Context-Aware Middleware: An overview

Daniel Romero

Context-Aware Middleware: An overview

Daniel Romero

University of Lille 1, INRIA Lille, Nord Europe, Laboratoire LIFL UMR CNRS 8022
Parc Scientifique de la Haute-Borne 40, Avenue Halley
59650 Villeneuve D'Ascq- France
daniel.romero@inria.fr

Abstract. In this paper we give an overview of context-aware middleware platforms. They are characterized in terms of properties such as communication, context management and adaptation. We present a description of different platforms and their properties.

key words: context-awareness, middleware platform, pervasive computing.

1 Introduction

Today is normal to find different computational entities that are part of environments where people develop their daily activities such as home, office and malls. These entities provide services that can help users in their activities. The entities are also characterized by their heterogeneity. This means that they use different communication mechanisms, connection technologies and data representations, and that they provide information with different processing levels. Some of these devices also change constantly of location producing variation in services availability. All these properties are typical of the so-called pervasive environments.

It would be desirable to access the services present in a pervasive environment throw devices like laptops, smart phones, and PDA increasingly powerful. However, they need some kind of mechanism that enable them to do it. Furthermore, it is also required to detect the changes in the context, which could affect the applications been executed in user devices. In computing, there exist a discipline that deals with this kind of concerns, the "context-aware computing". As in [1] is established, the context-aware computing uses the context in order to provide relevant information and services to users. The context is understood as "any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical or computation object". Inside of this area, a special kind of application has emerged, the context-aware middleware solutions. In general, they provide the same functionality associated with traditional middleware in terms of communication. However, they also provide context management and adaptation to deal with the different resources present in the environments. In this paper we give an overview about these middleware platforms and their different features.

The rest of this paper is organized as follow. In section 2 we present a motivation scenario. Section 3 describes some relevant properties in this kind of middleware. Section 4 presents some context-aware middleware platforms and their characterization in

terms of properties described in section 3. Finally, in section 5 we give some conclusions.

2 Motivation Scenario

In this section we present a simple scenario to contextualize the use of context-aware middleware platforms. Alice, a philosophy student, goes to the mall. She has a smart phone device with a context-aware middleware previously installed. The first time that she gets in the mall, her mobile device detects an available wireless network and accesses it. Once connected, the middleware platform discovers a location service, advertised with the UPnP [2], which shows Alice position in a mall map and allows device to locate the different mall stores. In the mall, Alice remembers a letter that she has to sent. She doesn't have the printed letter but she has the document on her smart phone. Using the mobile device, Alice is able to find a printer service, published using Service Location Protocol (SLP) [3], and prints a copy. Using the location service, Alice knows where is the printer that she has used and locates the post office in the mall. After sending the letter, Alice goes to a zone in the mall, where her phone detects a Web Service that provides different day's promotions in bookstores. By the user preferences, Alice smart phone knows that she is interested in books related to philosophy subjects. Using this information, the mobile device is able to inform Alice about promotions related to this kind of books. The promotion service is provided in three ways: text only, text and images and video. The phone selects the video option considering the Alice preferences, the memory capacity and battery level. Alice consults the promotions while she goes to the melt area. In this area, the promotion service is not available, but Alice doesn't note this because her mobile is able to continue using the service. When the device battery is low, the promotion service is change to text only mode.

3 Context-Aware Middleware: General Characterization

The context-aware middleware platforms are an answer to challenges associated with service discovery, mobility, environmental changes and context retrieval [4]. These challenges are typical in ubiquitous computation. Hence, for facing these challenges it is necessary to provide functionality associated with the properties described below.

1. *Context Management*: this property is associated with context retrieval and processing. The context retrieval means to deal with different kinds of context providers, data representation and communication styles. Furthermore, the context information has to be processed, composed and distributed. In our scenario, the middleware has to retrieve information associated with the mobile resources (available memory, battery level), user preferences (reading interests and kind of favorite services) and the wireless network. Events and policies are the principal mechanisms used to deal with context retrieval and processing.
2. *Adaptation*: it refers to actions that must be triggered when some event occurs in the environment. This means that the middleware platform should react to context

changes. This reaction consists in structure and/or behavior changes. In our scenario, the two kinds of adaptation are needed. Structural adaptation (i.e. to load new functionality) can be required in order to use the available services in the mall (location service, promotion service, printer service). In our scenario, behavioral adaptation takes place when the middleware has to decide how to chose the quality of service according to the memory and battery level. But the adaptation is not only required at runtime. Considering the different kind of entities that is possible to find in pervasive environments, the middleware should be flexible enough to support different configurations and to be deployed on diverse devices. Hence, dynamic and static adaptation are required.

