International Journal of Engineering Research and Technology. ISSN 0974-3154 Volume 10, Number 1 (2017)

Pervasive Context-Aware Computing
Survey of Context-aware ubiquitious middleware systems

Author 1
Mr. Anil R. Surve
Department of Computer Science and Engineering
Walchand College of Engineering
Sangli, India
anil.surve@walchandsangli.ac.in

Author 2
Prof. Dr. Vijay R. Ghorpade
Department of Computer Science and Engineering
D.Y. Patil College of Engineering & Technology
Kolhapur, India
vijayghorpade@hotmail.com

Abstract
Pervasive computing is promising emerging endeavor which focuses on the capability for users to perform daily computer activities unobtrusively. Pervasive computing solutions are broad to encompass variety of technologies and real life applications. It provides an environment for people to interact with embedded computers. Networked devices are aware of their surrounding objects, peers and are aimed to use or provide services from peers in an effective manner. Pervasive applications leverages the existing blend of computing elements that already surrounds us. It can enrich our lives by enabling automation of mundane activities through ubiquitous applicability. It facilitates users to move seamlessly and provides services regardless of location, time or platform collaboratively and proactively. To realize this notion, various frameworks and middleware technologies of context-aware systems are needed to be explored. This paper is an attempt to explore those especially for potential social domains.

Keywords: Pervasive Computing; Ubiquitous Computing; Context Awareness, Mobile Computing ; Middleware ; Internet of Things**.**

Introduction

Mark Weiser, researcher at Xerox PARC who is known as father of pervasive computing envisioned that the 21st century will witness technological revolution which will be part of the everyday, the small and the invisible entities around us. The traditional computing profound technologies will almost disappear. There will be wearable computing elements which will weave themselves into the robes of everyday life as if they will be indistinguishable computing entities separately [1]. Till then as of now many research attempts are emerging as contribution in realizing his dream vision. Pervasive computing is integration of *ubiquitous computing*, *context-awareness*, *ambient intelligence* and *wearable computing* [2]. Context awareness has become thrust research area in computer science. The focus on context-aware computing evolved from applications of desktop, web, distributed, mobile to the Internet of Things (IoT) over the last decade [3]. In the zest of research contribution noteworthy prototypes, systems, and solutions using context-aware computing techniques are successfully enrolled. Among them most of the prosperous solutions aimed to gather data from a variety of physical (hardware) and virtual (software) sensors. Further processing and analyzing sensor data from all the sources is possible. The advancement in sensor technology is offering sensors which are more powerful, cheaper and compact for use. As a result, a large number of sensors are being deployed and it ispredicted that the numbers will grow swiftly over the next decade. Context-aware computing aims to store context information associated to sensor data for its significant interpretation as per business logic. While scaling of sensors with generating voluminous data, the conventional application based approach becomes infeasible. In order to address this inefficiency, significant middleware solutions are introduced by researchers. The notion of middleware solution also supports different essential arenas, such as device management, interoperability, heterogeneity, platform portability, context-awareness, security and privacy. This paper is a survey with the zeal of potential focus on identifying the context-aware computing concepts, features and functionalities. Also middleware techniques that are utilized for potential social domain are deemed.

Concept of Pervasive Context

Context by definition means any information which will help us to characterize the situation of an entity. An entity may be a person, place, or object which is pertinent to the interaction between a user and an application [3].

The pervasive context can also be termed as circumstance, situation, phase, position, posture, attitude, place, point, terms, regime, footing, standing, status, occasion, surroundings, environment, location, dependence etc.

To categorize as raw sensor data which is unprocessed and retrieved straight from the data source, such as sensors. Context information is generated by processing raw sensor data. Further, it is checked for consistency and meta data is

added. For example, the sensor readings produced by GPS sensors can be considered as raw sensor data. GPS sensor readings are represented as geographical location of context information. If this data can generate context information then it is treated as context data. Therefore, mostly what is captured from sensors are data not the context information [3].

Also data is extended as set of interrelated events by means of logical and timing relations among them. They also describe an event as an incidence that triggers a condition in a intended area. These events are categorized as: *discrete events* and *continuous events.* In discrete events an event happens at particular time such as a door open, lights on, object entry etc.On the contrary, continuous events are those event instances lasting for at least some amount of time [3].

Context Awareness Concepts

Context awareness imposes conscious focus on computer applications and systems. A context-aware system utilizes context to deal with relevant information and services for the intended user. The relevancy depends on the user specific tasks. The context awareness frameworks are typically meant to support data acquisition, meaningful data representation, delivery of service and reaction.
Context models recognize a tangible subset of the context that is sensibly attainable from sensors, applications and users and have capability to be used in the execution of the task. Wherein the context model built for a specific context-aware application is usually explicitly programmed by the application developer [3].

