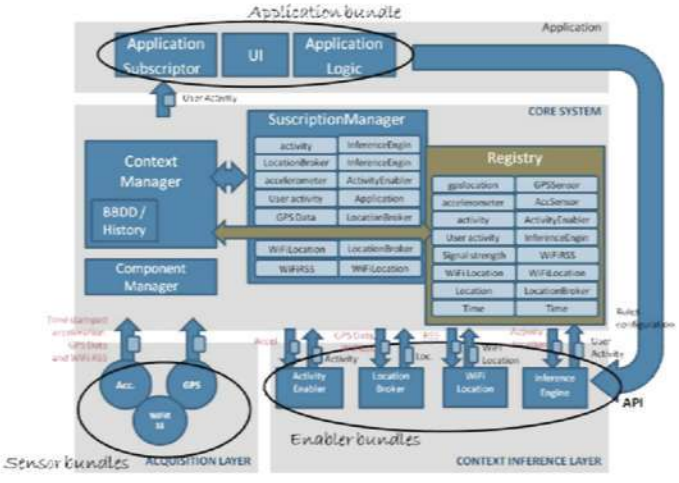

Fig. 2. Bundle deployment in CASanDRA Mobile

The application will be finally deployed in a separate bundle. That makes eight independent bundles (Fig. 2) that can be developed in parallel, using mock components to simulate the others. Final integration is expected to be seamless if every single bundle is adequately tested.

6 Conclusions and Further Work

The development of light fusion strategies for context management remains a challenge: from stable and efficient and accurate context feature extraction, to complex reasoning or uncertainty management and context sharing, there is a way to go to have systems for resource-constrained mobile nodes. Additionally to the specific issues related to context management, common problems of mobile application development (such as portability, modularity or scalability) remain without universal solution.

CASanDRA Mobile is an attempt to address both aspects, delivering a full-featured but light fusion framework for context management in mobile devices. The platform is still in its infancy, but its first version demonstrates the feasibility and convenience of building the framework on the service oriented architecture implemented through mOSGi. We expect to deliver results on performance tests over an intensive consumer of context data application. The framework, which aims at being transparent to the applications developer, works on a general model of resources and context elements, so it defines interfaces which enable that middleware services use new resources as they appear in the environment.

Our current lines of work are currently focused on improving: 1) a light strategy for 'quality of context' control during all the fusion process; 2) a fusion module to manage position estimation in a seamless manner; 3) an stable activity inference system which uses Bayesian logic; 4) a model for context sharing context among different devices with the objective of improving context estimation and 4) a reasoning service including ontology processing. Of course, these are still a small part of the fusion problems to solve in order to have an operative framework.

Acknowledgments. This work has been supported by the Government of Madrid under grant S-0505/TIC-0255 and by the Spanish Ministry of Science and Innovation under grant TIN2008-06742-C02-01.

References

1. Schilit, B.N., Theimer, M.M.: Disseminating active map information to mobile hosts. IEEE Network, 22–32 (September/October 1994)
2. Hong, J.-Y., Suh, E.-H., Kim, S.-J.: Context-aware systems: A literature review and classification. Expert Systems and Applications 36, 8509–8522 (2009)
3. Bernardos, A.M., Tarrío, P., Casar, J.R.: CASanDRA: A framework to provide Context Acquisition Services And Reasoning Algorithms for Ambient Intelligence Applications. In: Proc. Int. C. on Parallel and Distributed Computing, Apps. and Tech., Hiroshima (2009)
4. Bernardos, A.M., Tarrío, P., Casar, J.R.: A data fusion framework for context-aware mobile services. In: Proc. of the IEEE International Conf. in Multisensor Fusion and Integration for Intelligent Systems, Seoul, pp. 606–613 (2008)
5. OASIS Standard, Reference Model for Service Oriented Architecture 1.0 (2006)
6. Chan, A.T.S., Chuang, S.: MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing. IEEE Transactions on Software Engineering 29(12) (2003)
7. Sobel, J.M., Friedman, D.P.: An introduction to reflection-oriented programming. In: Proceedings of Reflection 1996, San Francisco (1996)
8. Capra, L., Emmerich, W., Mascolo, C.: CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications. IEEE T. on SW Engin. 29(10), 929–945 (2003)
9. Sørensen, C., Wu, M., Sivaharan, T., Blair, G.S., Okanda, P., Friday, A., Duran-Limón, H.: A Context-Aware Middleware for Applications in Mobile Ad Hoc Environments. In: Proc. 2nd W. on Middleware for Pervasive and Ad hoc Computing, pp. 107–110. ACM, NY (2004)
10. Grace, P., Blair, G.S., Samuel, S.: A Reflective Framework for Discovery and Interaction in Heterogeneous Mobile Environments. ACM SIGMOBILE Mobile Computing and Comms. Review on Discovery and Interaction of Mobile Services 9(1), 2–14 (2005)
11. Edwards, W.K., Newman, M.W., Sedivy, J.Z., Smith, T.F.: Bringing Network Effects to Pervasive Spaces. Pervasive Computing 4(3), 15–17 (2005)
12. Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. IEEE Pervasive Comp., 51–59 (April-June 2005)
13. Yamabe, T., Takagi, A., Nakajima, T.: Citron: A Context Information Acquisition Framework for Personal Devices. In: Proc. 11th Int. Conf. on Embedded and Real-Time Computing Systems and Apps., pp. 489–495. IEEE Computer Society, Los Alamitos (2005)
14. Alia, M., Horn, G., Eliassen, F., Khan, M.U., Fricke, R., Reichle, R.: A Component-Based Planning Framework for Adaptive Systems. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4276, pp. 1686–1704. Springer, Heidelberg (2006)
15. Sharmin, M., Ahmed, S., Ahamed, S.I.: MARKS for Mobile Devices of Pervasive Computing Environments. In: Proc. 3rd Int. Conf. on Information Tech., pp. 306–313 (2006)