3. *Communication*: this property is necessary to interact with the different entities in the environment. The middleware platform should be able to support different kinds of protocols, interaction paradigms, and communication technologies. The configuration should be not only static but also at runtime when new services can require communication mechanisms not been supported until that moment.
4. *Service Discovery*: it is the capacity for discovering and invoking available services in the environment. This task is done by means of service discovery protocols. These protocols define how the services have to be advertised, discovered and in some cases accessed. Hence, they are built upon communication protocols. It is possible to find different services discovery protocols developed in the academia, the industry and by software vendors. Examples from academia are Ninja Service Discovery Service (SDS) [5] and Intentional Naming System (INS) [6]. Jini [7], UPnP and Rendezvous[8] were created by software vendors. Salutation, SLP and Bluetooth SDP [9] are industrial standards. However, despite the existence of these protocols, some middleware platforms define its own discovery protocols to support the access of new services. In the scenario, the services are advertised using UPnP and SLP.
5. *Persistence*: in some cases it can be useful to storage the context because the user movements and connection variations cause changes in services availability.
6. *Application Building Support*: some approaches provide frameworks that help in the application development and/or adaptation in order to work with the middleware platform.
7. *Paradigms used*: it is important to select the suitable paradigms to develop the platforms in order to satisfy the adaptation issues.

4 Platforms

This section presents some existing context-aware middleware. In the table 1 they have been characterized using the properties presented in section 3.

4.1 Gaia

`Gaia` [10, 11] is a distributed middleware that provides similar functionality to an operating system. It allows the coordination of software entities and heterogeneous networked devices in active spaces (physical spaces). `Gaia` provides services for event

Table 1. Context-Aware Middleware

Middleware Platform	Developer	Paradigm Used	Adaptation		Context Management	Service Discovery	Communication	Persistency		Application B. S
			Static	Dynamic				Data	Context	
Gaia	University of Illinois at Urbana-Campaign	Components	AL	AL	First order logic and boolean algebra	Events	Events and RPC	X	X	Framework
Gaia Microserver	University of Illinois at Urbana-Campaign	Components		AL	Services provided by Gaia	GAIA Proxy	Events and RPC			
CORTEX	Lancaster University	Reflection, components, SOs component frameworks	ML	ML	SOs	SLP, UPnP SOS	Events-SOAP		X	
Aura	Carnegie Mellon University	Components		ML	Situation recognition, proxies	Aura suppliers	Connectors	X	X	
CARISMA	University College London	Reflection		ML	Policies associated with each application	Not supported	According to applications needs			
Middlewhere	University of Illinois at Urbana-Campaign	Components			Algorithm for multisensor location fusion		Events			
MobiPADS	HK Polytechnic University	Components, reflection		AL, ML	Events, mobilets	Mobilets	Events			
SOCAM	National University of Singapore	Ontologies, SOA		AL	Situation recognition, ontologies	SLM service discovery	Events		X	
RCSM	Arizona State University	Components	ML	AL	Situation recognition	R-CDP protocol	Messages		X	
CAPNET	University of Oulu	Components	ML		Wrappers for context sensors	Service discovery component that locates service provides by other components	Events asynchronous messages, RPC, TCP/UDP, HTTP, multicast		X	Framework

management (that means to distribute events in active spaces), context information query (in order applications can adapt to the environment), detection of digital and physical entities, storage of the information associate with entities (space repository), and file management (by means of a context file system that generates a virtual hierarchy associating file to relevant context information). Gaia also provides a framework for building (or to adapt existing) context-aware applications. The Gaia architecture can be seen in figure 1.

