IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 11, November 2015

A Survey of Context-Aware Framework for Pervasive Environment

K. H. Walse[1], Dr. R.V. Dharaskar[2], Dr. V. M. Thakare[3]

Asst. Professor, Dept. of CSE, Anuradha Engineering College, Chikhli, India [1]

Former Director, DMAT- Disha Technical Campus, Raipur, CG, India [2]

Professor& Head, P.G. Dept. of Computer Sci., S.G.B. Amravati University, Amravati, India [3]

Abstract: The context aware framework is the backbone of context aware application the framework. It is because the context has continuously been in the process of evolution. It is implemented depending on the requirements of situation. Many architectures or framework were proposed in order to support the development and to ease the implementation of context-aware systems. Special thrust has been put on characteristics related to the application domain and techniques used. Therefore, a survey of such architectures makes comparison between them. Their evaluation is strongly recommended. Earlier surveys are either restricted to a limited number of architectures or they do not offer a good comparison or their evaluation is not based on appropriate criteria which keep them as mere descriptions. The present survey/research is made with the aim finding out the relevant architectures which mark the evolution of context-aware systems based on criteria related to pervasive computing. The present study would serve as a guide to developers of context-aware systems and help them to make architectural choices.

Keywords: Context-aware, Pervasive Computing, Context-Aware Computing, Framework.

I. INTRODUCTION

Information Technology has rapidly brought transformation not only in scientific field but also commercial and personal spheres of individual and social life. Advanced technology is used in smart phone to make the conversation more apt, instant and superb; which is known as Pervasive computing (or ubiquitous). It aims to provide proactively adapted services to both user and applications according to the universal context. The main characteristic of devices in such system is their context awareness. Since its inception, pervasive computing has required tools, methods and concepts to support the context awareness system. It brings ease in their design and implementation. Its tools include architectures, frameworks and middleware. The system architecture is created early in the development process. It permits the creation of a high level design of the system to ensure smooth fulfilment of requirement. The architecture design is an important step in the development of context-aware systems. In view to add in-depth and pervasive utility to the context-aware systems, many researchers have proposed several architectures, frameworks and middleware. The main emphasis of these researchers is undoubtedly on application domain and techniques used. Many surveys and researches are done to evaluate these proposed architectures; but they could not cover all architectures that mark the evolution of pervasive computing especially in regard of application domain. These surveys did not offer concrete comparison or evaluation. Instead they are simple descriptions. Even if comments about pervasive computing exist in a limited number of surveys, it appears in cursorily. These surveys were not based on criteria related to particularities of the pervasive computing. Thus, the aim of the present research is to make a survey of relevant architectures that mark the evolution of context-aware systems. The present research takes in its ambit the localization-aware systems up to present context- aware systems. It attempts to present a comparison and evaluation of architectures on various criteria which are considered important for pervasive computing. These include context abstraction level, communication model, reasoning system, extensibility and reusability. One of the main objectives of the present research is to come up with a survey that will serve as a guide to developers and architecture designers of context-aware programmes.

The rest of this paper is organized as follows, in section II we include the concept of context aware computing and in section 3, basic need of framework and section 4 contains surveys done until now on context-aware architectures and show their feature and weaknesses. In section 5, we present the comparison of the framework depending on various parameter and criteria used, and argue their use in pervasive computing and show their strengths and weaknesses. At last we conclude this paper.

II. CONTEXT-AWARE

According to Researchers Schilit and Theimer (1994) the context is referred to context as location, identities of nearby people and objects, and changes to those objects. Ryan, Pascoe(1998) defined context as the user’s location, environment, identity and time. Dey and Abowd’s definition (2001) also defined that Context is any information that can be used to characterize the situation of an entity. Again another researcher Brown, Bovey (1997) defined context as location, identities of the people around the user, the time of day, season, temperature, etc. An entity is a person, place or object that is considered relevant to the interaction between the users and corollary interaction between human and computers in socio-technical systems, which takes place in a certain context referring to the physical and social situation in which computational devices and environments are embedded. The context is of paramount importance. It is often determined by the people involved with their background knowledge and their intentions. It also encompasses the objective of the interaction and the time and place where the interactions occur. An important aspect of context-aware systems is how the information representing the context is obtained. Many activities which people are engaged in take place inside a computational environment of today's advanced world of simultaneous and pervasive computing. It can be tracked and analyzed by software components if the information about activities, time, and location are provided. It is emerging and is unbounded field. Some context parameters may be inferred from partial designs but the intent of the designer may need to be articulated explicitly via specification components. The objectives and purposes of the context information depend upon the important aspect of context-aware systems. Rather than creating problem by overloading excessive information, the context should be used to say'right' information, at the 'right' time, in the 'right' place, in the 'right' way to the 'right' person**.**

Context information is almost any information available at the time of an interaction. It can be termed as context information. Some examples of the context information are**:** identity and spatial information which includes location, orientation and speed. Secondly acceleration-temporal information includes time of the day, date and season of the year etc. Environmental information includes temperature, air quality, light or noise level etc while availability of resources include battery, display, network and bandwidth.

III. DEFINITION OF FRAMEWORK

The next one of the most important aspects is the framework. It is the part of building; or an object which supports its weight and gives it shape. The structure of a particular system include a set of beliefs, ideas or rules that is used as a basis for making judgments, decisions, etc.

