A Brief Overview of Context Aware System

¹R Suresh Kumar, ²R. Mohandas, & ³Jerome christhudass

¹Department of ECE, Chennai Institute of Technology, Chennai, Tamilnadu, India.

²Department of ECE, Balaji Institute of Technology & Science, Warangal, Telangana, India.

³Department of Biomedical Engineering, Karunya Institute of Technology and Sciences, Coimbatore, Tamilnadu, India.

***Corresponding Author: jeromechristhu@karunya.edu**

Received: 02.01.2021,

Revised: 05.03.2021,

Accepted: 15.05.2021,

Published: 16.08.2021

DOI:

10.53409/mnqa/jcsit/2203

Abstract: Context Awareness emerged as a concept from Ubiquitous computing, which is becoming a reality by emphasizing the integration of the data space and the physical space. With its aid, people may receive and analyze data at any time and from any location using a device that can connect to the internet. As a result, it can reduce the complexity of using the gadget and make people's life easier and more efficient. Context aware systems (CAS) are an effective approach for dealing with day-to-day tasks. Context aware frameworks provide up completely new possibilities for applications developers and end users by collecting contexts data and changing system behaviour respectively. This survey presents an overview of context aware systems. This study analyzes the concept of context aware systems, network architecture, application, and user interface, as well as thorough information of every layer of context aware system. As a consequence of the survey, a general procedure layout in CASs is provided, and the architectural contemplations of CASs are clarified.

Keywords: Context framework; Context-awareness; Sensors; Context middleware; CAS Architecture.

I. INTRODUCTION

Context awareness is the ability to utilize context data. A system is contexts aware if it could understand, extract, and utilize context data and adjust its performance to the present contexts of use. The name context aware computing was often known by people who work in contexts aware, where it was believed which context was a source in its endeavours to disperses and directly interlace computers innovation into our life. By detaching users from devices, developing pervasive or ubiquitous computing technologies enables "anytime, anybody, anywhere" computing. To provide adequate user assistance, services and applications must be aware of their contexts and, as a result, respond to their changing circumstances, which was referred to as context awareness. The context is important because it provides information on the current state of people, places, technology, and natural phenomena. The context is a data that may be utilized to explain the condition of a material. An element is a place, person, or object that was thought to be important to the communication among a user and application, such as time, locations, activity, and the priorities of every entity [1].

One objective of CAS was to collect and use data on the contexts of a device to offer services that were suitable to the unique

persons, time, location, events, and so on. These frameworks intend to offer contexts-aware communications, data, and computational access [2]. CAS differs from traditional frameworks in that it provides remarkable characteristics like as heterogeneity, high complexity, and AI [3]. Because the characteristics that define a contexts-aware framework like area and day length, were rapidly changing. As a result, the usage and relevance of CAS was growing, and it is already playing an important part in future wireless systems.

It has been identified as essential and highly beneficial for a variety of reasons, including the fact that the context may reduce input costs, provide an invigorating user experience with little efforts on the user's side, and users could benefit from sharing contexts [4].

II. CONTEXT AND CONTEXT AWARE SYSTEM

"A system is context-aware if it utilizes context to give related data and/or services to the client, where relevancy based on the client's work." Where, context is any data which could be utilized to classify the present condition of a user [5].

The presented context is made of many elements, out of which can be simply distinguish as:

Location: Position, direction, speed, and so forth.

User Identity: Profile, inclinations, biometrics, social data and so forth.

Time: Present date and time or future occasions, duration and so forth.

Activity: Walking, resting, sitting, and so forth.

Current Task: Work or social gathering, wellness, studying, and so forth.

Environment: Temperature, humidity, light and noise levels.

Hardware: Present device data, network, and encompassing devices [6].

CAS may be divided into two broad categories: local and distributed. Local systems serve as frameworks in those applications, and sensors are tightly integrated (by a direct physical association). A distributed framework, on the other hand, does not have the direct physical link among the application and sensor. Because of the improper link, it could have many applications taking data from a single sensor. Furthermore, data generated by the multiple scattered sensors can be utilized by a single application. There are two types of distributed systems: collaborative and non-collaborative. Distributed collaborative framework is a system that enable more than two distant persons to work together to obtain a common target. Non-collaborative frameworks, on the other hand, exclusively promote individual goals.