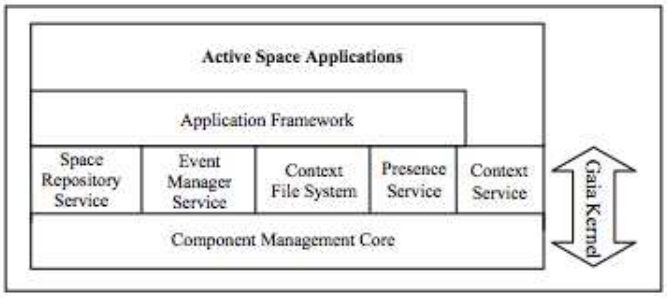

Fig. 1. Gaia Platform Architecture

4.2 Gaia Microserver

Gaia Microserver [12] is a lightweight middleware platform that extends Gaia (see section 4.1) in order to support access to the native capabilities of the mobile devices. The middleware exports the functionality oered by the device to Gaia as components. The Microserver allows interoperate the native C++ code with the java code. Gaia uses the platform to deliver dynamic software components and multimedia contents to users through their mobile devices. The Gaia Microserver components are showed in figure 2. In our scenario, Gaia middleware could be used to detect the different services present in the environment, i.e., the location service, the printer service and the promotion service. Then, those services could be accessed through the mobile device using the Microserver. It will allow device to get the new functionality required in order to use the services.

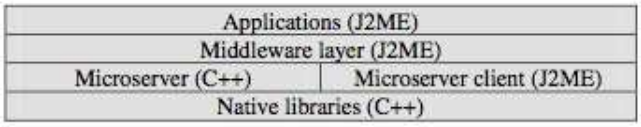

Fig. 2. Gaia Microserver Components

4.3 Aura

Aura [13] is an architectural framework for ubiquitous computing applications. It provides services for managing tasks, applications and context. In Aura, the tasks are abstract representations of a collection of services. When a user moves to another environment, the task representation is migrated and the service providers associated are instantiated in the new location. The services are provided by existing applications. Aura gives supports for registry and access of services. Context observers provide the context information and it is used to derive user intents. In the scenario, the service providers can be considered like the applications that provide the Aura services. The sending of the letter and the searching of book promotions can be considered as Aura tasks. The task concerning the letter includes the location service and the printer service. The task associated with the books comprises only the promotion service. In this last task, the task migration of Aura can be used in order to continue using the promotion service in the melt area. Figure 3 shows the Aura architecture.

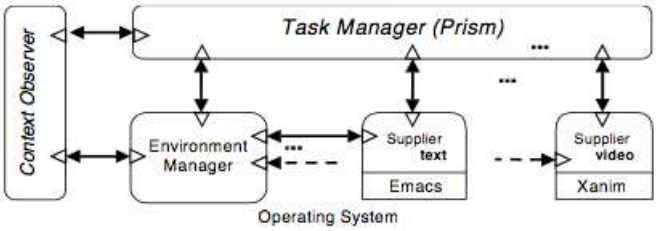

Fig. 3. Aura Framework Architecture

4.4 CORTEX

CORTEX [14, 15] is a context-aware middleware for pervasive and ad hoc environments. The platform is based on concepts of sentient objects (SOs) and component frameworks. SOs are autonomous entities that can get data from the environment and share information between them. They consume and produce events. SOs are able to make decisions and perform actions based on the information sensed. The component frameworks offer services (to the SOs) such as publish-subscribe (used for discovery), group communication, context retrieval, and QoS management. CORTEX can be reconfiguration at runtime using a reflection API. In our scenario, each service discovery protocol (SLP, UPnP and Web Services) could be associated to an specific SO. This SO has the responsibility to enable the device in order to discover and access the services. Another SO could consolidate all services (filtering them using the Alice preferences) and a CORTEX application could use this SO in order to show the available services (by categories), which would allow Alice to send the letter and to knows the book promotions. QoS management service could be used to select the correct promotion service considering the battery level and the available memory. This QoS management could

also be used to monitoring those resources (battery and memory) in order to adapt the QoS. The CORTEX platform is illustrated in figure 4