A structure for supporting or enclosing something else, especially a skeletal support used as the basis for something being constructed working definition of framework as follows, "A software framework, in computer programming, is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing specific functionality"

IV. CONTEXT-AWARE COMPUTING

Context aware computing refers to a general class of mobile systems that can sense their physical environment, i.e., their context of use and adapt their behavior accordingly. Such systems are a component of a ubiquitous computing or pervasive computing environment. Three important aspects of context include where you are, who you are with, and what resources are nearby. Although location is a primary capability, location-aware does not necessarily capture things of interest that are mobile or changing. Context-aware is used more often to include nearby people, devices, lighting, noise level, network availability and even the social situation; which include whether you are with your family or a friend from school. The context-aware computing cycle consists of discovery; which is covered as dynamic data, selection which is often location-based information and the use which covers application and adaption. Context-aware computing is the ability of a mobile user's application to discover and react to changes in the context in which they are situated. To exemplify a few concrete examples could be illustrated. To provide time-of-use assignment of engineering trade-offs such as protocol constants and layers are used. Providing a display of interesting located objects, both nearby and far away, it helps to navigate the computerized world. It also used to keep a record of located-objects and persons one has encountered, for use by applications such as 'activity-based information retrieval' which include the context at the time the data was stored to assist in retrieval. It also helps to detect nearby people, located-objects, or services that are relevant to reminders or actions set to be triggered by their presence. It could easily track a particular located-object as it moves around a region. Examples include tracking a co-worker mobile user wish to talk to and tracking the office coffee cart in order to be made aware when either is nearby.

V. DESIGN OF CONTEXT-AWARE SYSTEM

Three main factors characterize the context-aware system which includes a distinction between context sensing and usage, a set of physical components to capture context information and a set software components to handle and manage context information. It, off course, includes its aptness to contextual changes in the environment. In Baldauf et al.'s survey of context-aware systems they identified some common architectural principles. The architecture of context-aware systems depends on factors such as whether sensors are local or remote, whether the system has many or few users, the available resources of the used devices (mobile devices vs. high-end PCs).

Another main architectural driver in context aware systems is the method of context-data acquisition. Three commonly used approaches are direct sensor access, middleware infrastructure and context server. In the direct sensor access approach, the system gathers the desired data directly from its sensors without preprocessing or using some other layer. Sensors acquisition is hard-coded in the clients, which gives a tight coupling between the clients and sensors. This approach is often used in devices where sensors are locally built in and not suitable for distributed systems. In the middleware infrastructure approach the sensor data acquisition is done through a middleware component. This approach hides low-level sensing details and avoids hard-coding in the clients. It improves reusability of sensors among different clients and simplifies extensibility of clients. The context server approach allows clients to remotely access data sources.

This approach introduces an access managing remote component, which facilitates concurrent client access of sources. The advantage of this approach is reusability of sensors by remote clients, which also relieves clients from resources intensive operations involved with acquisition. Thus, this approach allows clients to be thin.

VI. CONTEXT PROVIDER

Context Provider is a framework that collects, analyzes, and archives the daily context extracted from on-device sensors, web services and social media. As shown in the Figure No 1, Context Provider layers between underlying sensors and various applications. It consists of an online supervised context learning component, an adaptive frequency polling module, a long-term data repository, the capability to offload analyses for non-time-sensitive data trends, a graphical user interface for user feedback and configurations and context aware medical applications [1].

A. Supervised Context Monitoring

Context Provider can prompt users for feedback confirmations to categorize events as following the current state and data patterns from biological and environmental sensors. For example, if the heart-rate monitor encounters an increase in pulse rate with unfamiliar data patterns from other sensors, Context Provider will prompt the user to verify whether the user needs medical attention. If medical attention is not required, the user is prompted to identify the current activity, so that various sensor readings and user feedback can be correlated to build a new context. Thus, the next time the heart-rate monitor detects an increase in pulse rate with similar data patterns from other sensors it will infer the user-supplied context. Threshold parameters are adjusted automatically to reduce the rate of user confirmations.

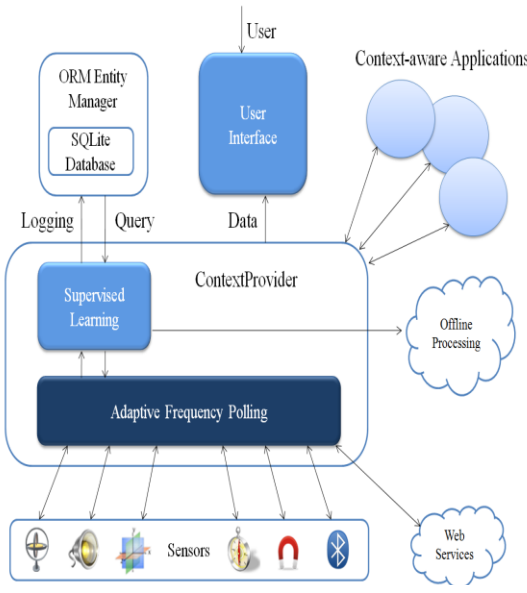


Fig.1.Context Provider System Architecture

B. Self-Adaptive Sensor Polling

Frequent readings from sensors may shorten the battery life significantly in sensor-rich smartphones; therefore, one objective for Context Provider is to poll data infrequently, while yielding accurate contexts. This frequency varies by sensor and the context to be analyzed. For example, accurate analysis requires polling rates in excess of 50 Hz, while footstep detection requires only 20 Hz polling.

