American Journal of Computer Architecture 2012, 1(3): 51-56
DOI: 10.5923/j.ajca.20120103.02

An Extensive Survey of Context-Aware Middleware Architectures

Aamna Saeed[*], **Tabinda Waheed**

Department of Computer Science, Military College of Signals, National University of Sciences & Technology, Rawalpindi, Pakistan

Abstract Context-awareness is a vital requirement in building valuable and capable adaptive systems. Context-aware ubiquitous computing focuses on the use of context of users, devices, environment, etc in order to offer services essential for a particular person, space and time. This paper provides a survey of context-aware middleware architectures. An overview of each middleware is provided, along with the description of the main features. Based on the conducted survey, this paper compares and contrasts the various characteristics of context-aware middleware architectures. We present the analysis of the middleware architectures based on several parameters including fault tolerance, adaptability, interoperability, architectural style, discoverability, location transparency and aspect oriented composition.

Keywords Context-awareness, Fault Tolerance, Location Transparency, Ubiquitous Computing

1. Introduction

Nowadays, personal digital assistants (PDAs) and mobile phones offer a multipurpose set of services that leads to emergence of various new applications. Mobility and context-awareness introduces further challenges. The applications need to adapt themselves to an altering environment. The problems could be solved by using an intermediate software layer that could perform the task related to mobility and context-awareness. Thus, it helps in avoiding the rising difficulty of the applications and lets the developers focus on application-specific tasks[1]. Mobile systems run in an extremely dynamic environment. Due to user mobility, execution context keeps on changing frequently. In order to understand context-aware middleware, first we are required to understand different types of contexts. Context can be external to computer systems as location and proximity or it can be internal context, such as available disk space[2].

Usually current location of a mobile unit determines its context which, then, specifies the environment where the computation related to the mobile unit is performed. The context also includes device characteristics, user's actions, services, and other resources of the system[3].

Context-aware systems consist of various distributed components such as sensors, actuators, context information stores, context information processors, etc. Today, it is widely accepted that, for reducing the complexity of context-aware applications and encouraging their reuse, additional infrastructural components are desirable. The aim of the middleware in traditional distributed systems was to hide heterogeneity and distribution by providing ways of treating remote resources as if they were local. For static environment, this may be useful, but in dynamic wireless environments it is not beneficial. Since applications often need to base decisions on information about distribution and the environment, middleware systems for Pervasive Computing focus on providing suitable abstractions for dealing with heterogeneity and distribution without hiding them and in some situations may provide information about distribution and heterogeneity as context information[4, 2].

Numerous approaches have been presented for building context-aware middleware architectures. In our research paper we have presented various context-aware middleware architectures and their vital characteristics. A detailed analysis is presented that compares and contrasts various features of context-aware middleware such as interoperabili ty, adaptability, location transparency, discoverability, architectural style, aspect oriented decomposition and fault tolerance.

2. Related Work

Nowadays extensive research is being carried out on context-aware systems. Many approaches for addressing the issue of context-awareness in middleware architectures have been presented. Thus new context-aware architecture keeps on emerging.

Gaddah and Kunz[3] have provided a general overview of the most relevant mobile middleware systems .They highlighted not only modern solutions but also objectives

* Corresponding author:
aamna.saeed87@gmail.com (Aamna Saeed)
Published online at http://journal.sapub.org/ajca

that still need consideration. The main purpose of the survey is to help out middleware researchers assess the strength and weakness of different middleware architectures. The authors have not given comparison of discussed middleware architectures.

Kjær[2] presented a survey of a chosen set of context-aware middleware architectures, and classified their characteristics and use according to their proposed taxonomy. 978-1-4244-6875-1/10/$26.00 ©2010 IEEE

Sadjadi[5] presented three orthogonal methods to categorize adaptive middleware. The author proposed three-dimensional taxonomy of adaptive middleware. He has not given comparison of various features of context-aware middleware architectures.