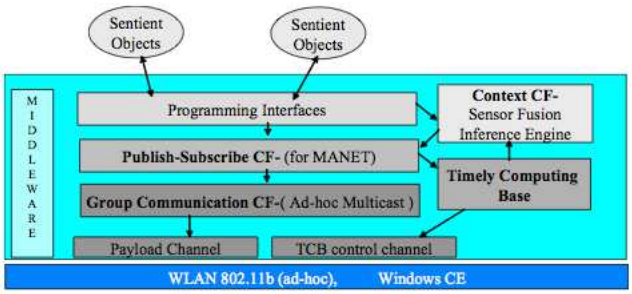

Fig. 4. CORTEX Platform Architecture

4.5 CARISMA

CARISMA [16] (Context-aware Reflective mIddleware System for Mobile Applications) uses the reflection paradigm to enhance the adaptive and context-aware mobile applications development. The idea is to customize the platform considering the applications needs. In order to do that the middleware behavior, with respect to an application, is reified as meta-data in an application profile. This profile contains the description of associations between the service that the middleware customizes, the policies that can be used in the service invocation and the context configuration that allows the use of the policies. In each service invocation, the client application passes its profile to the platform and it determines the policies that can be applied according to the current context. CARISMA provides a reflective API that allows, at runtime, to modify the associations describes in the profile. The conflicts that may arise between profiles are resolved using a micro-economic approach. In this approach, the system is modeled as an economy where the consumers (applications) reach an agreement about a limited set of goods (the policies) using the middleware platform like auctioneer. In the scenario, it would be possible to have a CARISMA application that accesses the promotion service. In the profile associated with this application, a policy could express the conditions (battery level and available memory) that allow the device to invoke the different kinds of the promotion service.

4.6 MobiPADS

MobiPADS [17] (Mobile Platform for Actively Deployable Service) is a system for mobile environments. In MobiPADS the services are called mobilets. They can migrate between MobiPADS environments. Each mobilet is divided in slave and master. The slave resides in the server side, and the master in the client side. The mobilets are configured as chained objects to provide augmented services and protocols to the

mobile applications. MobiPADS achieves context-awareness with the utilization of an event notification model. It monitors the status of the interested context and notifies the changes to the subscribed entities. The event notification model allows the composition of primitive events into an event graph. In this way, when an event service is built and subscribed, it has to monitor and analyze the basic events and match them according to the event graph structure. The adaptation can be made in the platform, based on system profiles, or in the mobilets, allowing them to change according to the events that they receive. The figure 5 shows the platform architecture. Using the MobiPADS platform in the scenario, the location service, printer service and promotion service could be encapsulated in mobilet objects. Of course, the slaves would reside in the MobiPADS side and the masters in the user's mobile device. The migration property of the mobilets would be used to make available the promotion service in the meal area. The event model of MobiPADS could be used to monitoring the battery level and memory capacity in order to detect when it is necessary to change the promotion service to the text only mode.

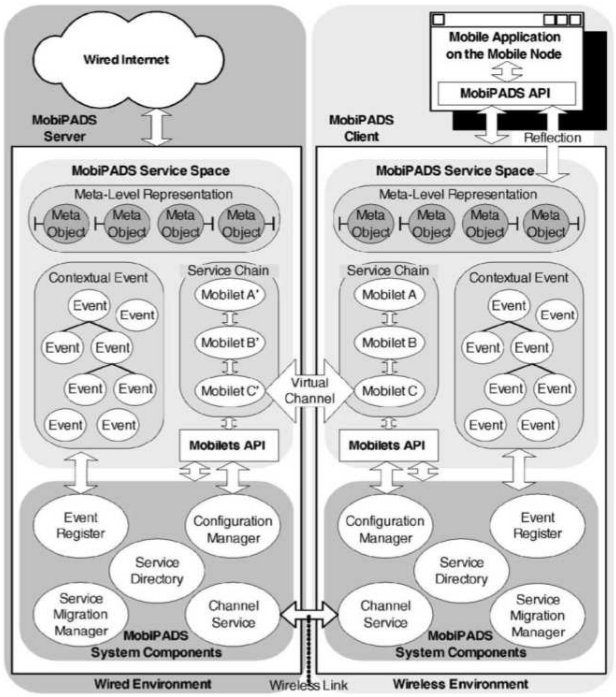

Fig. 5. MobiPADS Architecture

4.7 MiddleWhere

`MiddleWhere` [18] is a distributed middleware architecture for location. The platform separates applications from location detection technologies. It allows the addition of different kinds of sensing technologies and determines the location information quality. The applications can make queries about the objects location or they can subscribe to be notified when a location condition becomes true. The location model applied in `MiddleWhere` is hierarchical and it uses three kinds of location: points, lines, and polygons. The location can be expressed in coordinates and a symbolic way. `MiddleWhere` also determines the quality of the location information according to its freshness (as time goes on, the quality of location data reduces). The platform also provides persistence for the location information. `MiddleWhere` has a reasoning engine that deduces spatial relationships between mobile objects and the physical environment. The architecture of the platform is showed in figure 6. `MiddleWhere` could be useful in the scenario in order to provide the location service that Alice use to know where is she and where are the printer and post services.