VII. CONTEXT-AWARENESS IN ANDROID

Smartphones are ideal for context-aware applications because they are relatively powerful and contain various sensors. Before we decided to go for Android for our context-aware framework, we evaluated other smartphone platforms: iPhone, Symbian, RIM, Windows phone and Linux. iPhone and Android are the two most promising platforms. It is because of their popularity, high usability, powerful CPUs and available sensors. The Android platform is preferred to iPhone because it uses Java as the main programming language. Again there were several machine learning frameworks available for Java. It provides access to more core OS functionality. Also it does not require any certification or developer registration to deploy the software to hardware and fourth mostly the Android SDK is available on multiple platforms. The context support in Android application framework consists of two main parts: Raw context data sources and Context processing. The support for raw context data sources contains a lot of packages and classes such as for the camera, Bluetooth scanning of nearby devices, sensor manager for controlling interaction with physical sensors on the Android device, geographical location, time, and sound recording. The sensor manager enables Android applications to access a wide range of sensors: accelerometer, light, magnetic field, orientation, pressure, proximity, and temperature. The Android application framework provides a very good starting-point for development of contextaware applications, but it lacks a generalized interface for context management and a discovery component for adaption. Further, if proactive context-aware applications are to be developed, machine learning is necessary to recognize previous context patterns. Separation between context acquisition and usage is very important for context-aware system architectures. Such separation of concern is well supported in the Android application framework through the broker architecture that provides an intent-based communication between components. The Android application framework uses a middleware infrastructure for context acquisition providing interfaces for various sensors in such a way that no data is accessed directly from the hardware. Further, access to remote context servers are supported in Android through various network APIs as well as specific APIs such as for Google Map. [2]

VIII. SERVICE-ORIENTED CONTEXT AWARE FRAMEWORK

Following the paradigm of service-orientation, the functions of the system are separated into distinct units that are accessible as services over the network. Contexts can be found in the lowest tier in the framework that contains the hardware and software components for context perception. The collected context values are forwarded to the Framework core. Intelligent contexts (e.g. location) can perform the framework's commands or reply its requests, e.g. to send context information continuously, on demand or in bursts. In the context-aware system we encounter several types of contexts, either simple or complex ones. The contexts in this tier may be inhomogeneous, requiring different interpretation and handling. The following list shows examples of possible contexts:

- Presence: is the user available at the given moment? Is a certain context of the user available at the moment?
- Time: it can be local time (hours and minutes) or the current season, decade, etc.
- Means of transportation: on foot, by car, using public transport
- Social issues or status: is the user a child, a pensioner, a worker
- Operating system: the contexts of the device used are also contexts of the user

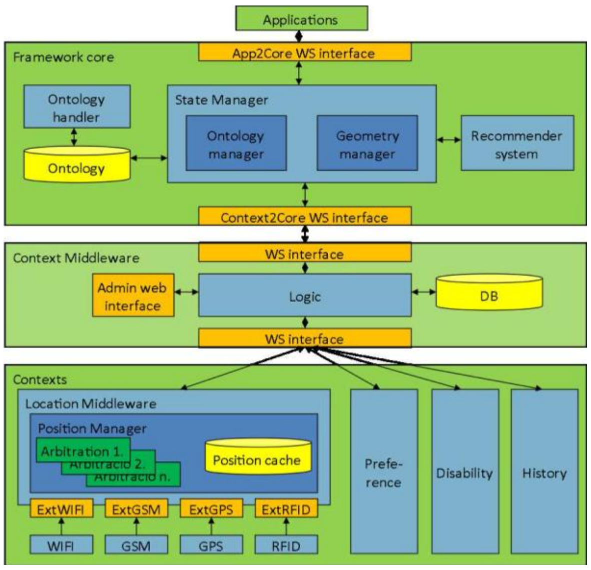


Fig.2.The components of the framework

The location context is a complex subsystem managed by the Location Middleware. It contains all the different, non-compatible positioning technologies and masks the differences so the framework can integrate any kind of positioning system via the Location Middleware. This module enables users to profit from the simultaneous or alternating usage of their different positioning devices. In an area covered with RFID readers and GSMsignal, high precision global positioning is impossible with the existing tools. However, by using a simple mobile phone and an RFID tag our system is able to determine the global position with high accuracy although neither the GSM nor the RFID technology is capable of doing this by itself. Using this module it is possible to achieve an increased precision in localization by automatically combining the sources of different positioning devices. Another benefit of this module is the possibility to detect the users' location in changing conditions, for users switching from one positioning device to another (e.g. caused by signal loss). This is useful in scenarios where both indoor and outdoor positioning is needed. Outside the user can be positioned by his GPS while getting indoors (and losing the GPS signal) the positioning continues as the GSM or with other indoor positioning systems that become accessible.

Our system was designed in a way that is able to represent location information acquired from various positioning sources. This ensures the compatibility with existing and future positioning solutions. Within our project two scientific groups are researching alternative positioning technologies and building prototype systems. One group for RFID and the other one for WLAN based positioning. We cooperated with these teams to inspect their solutions and to integrate the positioning solutions into the framework. For the position data input we provide a Web Service that accepts location data input using a fixed parameter set. Apart from raw coordinate data, location data also includes the coordinate system, precision and probability information. So as to accommodate to the model of the framework, each team developed an extension to their system (Ext RFID and Ext WLAN according to the previous sections) that sends the position information to the Web Service.