Mustafiz and Kienzle[6] conducted a survey of specialized software development methods, frameworks, middleware, software architectures, and other approaches that assist developers in producing dependable software. The paper gives a comparison of discussed methods, frameworks and middleware architectures based on safety, security, availability, maintainability and QoS. In this paper the authors mainly focuses on dependability issues and have not included other important features like interoperability, adaptability, context-awareness, location transparency etc.

Baldauf, et al.[7] presented a survey in which they have illustrated various design principles and context models for context-aware systems .They have presented different existing middleware and server-based approaches to ease the development of context-aware applications. The authors have provided a good analysis of various context-aware systems but the comparison is performed on a very limited set of context- aware systems. Other well known context-aware middleware architectures like Aura, CARMEN, CARISMA and various others have not been incorporated in the survey.

Our survey has been extensively conducted and included all the well known context-aware middleware architectures. The analysis compares recently proposed middleware architectures with the existing middleware architectures. The analysis is based on the vital characteristics of context-aware middleware including fault tolerance, interoperability, adaptability, architectural style, discoverability, location transparency and aspect oriented decomposition.

3. Overview Of Context-Aware Middleware Architectures

This section provides an overview of each context-aware middleware architecture .By examining existing context-aware systems; we have identified some important features for comparison which are: are architectural style, location transparency, Aspect-oriented decomposition, service discovery, fault tolerance, adaptability and interoperability.

Architectural style of ant middleware architecture is of primary importance. It defines the way different components are arranged in middleware. Extensibility and flexibility of any middleware are greatly dependent on architectural style. It also influences the adaptivity mechanism in middleware.

One of the most important functionality required in context- aware environment is adaptability. Adaptability depicts the ability to change the behavior according to varying environment. It can be static (occurs at start-up or compile time) or dynamic (takes place at run-time)[5].

Service Discovery is a vital requirement in ubiquitous computing environment. It determines how applications discover other entities and how they can be discovered by other entities[8].

Fault tolerance determines the reliability and safety features of any middleware architecture. Fault-tolerant middleware enables applications to continue operating in the presence of faults[6].

Another main purpose of middleware is to provide interoperability. It makes two various systems to exchange information and to utilize that exchanged information. In context-aware environments various mobile devices needs to communicate but some context-awaremid dleware architectures do not offers this facility[8].

In aspect oriented decomposition cross cutting concerns are separated into modules known as aspects. Aspect-orient ed decomposition allows separation of cross-cutting concerns at development time, compile time or runtime Aspects encapsulate non-functional behavior. In real-time applications, where safety and security concerns are vital, it is required to adopt such a context-aware middleware architecture which supports aspect-oriented decomposition [5].

Location transparency is a significant feature which overcomes the requirement for client objects to exactly identify the location of a server object when interacting and requesting services offered by the server object[8].

3.1. Aura

Aura is suitable architecture for ubiquitous computing. It is based on the idea of personal Aura and acts as a proxy for the user it represents. When the user environment changes, its Aura provides support to user tasks by adapting to local resources. Aura provides services for management of tasks, applications, and context .It supports a mobile users moving across different environments, by moving the representation of the task. In Aura architecture Context Observer is provided for context management .For mobile users Service providers are created at the user's new location for the task. The context observer collects context information, and reports changes to the Task- and Environment Managers[9, 10].

3.2. Capnet

CAPNET is a context-aware middleware architecture used specially for mobile multimedia applications. CAPNET is capable of service discovery, user interface building, manag ement of the local and network resources, asynchronous messaging, context information management and storage. It provides support for wide range of context information which includes location, time, and user's preferences. The middleware has also the ability to switch traffic from one network connection to another. It has the ability to locate the services and software components. The middleware provides services for creating multimedia messages when predefined context is identified. It supports development of complex context-aware multimedia applications for mobile devices [1].

3.3. Carisma

CARISMA handles the adaption of middleware depending on the requirements of the applications. In CARISMA profiles exist as meta-data of the middleware for each application. The profiles comprises of passive and active parts. In the passive parts, actions the middleware should take in response to specific context events are described. The active information specifies relations between services used by the application and the rules that should be applied to deliver those services. Different environmental conditions can be specified to determine how a service should be provided to the requested application. Reflection can be used by the application at any time to modify the profile kept by the middleware[2].