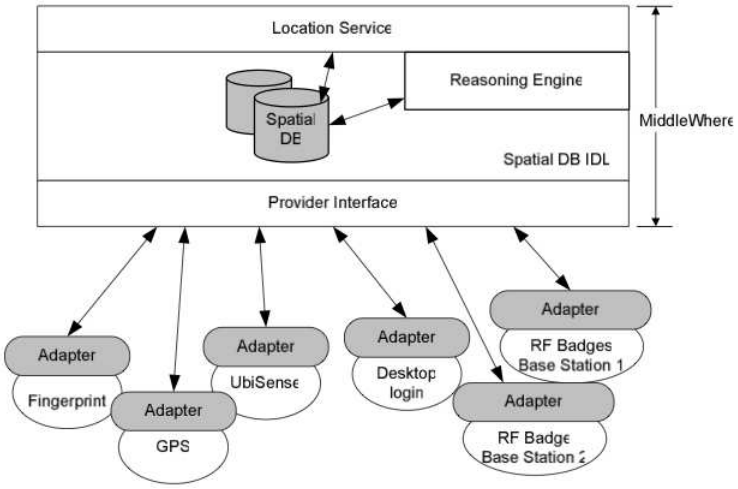

Fig. 6. MiddleWhere Architecture

4.8 SOCAM

Service-oriented Context-Aware Middleware (`SOCAM`) [19] is a platform to build context-aware mobile services. `SOCAM` uses ontologies to model the context. The platform can supports semantic representation, context reasoning and context-knowledge sharing. `SOCAM` is composed by Context Providers, Context Interpreters, Context Database, Location Service, and Context-aware Mobile Services. The Context Providers give context information and represent it as context events in the form of OWL descriptions. The

Context Interpreters provide high-level context information. Hence, they are also Context Providers. The Context Interpreters include Context Reasoners (containing rules that triggers actions associated with context changes) and Context Databases (that contains instances of the current ontology). The Location Service allows the platform to locate context providers.. The Mobile Services are applications and services that use the context information and adapt their behavior according to this information. They can obtain the context by querying the Context Providers or by listening specific events sent by the Context Providers. Figure 7 shows the SOCAM architecture. Applying SOCAM to the scenario, service providers would be represented by context providers in the Context Middleware Layer that allows the detection and invocations of services. The memory, battery and user preferences would be associated to Context Providers and Context Interpreters in order to adapt the services that depend on them. In Context Application Layer each service (promotion, service and location) would also have a context-aware service representation. It would be useful in the promotion service case to adapt the mode according to the battery and memory changes.

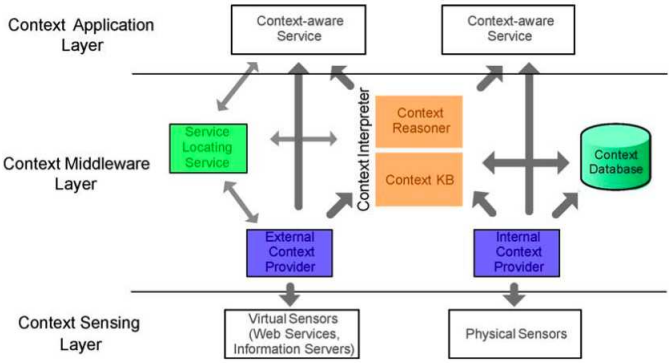

Fig. 7. SOCAM Architecture

4.9 CAPNET

CAPNET [20] is a context-aware middleware for mobile multimedia applications. The platform is composed by components. Each component offers a special service to other components and applications. The core-components of CAPNET are: Component Manager (that control the components and their stubs), Connectivity Manager (that controls and monitors the connection of the mobile devices), Messaging (that creates channels and supports asynchronous communication, remote procedure calls and channel-related operations) and Service Discovery (that locates services and available components). The Service Discovery component is based on Jini technology. CAPNET also has other components that are necessary to support context-aware applications. These components are: Context-Based Storage, Context, User Interface and Media. The Context-Based Storage component stores and retrieves context data by request. The Context