The goal of the Context Middleware tier between the Framework core and the contexts is to mask the differences between the different context types and allow the unified context handling and querying for the Framework core. This module transforms the heterogeneous context data into a homogeneous, semantic representation enabling the Framework core to operate independently from the structure of the different contexts and ensuring that the system can easily integrate future contexts. It also forwards the framework's requests to the intelligent, interactive contexts. The Context Middleware contains a database that matches the context-devices with their owner (a user in the system) and with the supported contexts. This database also stores other device- and context related information. As the Context Middleware hides the low-level devices from the upper tiers, it enables the Framework core to work with high-level user-assigned context data so it does not have to deal with low-level devices.[3].

IX. CLASSIFICATION OF ARCHITECTURE

- Direct sensor access: Tightly coupled, No extensibility
- Middleware : Hiding low-level sensing details, Extensible
- Context server – Permit multiple clients access to remote data sources**.** They relieve clients of resource intensive operations; again it has to consider appropriate protocols, network performance, quality of service parameters.

X. CONTEXT AWARE ARCHITECTURES

C. A. Sensay,
SenSay introduces the following four states: Uninterruptible, Idle, Active and the default state which could be termed as Normal. A number of phone actions are associated with each state. For example, in the Uninterruptible state, the ringer is turned off. A closed

architecture was adopted with five functional modules which include the sensor box, sensor module, decision module, action module and phone module. The figure 3 shows the following components. The sensor box collects physical sensor data from left to right; the software based sensor module queries that data the decision module determines the phone's state; the action module sets that state and the phone module provides access to the mobile phone operating system and user interface[4].
The primary purpose of the action module is to support some basic operations on the phone.

- Ringer control:off/low/medium/high
- Vibrate control: on/off
- Send SMS to a caller
- Make call suggestions
- Provide access to the electronic calendar

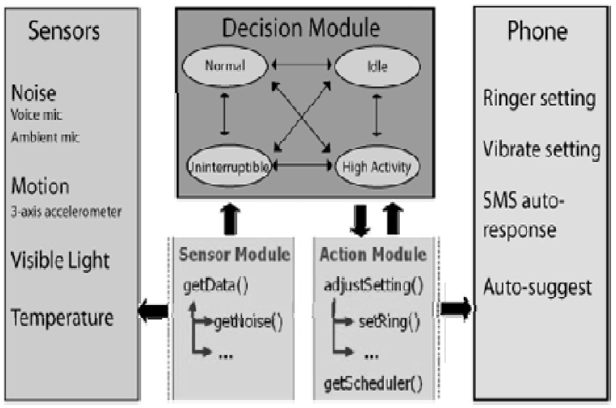


Fig.3. SenSay architecture depicting the 5 modules.

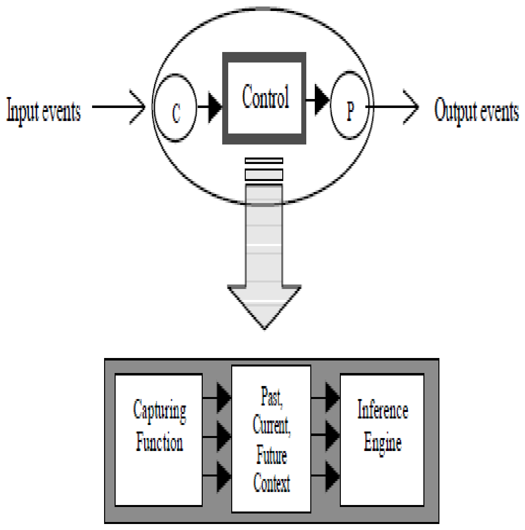


Fig.4.The Sentient Object Model

D. B. Cortex
The Central to the CORTEX architecture is the notion of a sentient object which is defined as an entity that is able to both consume and produce events as shown in figure 5. That is, sentient objects could be defined as the objects that receive events as input, process them and generate further events as output. Input events are received either from sensors or from other sentient objects.

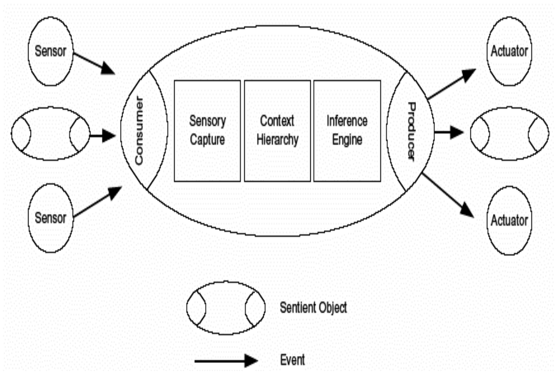


Fig.5.CORTEX Architecture

Similarly, output events are sent either to actuators or other sentient objects. Context-aware middleware approach ,Architecture is based on the Sentient Object Model Supports a graphical development tool for building sentient objects specify relevant information about sensor acutuator, Specify context hierarchy and production rules ,no need to write any code[5].