Life Cycle of Context Awareness

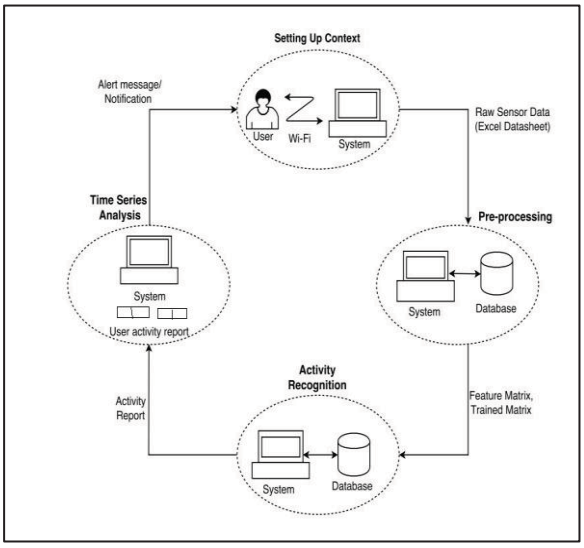

Figure 1: Generic life cycle of context awareness

Figure 1 depicts generic context information life cycles which consist of four main phases which consists of context setting, pre-processing of sensed information, task/activity generation & analysis mainly time series.

Models of Context awareness

The two main models exercised are Implicit and Explicit.
Implicit context model: applications are devised using standard libraries, frameworks and toolkits for primitive operations such as context acquisition, preprocessing, storing, and reasoning activities. It offers a standard design to follow which makes it easier to construct the applications quickly. However, still the context is hard bound to the application.
Explicit context model: applications are developed with context management infrastructure mainly using middleware technologies. Hence, actions such as context acquisition, pre-processing, storing, and reasoning lie outside the application boundaries.

Context attribute is important constituent of the context model which describes the context. It has an identifier, a type and a value, and optionally a collection of properties describing specific characteristics.

Context-awareness Features

Presentation: Context can be configured to choose what information and services needed to be accessible to the user. Providing the idea of presenting information based on context such as location, time, etc. Main motive is to support any service, anytime, anyplace, with anything and anyone preferably using any path/network.
Execution: Proactive execution of service expected automatically based on the context with node to node communication.
Tagging: Context entities are tagged together with the sensor data so as to infer meaningful information. As there exists a large number of sensors attached to everyday objects to produce large amount of sensor data that has to be collected, analyzed, fused and interpreted.

Context types: Socializing context involves framing various context types mainly user computational, environmental, historical, social networking, spatial (profiled user identity, location, time), activity based, sensor based and cognitive [4].

Context views: Context can be viewed as *primary* and *secondary*. In primary, context information retrieved directly without performing any kind of sensor data fusion operations. For example, GPS sensor readings as location information, sensor data, RFID tag identification etc. In secondary context, information is processed or computed using primary context elements by using sensor data fusion operations or data retrieval operations. For example web service calls for identifying the distance between two sensors by applying sensor data fusion operations on two raw GPS sensor values. Location, identity, time, and activity noted as important context information. The categorization is dominated by who, when, where, why, what objective factors [12].

Context awareness levels

The configuration of context awareness allows users to organize their preferences, likes and prospect values to the

context system manually. In *passive level* the system constantly monitors the environment and proposes the appropriate options to the users for actions. In *active level* the system continuously and autonomously monitors the situation and acts proactively for the profiled user preferences.

Middleware Support for Context Awareness

Middleware is employed to create a smart context environment with embedded and networked computing devices. It provides profiled users aspects like seamless service access centricity, autonomous detection of application requirements and automatic service provisioning [4][5]. As a matter of fact the heterogeneity of hardware, software, and network resources pose genuine coordination issues and demand for comprehensive knowledge of individual elements and technologies. In order to ease coordination and to assist application developers, different middleware platforms have been projected by researchers [4]. Leveraging available middleware technologies are advantageous for varied features such as management of context, data or service in application developments. The notion of middleware is to match context resource specific application-demands with service provisioning capabilities, ensuring quality and efficiency in a truly ubiquitous manner. To list out potential applications using middleware support are smart-spaces, health-care, ambient living, social networking, entertainment, logistics and intelligent transportation systems [13].

Middleware Characteristics

The requirements of pervasive applications are truly application-specific. It demands high flexibility, re-usability, reliability, localized scalability, adaptability and context-awareness characteristics. Use of middleware is primarily to hide the complexity and heterogeneity of underlying entities and their interactions in transparent manner. Also middleware need to provide abstraction into intuitive and accessible programming constructs. Also special mechanisms are available to support situation-aware services to the users. These range from raw context collection, storage and processing, higher-level context derivation, context inconsistency resolution, service discovery and composition to support users unobtrusively. It is expected that middleware layer has to be lightweight so as to fit in wireless sensor nodes or embedded portable devices which are having constrained by restricted processing powers and energy backups.