component provides context information acting as a wrapper for context sensors. The User Interface component supports the design and implementation of UIs application. In order to that, the UI allows three different techniques: abstract UIs (XML), plug-in UIs (downloadable Java code) and Web-based UIs (HTML). The media components provide functionality to capture images, audio and video, facilitating the portability and scalability of native media capabilities across the various devices. The CAPNET platform is illustrated in figure 8. In the scenario, CAPNET could be used to discover the services but it will required to have some kind of adapter that allow to use the services as Jini services. The multimedia CAPNET capabilities could be used to provide the best video quality according to the resources in Alice's smart phone.

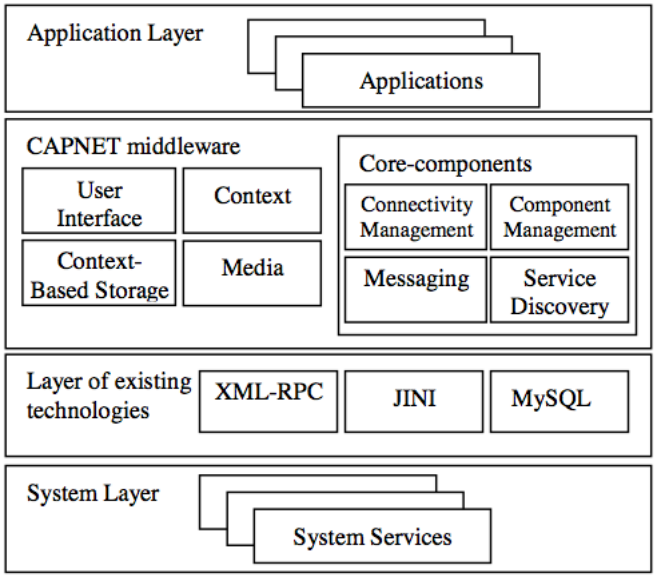

Fig. 8. CAPNET Architecture

4.10 RCSM

Reconfigurable Context-Sensitive Middleware [21, 22] (RCSM) has been developed to provide development and runtime support for situation-aware (SA) software in ubiquitous computing. It provides a SA-IDL that allows the specification of context-situation relevant for an application, the actions to be triggered, and when these actions has to be executed. The SA-IDL specifications are compiled to generate application skeletons. These skeletons interact at runtime with the RCSM Object Request Broker (R-ORB) and the SA processor. R-ORB deals with the context discovery, collection and propagation. To do that, R-ORB has a context manager that uses a context discovery protocol to register local sensors and to discover remote sensors. When an application starts up, the

discovery protocol is used to find the sensors (local or remote) that satisfy the context requirements. The SA processors manage the SA requirement specifications, stores the context, recognize the situation based in the context and determine when the actions have to be triggered.

5 Conclusions

In this article we have presented several context-aware middleware platforms. They are characterized in terms of some relevant properties for this kind of application. Furthermore, the applicability of context-aware middleware is illustrated with a simple scenario.

All the context-aware middleware platforms reviewed allow devices to recover and process context information. The platforms also offer different mechanism to detect changes in the environment and trigger adaptations (in the application and/or middleware level) associated with these changes. However each one focuses in a specific aspect of pervasive environments. For example, `Aura` deals with user tasks and the migration and adaptation of them according to the current user context. `Gaia` offers access to heterogenous resources available in active spaces. `MiddleWhere` is focused in location-awarennes. `MobiPADS` offers service migration and adaptation of the service composition represented by the mobilets. Then, the selection of the more suitable platform will depend on the requirements of applications that are being built.

Although all middleware platforms could be used in the scenario proposed, it is not always straightforward. In some cases, it is necessary to create some kind of representation for the service providers in order to use all the capabilities of the platforms. Besides, some platforms do not consider the dynamic load of functionality to use services that are discovered at runtime, limiting their use in pervasive environments.

Finally, in spite of the existence of different context-aware middleware platforms, there is not a definitive solution for all the challenges associated with pervasive computing. In fact, there are some projects that continue the research in this area, such as the Cappucino Project[1] (from Inria Laboratory) and the MUSIC Project[2].