E. C. Muffin
Muffin is the prototype of mobile device for studying context awareness. It was developed in the collaboration work with Nokia Research Center. The significant characteristic of Muffin is its sensing capability. It has thirteen kinds of built-in sensors in the PDA sized box. It also has two kinds of externally attached sensors; which can be roughly divided into four groups. The first group is environmental sensors. It works as a relative humidity sensor and a barometer. The second group is physiological sensors; which include an alcohol sensor, a pulse sensor, a skin temperature sensor and a skin resistance sensor. The third group is motion and location sensors. It is a compass or tilt sensor. It has a 3D linear accelerometer, a grip sensor, an ultrasonic range finder and a GPS receiver. The ultrasonic range finder and the GPS receiver are externally attached as optional sensors. The last group includes remaining sensors which are an RFID reader, front/rear cameras and a microphone. Linux operating system runs on Muffin. Therefore each sensor can be accessed as a device file (e.g., /dev/AccelX). Also Muffin equips ordinary user interfaces (e.g., touch screen, micro joy stick, microphone, vibration motor) and connection interfaces (e.g., IrDA, Bluetooth, wireless LAN, USB port) [6]

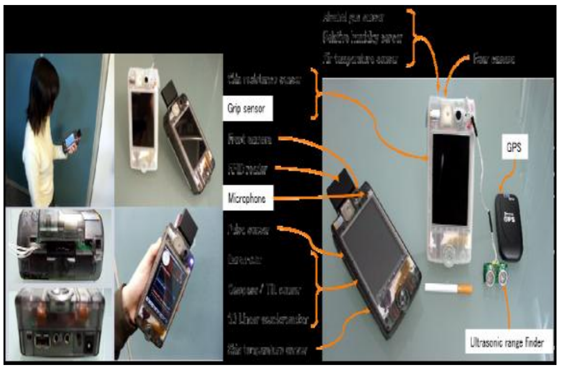


Fig.6.Muffin terminal and available sensors

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 11, November 2015

F. Citron

In the implementation of citron architecture, the blackboard architecture is employed to coordinate context analysis modules. Further, each context analysis module is defined as a worker for exclusive context extraction. The blackboard architecture is a data centric processing architecture. It has originally been developed for speech understanding and artificial intelligence[12]. There are one shared message board and multiple worker modules for collaborative data analysis. Each module reads information from the message board as a resource. After processing the data it writes the result to the board. Thus, module communication is established without the knowledge of other modules. Further, extracted data is analyzed and complimented in the communication process [6].

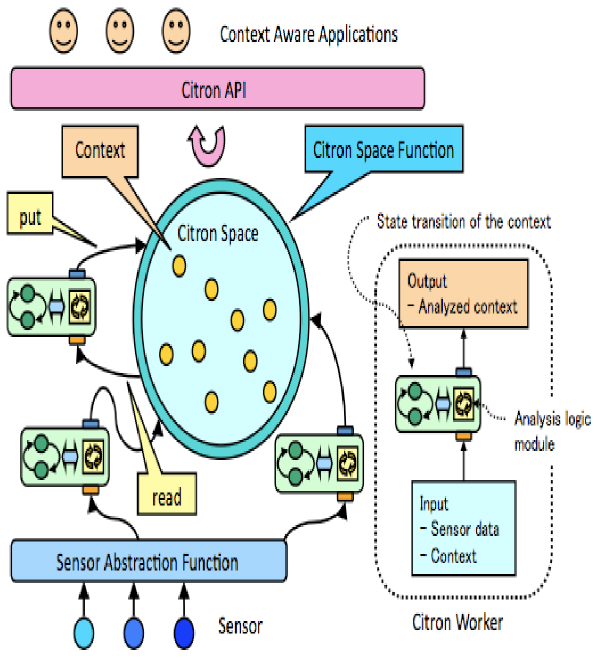


Fig.7. Citron architecture overview

G. Gaia

Middleware infrastructure: –Gaia extends typical OS concepts to include context-awareness.

Middleware is used for context awareness and semantic interoperability. It has been integrated into Gaia. Gaia is the infrastructure for smart spaces, which are ubiquitous computing environment. It encompasses physical spaces. The main aim of Gaia is to make physical spaces like rooms, homes, building and airports intelligent, aid humans in these spaces. Gaia is an agent based ubiquitous software platform project which is aiming for providing user-centric, resource-aware, multi-device, context-sensitive mobile services in active spaces, extending context-awareness to traditional operating system concept. Gaia coordinates heterogeneous software entities and networked devices. Gaia represents context data using 4-ary predicates in the format of Context (<Context Type>, <Subject>, <Relater>, <Object>).

Synthesizer infers high-level context data from low level context data [15].

H. Hydrogen

Hydrogen [12] is another open and extensible framework to support context-awareness on mobile devices. Hydrogen is similar to Context Toolkit in that the aim of Hydrogen is also providing reusability of software components. It differentiates between a remote and a local context and allows context sharing among devices so that sensed context information can be provided for more than the device connected to the sensors. Hydrogen does not support resource discovery and persistent storage functionalities. Hydrogen has a there layered architecture – Adaptor Layer, Management Layer, Application Layer. All layers are located on one device, and there is no central server. Adaptor Layer is responsible for separating context storing, sensing from other layers and provides the same context information to multiple applications. The management layer performs context sharing with other devices using peer to peer communication via WLAN, Bluetooth. Hydrogen adapted object-oriented context model as its context model. Since the context model can take advantage of inheritance, various context objects (e.g. Location Object, Device Object) can be supported by inheriting the super-class Context Object. The context server stores all contextual information about the current environment of the device. Further, the context server has the possibility to share its information with other devices in range. [8][14].