3.4. Carmen

CARMEN is middleware that aims for context-aware resource management .It has the capability of supporting the automatic reconfiguration of wireless Internet services in accordance with the context alterations. CARMEN has the ability to manage resources in wireless environment in case of temporary disconnects. Proxies are used in CARMEN which acts as the mobile agents existing in the same CARMEN environment as the user. For each mobile user there exists a proxy which provides access to resources required by the user. When migrating across different environments, the proxy is responsible to make sure that resources are also accessible in the new environment[11, 2].

3.5. Cooltown

The Cooltown middleware architecture is proposed for supporting communication between wireless, mobile devices and a web-enabled environment. The main principle behind Cooltown is that devices, people, and things have a web-presence recognized by a URL. This URL provides a good interface to the entity. Users utilize PDAs to interact with the provided web-services in a web-enabled environment. Cooltown requires wireless Internet access for users to communicate with the system. In local device to device communication URLs are passed among devices[2].

3.6. Cortex

CORTEX is a context-aware middleware architecture that provides support for both pervasive and ad hoc environments. The middleware architecture consists of a number of component frameworks (CF).The CORTEX architecture utilizes reflection and component technology. In CORTEX efficient mechanisms are provided for context-awareness and intelligent decision-making. It is a flexible framework that enables the use of a number of various service discovery protocols[12].

3.7. Gaia

Gaia is a middleware architecture which has the ability to manage resources enclosed in physical spaces. In Gaia heterogeneity of active spaces are hidden, and they are presented as a programmable environment, rather than a set of individual and disconnected diverse devices. Mainly the focus of Gaia is on the interaction among users and active spaces .Gaia has the important characteristic of providing functionality to customize applications in different ways. "User data and applications can be mapped dynamically to the resources provided in the current environment." Users can move across various active spaces[13].

3.8. MiddleWhere

MiddleWhere is a context-aware middleware architecture which makes it possible to combine various location detection technologies. It aims at providing location information to the applications obtained from different location sensing technologies. It offers the functionality of incorporating extra location technologies dynamically as they become available. Location Providers provides the Location information which is stored in a spatial database. Location is determined by the reasoning engine which uses the location information derived from different location providers .Location is provided by a location service which makes use of the spatial database and the reasoning engine[14, 2].

3.9. FlexiNet

The FlexiNet is Java middleware architecture. It focuses on various problems of configurable middleware. Proxies represent the Interface on remote objects. Binders are provided for Proxies to make remote access. Each binder has the ability to create a generic binding among a local proxy object and the remote object it represents. FlexiNet supports the concept of multiple name spaces for interfaces to provide flexibility. The modularity makes it easier for management policies to be plugged in[3, 15].

3.10. Nexus

The NEXUS middleware architecture is proposed for all types of location-aware applications. It comprises of four layers that work together: the user interface, the sensor systems, the communication and the data management. The user interface runs on the mobile device carried by users .The user interface enables the interaction between location-aware applications and NEXUS platform. The sensor systems are required to provide positioning information to the NEXUS system. The communication unit handles data transfer among the various components of NEXUS .The data management organizes the data in a distributed environment and supports sharing of processing between different servers. Clearly defined interfaces among each of the three layers guarantee least dependency among the layers[16, 3].

3.11. One.world

One.world middleware architecture supports development of pervasive applications. The primary focus of One.world is on those applications which automatically adapt to extremely dynamic computing environments. One.world architecture is designed for satisfying all core requirements. Additionally it provides mechanisms for application migration, data storage and fault tolerance. One.world requires Java Virtual Machine in order to provide a uniform execution environment across diverse devices. It is essential that the mobile terminal should also support Java[8].

3.12. AspectIX

AspectIX is a context-aware middleware that supports aspect-oriented decomposition. It is based on the distributed object model. The principal behind AspectIX is to separate non-functional cross-cutting concerns from functional logic. J. Aspects encapsulate cross-cutting concerns (i.e., non-functional behaviour like safety, security etc). AspectIX supports the concept of dynamic weaving of aspects i.e. Aspects can be added or removed at run-time. Thus its architecture is more flexible to changes[17, 5].