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, London, UK, Springer-Verlag (1999) 304–307
2. Corporation, U.I.: Upnp technologythe simple, seamless home network. `http://www.upnp.org/resources/whitepapers.asp` (1999)
3. Guttman, E., Perkins, C., Veizades, J., Day, M.: Service location protocol, version 2. `http://www.salutation.org` (1999)
4. Romero, D., Parra, C., Seinturier, L., Duchien, L., Casallas, R.: An sca-based middleware platform for mobile devices. In: 2008 Middleware for Web Services (MWS 2008) Workshop at EDOC2008, Munich, Germany (sep 2008)

[1] Cappucino Project: `http://cappucino.oqube.com/`

[2] MUSIC Project: `http://www.ist-music.eu/`

5. Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., Lilley, J.: The design and implementation of an intentional naming system. In: SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles. Volume 33., New York, NY, USA, ACM Press (December 1999) 186–201
6. Hodes, T.D., Czerwinski, S.E., Zhao, B.Y., Joseph, A.D., Katz, R.H.: An architecture for secure wide-area service discovery. Wirel. Netw. **8**(2/3) (2002) 213–230
7. Sun Microsystems, I.: Jini specifications and api archive. `http://java.sun.com/products/jini/2_0_2index.html` (2005)
8. Cheshire, S., Krochmal, M.: Dns-based service discovery. `http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt` (2008)
9. SIG, B.: Specification of the bluetooth system. `http://www.bluetooth.com/NR/rdonlyres/F8E8276A-3898-4EC6-B7DA-E5535258B056/6545/Core_V21__EDR.zip` (2007)
10. Román, M., Hess, C.K., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: Gaia: A Middleware Infrastructure to Enable Active Spaces. IEEE Pervasive Computing (Oct–Dec 2002) 74–83
11. Henricksen, K., Indulska, J., Mcfadden, T.: Middleware for distributed context-aware systems. In: International Symposium on Distributed Objects and Applications (DOA, Springer (2005) 846–863
12. Chan, E., Bresler, J., Al-Muhtadi, J., Campbell, R.: Gaia microserver: An extendable mobile middleware platform. In: PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, Washington, DC, USA, IEEE Computer Society (2005) 309–313
13. Jo a.P.S., Garlan, D.: Aura: an architectural framework for user mobility in ubiquitous computing environments. In: WICSA 3: Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture, Deventer, The Netherlands, The Netherlands, Kluwer, B.V. (2002) 29–43
14. Sorensen, C.F., Wu, M., Sivaharan, T., Blair, G.S., Okanda, P., Friday, A., Duran-Limon, H.: A context-aware middleware for applications in mobile ad hoc environments. In: MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, New York, NY, USA, ACM (2004) 107–110
15. Blair, G.S., Coulson, G., Grace, P.: Research directions in reflective middleware: the lancaster experience. In: ARM '04: Proceedings of the 3rd workshop on Adaptive and reflective middleware, New York, NY, USA, ACM (2004) 262–267
16. Capra, L., Emmerich, W., Mascolo, C.: Carisma: Context-aware reflective middleware system for mobile applications (2003)
17. Chan, A., Chuang., S.N.: Mobipads: a reflective middleware for context-aware mobile computing. IEEE Transactions on Software Engineering **29**(12) (2003) 1072–85
18. Ranganathan, A., Al-Muhtadi, J., Chetan, S., Campbell, R., Mickunas, M.D.: Middlewhere: a middleware for location awareness in ubiquitous computing applications. In: Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, New York, NY, USA, Springer-Verlag New York, Inc. (2004) 397–416
19. Gu, T., Pung, H., Zhang, D.: A middleware for building context-aware mobile services (2004)
20. Davidyuk, O., Riekki, J., Rautio, V.M., Sun, J.: Context-aware middleware for mobile multimedia applications. In: MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia, New York, NY, USA, ACM (2004) 213–220
21. Yau, S.S., Huang, D., Gong, H., Seth, S.: Development and runtime support for situation-aware application software in ubiquitous computing environments. In: COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), Washington, DC, USA, IEEE Computer Society (2004) 452–457

22. Yau, S.S., Karim, F., Wang, Y., Wang, B., Gupta, S.K.S.: Reconfigurable context-sensitive middleware for pervasive computing. IEEE Pervasive Computing **1**(3) (2002) 33–40