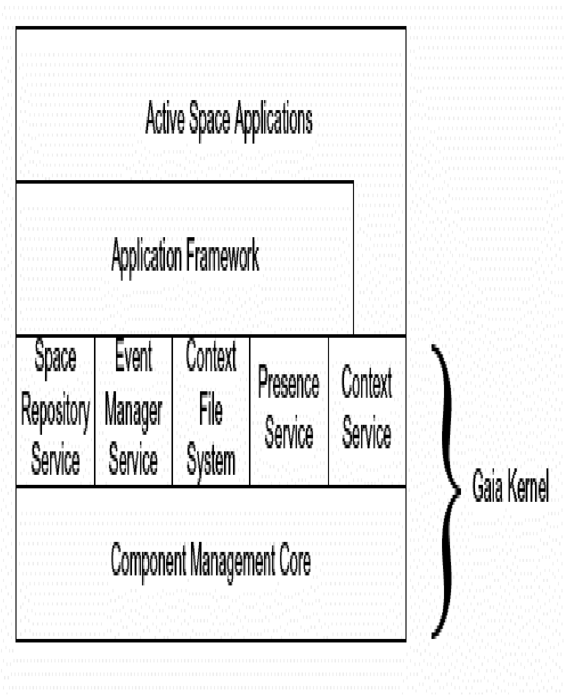


Fig.8.Architecture of hydrogen project

I. Cass

The CASS tool (Context-awareness sub-structure) is a middleware for supporting the development of context-aware applications. It provides a good abstraction of contextual information. It also uses an object oriented model for context description. The architecture (figure 10) is based on a server containing a database of contextual information and a knowledge base with an inference

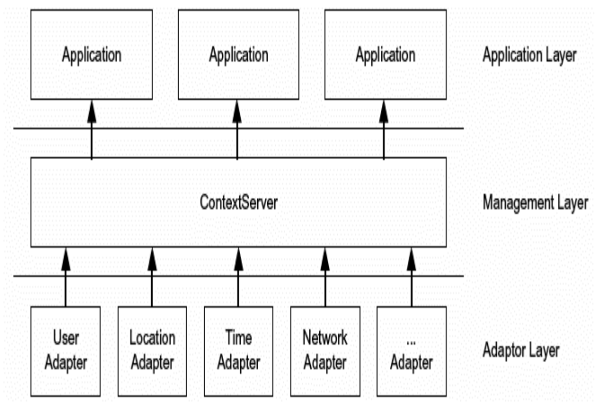


Fig.9.Architecture of hydrogen project

engine to infer other contextual information using a back chaining mechanism. The mobile devices are equipped with various sensors to perceive context variation. These work to send them to the server without local processing. Mobile devices and the server communicate via wireless mode. The server also contains a module for context interpretation that provides it with a higher level of abstraction. The architecture provides a good modularity. It allows easy modification of server components in particular the inference engine. The mobile devices do not make any processing, in fact all processing job is done by the server. It limits the autonomy needed for pervasive systems. However, it enhances the extensibility of the system. It again adds or removes devices which require only the configuration of the server. Due to its interpretation module and a reasoning mechanism, CASS also provides a good abstraction of context. It makes work more proactive way; however, the centralized architecture has its weakness too. If the server is down, all the system relied on it would be affected and becomes non operational.

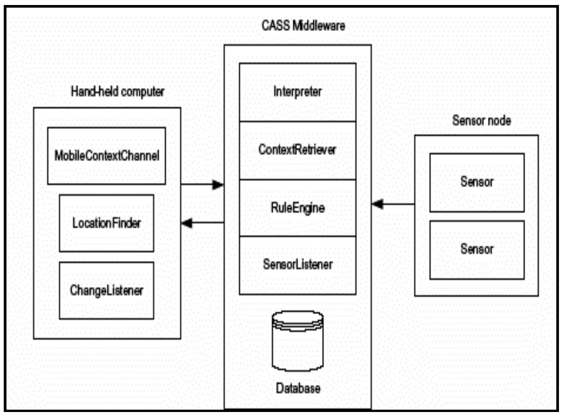


Fig.10.The CASS architecture

H. Context Management Framework

The CMF (context management framework) [11] allows semantic reasoning on context in actual time. It also works even in the presence of noise, incertitude and rapid variation of context. It delivers contextual information to applications by using a communication model based on events. The framework proposes client/server (figure 11) architecture composed of the following basic components.

- Context Manager: It is responsible for the storage of contextual information on server and the delivery of context to clients using different kinds of mechanisms (request/response, subscription/notification, etc.)
- Resource Server: It is responsible for the acquisition of contextual information from physical sensors and their interpretation according to a specific format before sending them to the context manager.
- Context Recognition Service: It responsible for the conversion of the data stream to a presentation defined in the context ontology.
- Change Detection Service: It is responsible for the detection of service change; and therefore the context change.
- Security: It is responsible for the verification and control of contextual information.