From a practical standpoint, CAS may be viewed as a tiered system comprised of raw data retrieval, sensors, preprocessing, an application layer, and storage. The Context Management

System (CMS) is in charge of collecting original data from sensors, abstracting and aggregating the discovered data into high-level contexts, and making it accessible to contexts-aware application [7].

Table 1. Architecture Types of CAS

Author	Winograd	Chen et al.	Perera et al.
Year	2001	2004	2014
Architecture Type	Widgets Blackboard Model Networked Services	Direct Sensor Access Context Server Middle ware-Based	Component-Based Node-Based Service-Based Distributed Client-Server Centralized

III. ARCHITECTURE OF CAS

CASs representative architectural styles may be classified into three types: distributed architecture, stand-alone architecture, and centralized architecture. Because each style has unique characteristics, benefits, and downsides, it is critical to choose the appropriate architecture style while designing context-aware frameworks. Several components of the context-aware framework may be dependent on its design approach [8].

Stand-alone, which is a critical architecture that specifically gets to sensors, does not take into account device context sharing. This architecture could be usually easily realized, but it had limitations due to the fact that it could not operate device coordinated effort. This design is appropriate for less

complex, simple, or domain-specific applications.

Distributed architecture Context-aware frameworks with distributed design may store context data in a large number of isolated devices, with no additional central server. Every device is free of diverse devices; hence, the CAS might disregard certain less important devices that have a blockage issue and still proceed with context aware activities. Every device deal with context data and exchanges context data with other devices by interacting via different devices; hence, ad hoc communication protocols are necessary. In any event, using ad-hoc communication standards makes it difficult for devices to understand the overall situation of each device. Cell phones often require computation control and resources; moreover, a distributed architecture is one of the constraints in handling computationally significant applications [9].

Figure 3. Centralized Architecture

Devices and sensors are connected with a centralized context server that has rich resources and computational control, and context data may be kept in both a central server and user devices. If a device needs context data from another device, it requests it from the central server and receives the result. Because each communication in this architecture is conducted by querying the context server, the transmission protocol might be simpler than in a distributed design. Many applications that need a lot of resources and money might be handled by using a computationally powerful device as a central server. Regardless, there is a disadvantage to this technique in that it can be significant if the central server fails or there is congestion [10].

IV. GENERAL PROCESS IN CONTEXT AWARE SYSTEMS

Various CASs have complex structures and multiple subcomponents that are in charge of representing, reasoning, controlling, and analyzing context data, and they provide their capabilities by the combination of the subparts in the frameworks. Regardless of the fact that there are numerous types of contexts aware frameworks, CAS typically follows four phases to process context awareness [11].

Similarly, the personalization and suggestion ideas into Schilt's classifications are:

- Context-Aware Personalization- Providing customized substance or data dependent on user's present context data.
- Context-aware Suggestion- Providing recommendations to user's conduct dependent on the user's present circumstance.
- Automatic Device Configuration- Automatically setting up device's specifications as per user's present circumstance.
- Context-aware User Interface Optimizing UI dependent on the user's present context [5].

Context-aware systems may be realized from a variety of perspectives. The approach is dependent on certain conditions and requirements, for example, the sensor's location (local or wireless), number of potential customers (one or more), resources available from devices used (hi-fi PCs or smartphones), or the decision to expand the framework. Furthermore, the technique for acquiring context information is critical when constructing CASs since it predetermines the compositional design of the framework to a limited extent in any scenario. Chen [14] outlines three distinct techniques for obtaining context data.

Direct sensors access- This method is frequently utilized in sensor devices that are in-built. The user programming accumulates the optimal data from the sensors directly, i.e., there was no extra layers for gathering and preparing data from sensor. Sensor drivers are built into the application. Therefore, this tightly linked technique is only viable in exceptional circumstances. As a result, it is unsuitable for suited frameworks due to its access quality directly, which falls short a component equal to coping with many concurrent sensor's access.