3.13. MobiPADS

MobiPADS architecture is especially designed for providing support to context-aware processing. It provides an execution platform. In response to environments of changing contexts, the platform allows active service deployment and reconfiguration of the service composition. The main entity in MobiPADS is Mobilets, which are the service providers. Mobilets move across different MobiPADS environments. Each mobilet comprises of a slave and a master. The slave resides on a server, whereas the master exists on a mobile device. In order to provide a particular service each slave and master coordinates with each other. MobiPADS needs the internal context of mobile devices to adapt to variations in the computational environment[2, 18].

3.14. Homeros

The HOMEROS architecture was designed to offer maximum flexibility, supporting service providers and user needs. Thus the extensibility of HOMEROS into distributed and hybrid architecture for good quality services is simpler. The services include user preference management, expert system, and multimedia processing. HOMEROS architecture aims to provide flexible user interface infrastructure. For efficient management of huge resources, context, location, and various services, HOMEROS adopts a hybrid-network model. It provides high flexibility to the applications by using dynamically configurable reflective ORB. It comprises of three layers, which are core component management layer, extended component service layer, and system support layer[10].

3.15. Socam

SOCAM is a middleware which provides support for most of the tasks concerned with context. The major tasks included are obtaining context from various sources; interpreting context; and sharing of context. The major characteristic of the SOCAM architecture is the provision of support for context reasoning. In SOCAM each component is designed as an independent service component. A Service Locating service provides the facility to locate and access the components[19].

4. Analysis

In Table I, we have summarized the main features of the discussed middleware architectures. Some middleware architectures adopt layered approach where different services are localized in layers with dependency among the layers. Such architectures support extensibility and flexibility to some extent. Other architectures are those in which modules or components represent the major building blocks. This kind of middleware is more modifiable and flexible as compared to layered architecture as in layered approach dependency exists among layers whereas modules or components are independent. Also modular architecture supports reusability like component-based design. NEXUS has service-oriented architecture which is useful for web-services. Location transparency is provided in CARMEN, CORTEX, FlexiNet, MobiPADS, Gaia, CAPNET, NEXUS, One.world and AspectIX whereas the rest of the middleware architectures in table I do not support location transparency. The only middleware which provides infrastructure supporting aspect oriented decomposition is AspectIX. Aspect oriented middleware focuses on aspects, which are the modules capturing cross cutting functionalities. AspectIX is open to aspects that are to be added at run-time. In applications where cross-cutting concerns (non-functional requirements like logging, security, safety) need to be woven into applications at compile or run time aspect-oriented decomposition is required. Fault tolerance is the feature that directly affects the reliability of the middleware architecture. As shown in the table I Gaia, Flexinet and One.world support fault tolerance and are considered to be the most reliable middleware architectures. In safety-critical systems where recovery from failure is crucial, middleware architectures supporting fault tolerance mechanism are useful. More reliable middleware architectures are appropriate in military command and control and medical applications. Adaptability is the capability of middleware to adapt to the varying environment. Most of context-aware middleware architectures support adaptability except Cooltown, middlewhere and SOCAM. Middlewhere and SOCAM both maintain context information and provides to the applications or mobile agents. In Cooltown context information is maintained by middleware and application using the context information adapts them. In real-time applications like air traffic control system or automatic car control system, which are time critical and need adaptation mechanism to direct the system to safe state in unpredicted variations, middleware architectures supporting adaptation are beneficial. Some middleware architectures provide dynamic adaptability (adapting at run time) as CARMEN, CAPNET, Flexinet, HOMEROS, MobiPADS and AspectIX. Aura, CARMEN, MiddleWhere and SOCAM do not aim for interoperability. Aura is a task oriented middleware and acts as a proxy when user changes his location. CARMEN provides proxies and when user moves form one environment to other, proxies provides access to resources needed by the mobile user. Gaia uses concepts of operating system and provides resource management and supports multi-device, context-sensitive, and mobile applications. Middlewhere manages and provides location information to applications. SOCAM middleware has the ability to meet the needs of context-aware systems regarding limited memory and CPU resources. MiddleWhere is developed as an extended Gaia service. It therefore does not hold the responsibility of service discovery. It is integrated with Gaia which performs the functionality of discovering appropriate service. Service discovery has not yet been included in CARISMA architecture and research is being conducted on this issue nowadays. All other middleware architectures support service discovery, as shown in table I.