The CMF uses ontology for context representation. However it does not offer a context reasoning module. It contains a good mechanism for context interpretation. It also provides a good abstraction of context and enhances the reusability in addition to a module for context security. It uses a server for context management (centralized system). It is the main problem because when the server is down all the system would be affected. Thus, it renders the devices less autonomous. It is something not desirable in a pervasive computing system. [9]

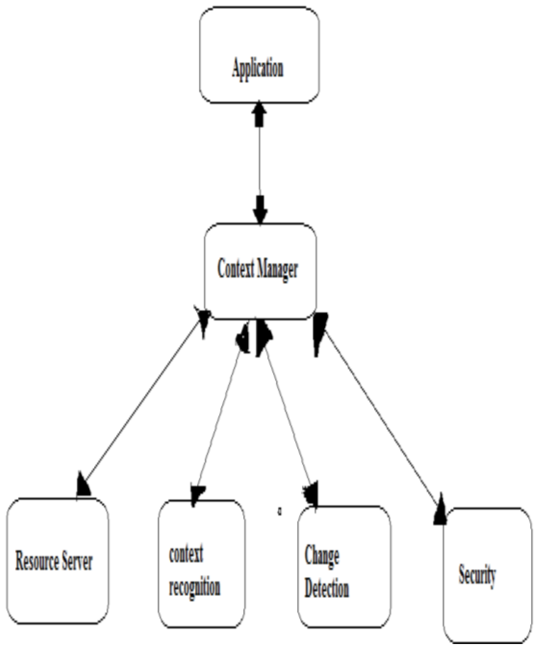


Fig.11.context management framework

J. I.The Wisdm Architecture

Data mining is generally done offline, but most sensor mining applications require results to be generated in real-time and poten-tially for a large number of users. For example, our ActiTracker activity recognition system is being designed to support thou-sands of users and provide real-time results via a web interface. Similar real time and scalability constraints exist for many other sensor mining applications, including map navigation, traffic analysis, and biometric authentication. Some applications, such as navigation, even require large numbers of users to generate useful results, since traffic information is inferred directly from the users. Finally, applications must also run on mobile devices with inhe-rently limited resources. These were our primary architectural concerns when designing the WISDM sensor mining architecture. The sensor mining process involves several steps. First the raw time-series data from the phone's sensors must be collected and stored locally. Since traditional predictive data mining algorithms (e.g., decision trees) do not operate directly on time-series data, the next step for traditional methods involves transforming the time-series data into examples that summarize the data over a fixed time period. For our current activity recognition and biome-tric applications we generate one example, with 43 fea-tures, from each 10 seconds of accelerometer data. Next, pre-built classifiers are used to generate predictions (our architecture also supports the dynamic creation of classifiers). The final step in-volves reporting the results back to the user, by sending them to the phone and/or making them available via the Web[7] .

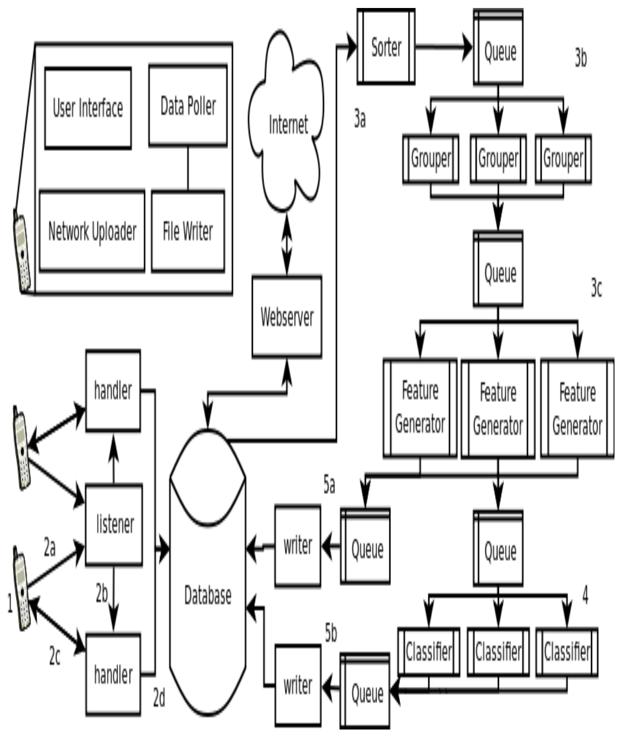


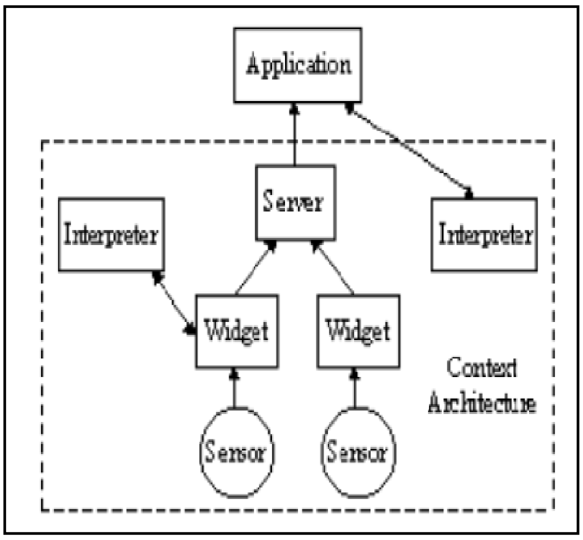
Fig.12. The WISDM Sensor Mining Architecture

K. Context Toolkit

The Context Toolkit [13] has a distributed architecture, but there still needs to be a central discoverer that is used for for discovering available widgets, aggregators, interpreters. It uses attribute-value tuple as its context data model, and stores context data and its history. The Context Toolkit provides a basic access control for privacy protection of users by providing context ownership concept.