Middleware infrastructures- Present programming configuration use review strategies to segregate, for example, graphical user interfaces and business logic. The middleware-based technique familiarizes a structured design with CASs with the goal of covering lower-level sensing information. In contrast to sensor access directly, this method allows for greater flexibility when the user code no longer has to be altered, and it rearranges the hardware-dependent sensing codes reusability due to the stringent inspection.

Context servers- The following consistent process was used to grant numerous users accesses to wireless data source. This technique extended the middleware-based design by offering accesses monitoring wireless segment. The collection of sensor data was transferred to this pretended context servers in order to assist simultaneous multiple accesses. Aside from sensor re-usage, the usage of a context server offers the advantage of relieving users of resource intensive chores. This is an important topic to consider because most of end devices utilized in CASs were likely mobile devices with limited memory, processing powers, and so on. As a result, while designing a CAS based on user server architecture, one must consider appropriate conventions, network execution, the type of service parameters, and so on [11].

The first layer is made up of a collection of diverse sensors. It's worth noting that the term "sensor" refers to both sensing equipment and any source of data that might provide useful context data. Sensors may be classified into three types based on how information is captured. Physical sensors are the most prevalent type of sensor. There are several equipment sensors available these days that can be used for recording all physical information. Table.2 represents several physical sensors with examples.

Sensors that operate virtually- Virtual sensor obtains contexts information from services or programming applications, i.e., it can be determined a representative's location by utilizing tracking system (physical sensor) as well as the virtual sensors, for example, by utilizing the electronic logbook, emails, travel booking system, and so on, for location data. Another context's features that virtual sensor could detect include, for example, the client's behaviour via watching movement of mouse and input from keyboard.

Logical sensor- This sensor utilizes a limited number of data sources and integrate virtual and physical sensors with more data from other sources or databases to explain high-level operations. A logical sensor, i.e., may be created to determine a representative's present position by analyzing login

at computers and devices database mappings to location data [12].

Table 2. Types of Physical Sensors

Context Type	Available Sensors
Light	Infra-Red, Photodiodes, Ultraviolet, Colour Sensors, etc.
Audio	Microphones
Visual	Different Cameras
Location	Indoor: Active Badge System, etc. Outdoor: GPS, GSM
Motion, Acceleration	Angular Sensors, Magnetic Fields, Accelerometers, Mercury switches, Motion detectors.
Temperature	Thermometers
Touch	Touch sensors
Physical	Biosensors to measure blood pressure, skin resistance, etc.

Table 3. Context Types and Related Virtual and Physical Sensors

Context	Physical Sensor	Virtual Sensor
Time	Clock	Operating system timer
Identity	Retina scanner, Fingerprint reader, microphone, RFID, Smartcard reader	Various authentication models at the application level or operating system.
Location	Indoor: Bluetooth, 802.11 cells Outdoor: GPS, GSM	User's login on locations aware system, Networked calendar system, IP subnet, Travel booking system.
Activity	UV-sensors, camera, accelerometer, mercury switch, thermometer, motion detector.	Mouse or Keyboard activity, application usage.

The second layer is in charge of regaining real-world background information. It generates appropriate drivers for APIs and physical sensors for logical and virtual sensors. The query utility is frequently actualized in reuse programming parts that make low-level hardware access details simple by providing numerous review techniques, for example, `getPosition()`. These components become replaceable by utilizing interfaces for components in charge of equivalent types of contexts. As a result, to change an RFID framework with a GPS while making no substantial changes to the present and above layers.

The preprocessing layer was not run in all CAS; nevertheless, if the original information is

overly coarse-grained, it may offer important data. This layer is in charge of reasoning and contextual data translation. The sensors questioned on the basic layer frequently produce technical information that application builders cannot use. As a result, the output of layer two is increased to the maximum contemplation state by this layer. The modifications include quantization and extraction operations. For example, an individual's precise GPS position is unlikely to be useful to an application, but the room name in which the individual is located may be.