Table 1. Evaluation of context-aware middleware architectures

No	Middleware	Architectural Style	Location Transparency	Aspect Oriented Decomposition
1	Aura	Modular(Task Manager, Environment Manager, Context Observer)	No	No
2	CARMEN	Layered(Metadata Manager, Context Manager, Event Manager, Discovery,Directory,Monitoring)	Yes	No
3	CARISMA	NA	No	No
4	Cooltown	Modular(Web Presence Manager, Description, Directory, Discovery Modules, Autobiographer, Observer and Control)	No	No
5	CORTEX	Modular(Publish-Subscribe, Group Communication, Context and QoS Management)	Yes	No
6	Gaia	Distributed Object System(Gaia kernel, Gaia Application Framework and Applications including Space Repository Service, Event Manager Service, Context File System Context Service, Presence Service)	Yes	No
7	MiddleWhere	Layered(Provider Interface, Location Service, Reasoning Engine)	No	No
8	CAPNET	Modular(Connectivity Management, Component Management, Service Discovery, Messaging)	Yes	No
9	Flexinet	Layered(Serial Layer, Name Layer,Rex Layer, Session Layer, UDP Layer)	Yes	No
10	NEXUS	Layered(Discovery Layer, Agent Layer, Service Layer)	Yes	No
11	One.world	NA	Yes	No
12	ASPECTIX	Fragmented and Distributed	Yes	Yes
13	MobiPADS	Modular(Configuration Manager, Service Migration Manager, Service Directory, Event Register, Channel Service)	Yes	No
14	HOMEROS	Layered(Core Component Management Layer, Extended Component Service Layer, System Support Layer)	No	No
15	SOCAM	Distributed with Centralized Server, Context Providers, Context Interpreters, Context database, Service Location Service)	No	No

No	Middleware	Fault Tolerance	Interoperability	Service Discovery	Adaptability
1	Aura	No	No	Yes	Yes
2	CARMEN	No	No	Yes	Yes(Dynamic)
3	CARISMA	No	Yes	No	Yes
4	Cooltown	No	Yes	Yes	No
5	CORTEX	No	Yes	Yes(Service Discovery CF)	Yes
6	Gaia	Yes	Yes	Yes(Context Service Module)	Yes
7	MiddleWhere	No	No	No	No
8	CAPNET	No	Yes	Yes	Yes(Dynamic)
9	Flexinet	Yes	Yes	Yes(Dynamic Discovery)	Yes(Dynamic)
10	NEXUS	No	Yes	Yes(Discovery Layer)	Yes
11	One.world	Yes	Yes	Yes(Service Migration Manager)	Yes
12	ASPECTIX	No	Yes	Yes(Extension of CORBA)	Yes(Dynamic)
13	MobiPADS	No	Yes	Yes	Yes(Dynamic)
14	HOMEROS	No	Yes	Yes(Component Repository responsible for Discoverability)	Yes(Dynamic)
15	SOCAM	No	No	Yes(Context Reasoning Engine)	No

When it is desired to efficiently support computational requirements of mobile users, it is crucial to maximize the utilization of resources provided. HOMEROS is the middle ware architecture which capably configures and monitors the environment to manage the heterogeneity of computing environments and variability of resources[18, 19].

5. Conclusions and Future Work

In this paper we have analysed and compared different context-aware middleware architectures. The comparison and analysis gave an insight into the strengths and weaknesses of the middleware architectures. This survey is the most extensively conducted survey on context-aware middleware architectures and provides an in-depth analysis; comparing the most important characteristics of context-aware middleware architectures. We present the analysis of the middleware architectures based on fault tolerance, adaptability, interoperability, architectural style, discoverability, location transparency and aspect oriented composition.

The comparison provides an excellent base for the efficient and appropriate use of the described middleware architectures according to their main features in various context-aware environments.