The Context Toolkit consists of context widgets, interpreters, and aggregators. Context Widgets are distributed sensor units encapsulating the complexity of actual sensors from applications. They can be used as reusable building blocks for context sensing. Context Interpreters interpret context information to abstract low-level sensed information into highlevel information to suit the expected needs. Context Aggregators are responsible for composing context information by subscribing to context information provided by Context Widgets and Context Interpreter so that it can hide even more complexity about the context.

Service: executes actions for applications framework is easy to implement, offers a distributed communication among system devices and reusable widgets but the discovery mechanism is centralized which does not make it a perfect peer-to- peer communication model. It has a limited extensibility when the number of devices increases. The architecture takes into account events (to notify context variation) by using a thread for each event which overloads the system and affects its performance. The architecture does not contain a layer or a module for context reasoning because the model used for context representation (key/value) does not permit a good reasoning [8].



XI. COMPARISON OF CONTEXT-AWARE FRAMEWORKS

TABLE 1: COMPARISON OF FRAMEWORKS

Tools	Parameter			
	Resource discovery	Context processing	Context model	Historical context data
Sensay	√	√	√	√
Cortex	√	√	√	≠
Muffin	√	√	√	≠
Citron	√	√	√	≠
Gaia	√	≠	√	√
Hydrogen	√	√	√	√
CASS	√	√	√	√
CMF	√	√	√	≠
Context toolkit	√	√	√	√

√=present. ≠ =absent.

XII. CONCLUSION

Context awareness is a vital feature of applications in computing. In this paper, we presented relevant context-aware framework that were proposed to support and ease the development of such system. For each architecture, we discuss its criteria and feature that are related to pervasive computing. This study shows that most of the proposed architectures are layered which allows the separation of context acquisition and context use in order to increase the level of context abstraction and hide the physical sensing complexity. This enhances reusability of the system. In order to offer proactive systems, architectures embed a reasoning system to ease adaptation task which is not present in all architectures but it becomes a important requirement for future systems. This study aim is useful for recommendation to developers and designers of context-aware computing systems and helps them decide on available framework choice.

REFERENCES

[1] Michael Mitchell, Christopher Meyers, An-I Andy Wang, Gary Tyson" ContextProvider: Context Awareness for Medical Monitoring Applications"

[2] Alf Inge Wang, Qadeer Khan Ahmad" camf – context-aware machine learning framework for android"

[3] L´aszl´oKov´acs P´eter M´at´etelki Bal´azs Pataki, Computer andAutomation ResearchInstitute of the Hungarian Academy of Sciences Department of DistributedSystems 1111 Budapest, Kende u. 13-17, Hungary" Service-oriented Context aware Framework.

[4] Daniel Siewiorek, Asim Smailagic, Junichi Furukawa, Neema Moraveji, Kathryn Reiger, and Jeremy Shaffer" SenSay: A Context-Aware Mobile Phone" Human Computer Interaction Institute and Institute for Complex Engineered Systems Carnegie Mellon University Pittsburgh, PA 15213, USA

[5] hector a. duran-limon, gordon s. blair, adrian friday, paul grace2, george samartzidis," context-aware middleware for pervasive and ad hoc environments" Computing Department, Lancaster University, Bailrigg, Lancaster LA1 4YR, UK.

[6] Tetsuo Yamabe and Tatsuo Nakajima" Possibilities and Limitations of Context Extraction in Mobile Devices: Experiments with a Multi-sensory Personal Device" International Journal of Multimedia and Ubiquitous Engineering Vol. 4, No. 4, October, 2009.

[7] Shaun T.Gallagher, Andrew B. Grosner, Tony T. Pulickal" Design Considerations for the WISDM Smart Phone-based Sensor Mining Architecture"Department of Computer and Information Science Fordham University 441 East Fordham Road Bronx NY 10458

[8] Sangkeun Lee, Juno Chang and Sang-goo Lee" Survey and Trend Analysis of Context-Aware Systems" School of Computer Science and Engineering, Seoul National University, Seoul 151-742, Republic of Korea

[9] Moeiz Miraoui1, Chakib Tadj, Chokri ben Amar" architectural survey of context-aware systems in pervasive computing environment', Ubiquitous Computing and Communication Journal.

[10] P. Fahy, S. Clarke, "CASS – a middleware for mobile context-aware applications", Workshop on Context-Awareness,MobiSys 2004.

[11] P.Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen, E-J. Malm, Managing context information in mobile devices", IEEE Pervasive Computing, Vol. 2, No. 3, July–September, pp.42–51, 2003

[12] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, "Context-awareness on mobile devices – the hydrogen approach", Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2002 pp.292–302.

[13] A. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications", Human- Computer Interaction, 16:97–166, 2001.

[14] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann" Context-Awareness on Mobile Devices- the Hydrogen Approach" Software Competence Center Hagenberg Hauptstraße 99, A-4232 Hagenberg.

[15] Anand Ranganathan and Roy H.Campbell"A Middleware for context aware agents in ubiquitous Computing Environments" University of IIIinois-Champaign,USA