In CASs with a varied set of context information sources, single context atom might be linked to higher state data in the layer. This method was also known as 'total' or 'combination.' The value of the single sensor was frequently not significant to the applications, but associated data may be highly valuable and precise. In this approach, the system may identify, for example, if a user was inside or outside by evaluating different physical variables such as light and temperature, or whether an individual is now heading to a meeting by gathering location and noise level. To ensure that this analysis works well, a large variety of statistical techniques are used, and some type of preparation stage is usually required.

Obviously, the application may also directly implement this review tool. Regardless, for a variety of reasons, this action must be evaluated and transferred to the context server. The evaluation advances reuse and, as a result, helps the enhancement of user applications. Furthermore, by developing such aggregators, users can wirelessly access network execution increments (as opposed to associating with many sensors) and limited user resources are stored [17].

Table 4. Types of Context Models

Context Models	Examples
Graphical model	Vector Space Model, Context extension of ORM
Key-Value Model	Context Toolkit, Schilt's Approach
Ontology Model	CoBrA, SOCAM, CoCA, CASS
Object-oriented Model	Hydrogen
Mark-up Scheme Model	GPM, CSCP, UAProf, CC/PP
Logic-based Model	McCarthy's Approach

The issue of detecting conflicts that may arise when using several information sources must also be addressed at this layer. For example, if a system is made aware of an individual's location by smartphone instructions and the camera identifying this individual, it may be challenging to select which

data to utilize. This debate was frequently brought closer by the use of additional information such as resolution data and time stamps.

The fourth layer, Storage and Management, organized the gathered information and makes it available to the user via an open interface. Users can gain accesses in two ways: asynchronous and synchronous. The user is scanning the server for changes in a synchronous manner via wireless method calls. In this manner, it transmits a message requiring a limited set of given information and waits for the server's response. Subscriptions are used in the asynchronous manner. Every user subscribes to explicit events that they are interested in. In the case that any of these incidents occurs, the user either made informed or the user's approach was directly incorporated employing the call-back. Because of quick alterations in the basic context, the asynchronous approach is becoming sensible in the majority of cases. The surveying technique requires more resources since context information must be requested on a frequent basis and the program must demonstrate for changes by itself, employing just a few types of context history [15-16].

Application layer, the final layer where the user is recognized. The actual response to numerous situations and context instances is carried out here. Data recovery and applications explicit context managements and reasoning are sometimes examined in the sort of operators that communicates with the context servers and acts as the extra layer among the application and the preprocessing layers. The display on mobile phones is an example of contexts logic at the user level: if the light sensor recognizes poor lighting, material might be shown in a greater contrast colour [12].

This context's life cycle is divided into four parts. To begin, the context should be gathered from several sources. The sources might be either virtual or real sensor (context collection). Second, the acquired data should be modelled and presented in a meaningful fashion (context modelling). Then, modelled information should be used to infer higher-level context data from lower-level original sensor data (contexts reasoning). Finally, both low and higher-level context should be supplied to context-interested users (context distribution) [13].

The first-generation context aware systems focused primarily on realizing location aware domain explicit applications, whereas the second and third generation context aware frameworks are similar in that they achieved more consensus and adaptability. They support a broader range of domains and context data. Recently, the ontological context model has been widely used in a variety of contexts since it provides convention and intelligent processing techniques [11]. Ensuring user privacy, as well as achieving execution and flexibility, have emerged as key problems in current frameworks [18-20].

V. CONCLUSION AND FUTURE WORK

In this survey, several context architectural standards and models for context aware frameworks were addressed, as well as various server-based and existing middleware techniques to deal with facilitating the evolution of context aware applications. The survey introduced the CAS idea, network architecture, application, and user interface, as well as thorough information of each CAS layer. The literatures for the concept and applications were evaluated and assessed using dimensions associated with progressing and emerging problems in CAS. Because only journal papers were extracted and analyzed, the study did not include the functions and cooperation approaches for logical components in the architectures of CoBrA, SOCAM, Hydrogen, GUIDE, CASS, Context Toolkit, Gaia, and others.

ETHICS APPROVAL AND CONSENT TO PARTICIPATE

Not applicable.

HUMAN AND ANIMAL RIGHTS

No animals/humans were used for studies that are basis of this research.

CONSENT FOR PUBLICATION

Not applicable.