It is of interest to categorize these context-aware middleware systems into taxonomy of context-aware middleware architectures. Although existing classification [2] provides a well organized, but it has not included some context-aware middleware architectures which have been included and analysed in our paper. Based on the comparison performed in this paper, future research can focus on addressing limitations of existing context-aware middleware architectures and propose new middleware architectures.

REFERENCES

[1] Davidyuk, O., Riekki, J., Rautio, V-M. & Sun, J, "Context-Aware Middleware for Mobile Multimedia Applications," In: Proceedings of the 3th International Conference on Mobile and Ubiquitous Multimedia (MUM2004), October 27-29.2004, College Park, Maryland, USA, pp. 213-220

[2] Kjær, K.E, "A Survey of Context-Aware Middleware," In Proc. Software Engineering 2007, ACTA Press, pp.148-155, 2007.

[3] Gaddah, A. and Kunz, T, "A survey of middleware paradigms for mobile computing," Technical Report SCE-03-16, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, July 2003.

[4] K. Henricksen, J. Indulska, T. McFadden, and S. Balasubramaniam, "Middleware for distributedcontext-awar e systems," In International Symposium on Distributed Objects and Applications (DOA), volume 3760 of Lecture Notes in Computer Science, pp 846–863. Springer, 2005.

[5] S. M. Sadjadi and P. K. McKinley, "A survey of adaptive middleware," Technical Report MSU-CSE-03-35, Department of Computer Science, Michigan State University, East Lansing, Michigan, December 2003.

[6] Mustafiz, S., Kienzle, J, "A survey of software development approaches addressing dependability," In: Guelfi, N., Reggio, G., Romanovsky, A. (eds.) FIDJI 2004. LNCS, vol. 3409, pp. 78–90. Springer, Heidelberg (2005)

[7] M. Baldauf ,S. Dustdar and F.Rosenburg "A survey on context-aware systems," Int. J. Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, 2007

[8] T Salminen, J Riekki, "Lightweight Middleware Architecture for Mobile Phones, "Proc. 2005 International Conference on Pervasive Systems and Computing, 2005.

[9] David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste, "Project Aura: Towards Distraction-Free Pervasive Computing," IEEE Pervasive Computing, April-June 2002.

[10] Han, S., Bong, Y. & Youn, H,"A New Middleware Architecture for Ubiquitous Computing Environment," In: Proceedings of the Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (WSTFEUS'04), May 11-12.2004,Vienna, Austria, pp. 117-121

[11] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "Context- aware middleware for resource management in the wireless Internet," IEEE TSE, 29 (12): 1086–1099, 2003.

[12] Sørensen, 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", Proc' of the 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC'2004) at Middleware 2004, Toronto, Canada, October 2004.

[13] Román, M., C.K. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," IEEE Pervasive Computing 2002,1(4): pp. 74-83.

[14] A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. H. Campbell, and M. D. Mickunas. Middlewhere, "A middleware for location awareness in ubiquitous computing applications," In H.-A. Jacobsen, editor, Middleware, volume 3231 of Lecture Notes in Computer Science, pages 397–416, Springer, 2004.

[15] Hayton R, Bursell MH, Donaldson D, Herbert A,"Mobile Java objects,"In International Conference on Distributed Systems Platforms and Open Distributed Processing (MIDDLEWARE), The Lake District, UK, 1998.

[16] Kaveh N. and Ghanea-Hercock R.. "NEXUS – Resilient Intelligent Middleware", BT Technology Journal, vol. 22, no. 3, pp. 209-215, July 2004

[17] Hauck F., Becker U., Geier M., "AspectIX: A Middleware for Aspect- Oriented Programming." Workshop on Aspect-Orie nted Programming. ECOOP'98, Bruselas (Bélgica), 1998.

[18] A. T. Chan and S.-N. Chuang." MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing," IEEE Trans. on Software Engineering, 29(12):1072–1085, 2003

[19] T. Gu, H. K. Pung, and D. Q. Zhang," A middleware for building context-aware mobile services," In Proceedings of IEEE Vehicular Technology Conference, May 2004.