AVAILABILITY OF DATA AND MATERIALS

The authors confirm that the data supporting the findings of this research are available within the article.

FUNDING

None.

CONFLICT OF INTEREST

The authors declare no conflict of interest, financial or otherwise.

ACKNOWLEDGEMENTS

The authors would like to thank their present employer for providing support while carrying out this research work.

REFERENCES

- [1]. Hong JY, Suh EH, Kim SJ. Context-aware system: A literature review and classifications. *Expert Systems with Applications*, 2009, 36(4), 8509-8522.
- [2]. Chan CS, Ostertag MH, Akyurek AS, Rosing TS. Context Aware System Design. *Micro-and Nanotechnology Sensors, Systems, and Applications*, 2017, 10194, 101940B1-18.
- [3]. Almutairi S. An Enhancement on Class Diagrams to Model a Secure Context Aware Systems. *IEEE International Conferences on Innovation in Green Energy and Health care Technologies*, 2017, pp. 1-7.
- [4]. Almutairi S, Aldabas H, Samaha AA. Review on the Security Related Issue in Context Aware Systems. *Int. J. Wirel. Mobile Netw.*, 2012, 4(3), 195-204.
- [5]. Schilit B, Theimer M. Disseminating active maps information to mobile host. *IEEE Network*, 1994, 8(5), 22-32.
- [6]. Chihani B, Bertin E, Jeanne F, Crespi N. Context-Aware System: A Case Study. *International Conferences on Digital Information and Communications Technology and its Application*, 2011, pp. 718-732.
- [7]. Loke SW. Incremental Awareness and Compositionality: A Design Philosophy for Context-Aware Pervasive System. *Pervasive and Mobile Computing*, 2010, 6(2), 239-253.
- [8]. Alegre U, Augusto JC, Clark T. Engineering Context-Aware System and Application: A Survey. *J Syst Software*, 2016, 117, 53-83.
- [9]. Kjaer KE. A survey of contexts-aware middleware. *International Multi-Conferences: Software Engineering*, 2007, pp. 148-155.
- [10]. Saeed A, Waheed T. An extensive survey of context-aware middleware architecture. *International Conferences on Electro/Information Technology*, 2010, pp. 1-6.
- [11]. Lee S, Chang J, Lee SG. Survey and Trend Analysis of Context-Aware System. *Information-An International Interdisciplinary Journal*, 2011, 14(2), 527-48.
- [12]. Baldauf M, Dustdar S, Rosenberg F. A Survey on context-aware system, *Int. J. Ad Hoc Ubiquitous Computing*, 2007, 2(4), 263-77.
- [13]. Perera C, Zaslavsky A, Christen P, Georgakopoulos D. Context Aware Computing for the Internet of Things: A Survey. *IEEE Commun Survey Tutorial*, 2013, 16(1), 414-54.
- [14]. Chen G, Kotz D. A survey of context-aware mobile computing research. 2004.
- [15]. Perttunen M, Riekki J, Lassila O. Context representations and reasonings in pervasive computing: a review. *Int J Multimedia Ubiquitous Eng*, 2009, 4(4), 1-28.
- [16]. Bettini C, Brdiczka O, Henricksen K, Indulska J, Nicklas D, Ranganathan A, Riboni D. A survey of context modelling and reasoning technique. *Pervasive Mob. Comput.*, 2010, 6(2), 161-180.
- [17]. Makris P, Skoutas D, Skianis C. A survey on context-aware mobiles and wireless networking: On networking and computing environment integrations, *IEEE Commun. Survey Tutorial*, 2012, 1(99), 1-25.
- [18]. Pooja S. G, Chaware SM. Context Aware Computing Systems: A survey. *Proceeding of the 2nd International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)*, 2018, pp.605-608.
- [19]. Yousef A. A Context Aware Framework for IoT Based Health Care Monitoring Systems. *Int J Adv Stud Comput Sci Eng*, 2020, 9(7), 1-9.
- [20]. Deebea K, Saravanaguru RAK. Context-Aware Elderly People Monitoring based on IoT. *J Xi'an Univ Arch Technol*, 2020. 7(3), 5797-5804.