Middleware Technologies

Gaia, Aura, PICO/SeSCo, CORTEX, One.World, Scenes, Activity-oriented computing and UIO are popular middleware systems. Main objective of which is providing programming abstractions (high-level programming interfaces to the application programmer) in different design choices of the context aware middleware paradigm [4] [6] [7] [8] [9] [10]. Gaia, Aura and AoC use a component-based programming model. There are two prime types of abstraction levels, *node level* and *system level*. The node level abstracts the environment as a distributed system consisting of a collection of heterogeneous computing devices. It also provides programming support for individual devices for their actions and cooperation. This is well supported by Aura, PICO/SeSCo, CORTEX and Activity-oriented computing (AoC) middleware. Aura manages every user's personal information which is comprised of task manager, environment manager, and context observer components [13]. Aura uses task abstraction to represent user applications composed of multiple abstract services called as *Suppliers* [11]. Another type of abstraction called *Connectors* which is abstraction of interconnections between the system components. *Activity* in Aura is a specific task used which is such an abstraction of user actions planned by the activity-oriented computing model. The Aura activities are computational abstractions which need to be initiated, suspended, stored and resumed on computing element. It hides heterogeneity of the underlying computing platform. *Delegent* is an abstraction of the mobile software agent, and device is an abstraction of computing devices. *Delegents*needs to be enabled by events which are taking place in the environment.

System level abstraction enables the environment as a single virtual system. Its main notion is to facilitate the developer to articulate a singlecentralized program into subprograms. Also it is intended to work with local nodes; the programmer has to utilize minimal set of programming primitives while making transparent the low-level worries. Typically distributed code generation, remote data access and management, and inter-node program flow coordination are achieved. Gaia and One.World support system level abstraction. In Gaia, active space is used as an abstraction of the smart physical environment. One.World, environment comes as an abstraction for incorporating data abstractions which are noted as *tuple* and functions as a container of related user applications. Comparatively node level abstraction facilitates higher flexibility and energy economy, minimum communication and interpretation overheads for application development. System level abstractions are simpler to use. The node level behaviors can be produced automatically relieving programmer to concentrate on the network-level actions, without bothering on sensor nodes collaboration to perform the allocated tasks.

Development support

Middleware applications entail different programming interfaces matching the underlying system architecture and functionalities. Precisely aspect of developing programming abstractions is *interface* type. This refers to the approach of the application programming interface (API). Programming abstraction is embodied as the programming interface. Gaia provides a standard programming interface of the active space model for developers to program the active space as a single entity. Aura provides special interfaces for *suppliers* and *connectors*. PICO/SeSCo offers resource service abstraction through a graph model for adaptation of existing devices. One.World gives a common API for service discovery and communication library for programmers. CORTEX supports APIs to enable event communication and API for timely-computing based (TCB) model [4].In general, the system

service functionalities provided among common services include context management, service management, reliability & security management. Context management services are responsible for contextual data acquisition, processing, and derivation of higher-level contexts, context dissemination, context inconsistency detection and resolution. Service management is responsible for service discovery, service composition, and service handoff in middleware environment. Reliability and security management is responsible for ensuring correct functioning of the system. It caters to several hardware and software related faults and also ensures protection of sensitive user information. Also various runtime supports are essential for the underlying execution environment. The running context aware smart applications can be seen as an extension of the embedded operating system. It supports functions such as scheduling of tasks, inter-process communication (IPC), memory control, and power control. Middleware runtime support is mainly in assistance for local processing, communication, energy management, and storage. It employs multi-thread processing, smart task scheduling, and synchronization of memory access.

Context Management

Managing context refers with how the data flows from phases in context-aware applications especially where middleware are employed. The major focus is on where the data is generated and where the data is utilized. Context-awareness is no longer limited to web or mobile applications. It has already become a service namely Context-as-a-Service (CXaaS) which are web-based context management services. The classical context management system life cycle has main phases such as Context acquisition, Information processing, and Reasoning and Decision.

Context acquisition: Tobuildcontext from physical or virtual sensors for developingcontext-aware middleware solutions, *push* and *pull* techniques are used. These distinctions are based on responsibility of gathering information. In push the system requests for data from sources such as query to sensor mechanism periodically. On contrast to this in the Push method, information is send periodically to the system. The frequency of information can be at periodic intervals or instantly as per evens occurred. Sensors can be categorized further as physical, virtual (data gathered and processed before sending such as calendar, tweets etc), logical (web service to combine physical and logical sensor data) [11]

Context representation:

Setting new context, information needs to be defined in terms of attributes, characteristics and relationships. It has to consider the previously specified context, quality-of context attributes and the queries for synchronous context requests. Context is modeled based on heterogeneity, mobility, relationships, dependencies, freshness, imperfection, reasoning, usability of modeling formalisms, and effective context provisioning. The most popular context modeling techniques are namely key-value, markup schemes, graphical, object based, logic based and ontology based modeling.

Key-value modeling: models context information as key-value pairs in different formats such as text files and binary files. Simplest and easy to manage smaller amounts of data. Not scalable and do not suit for storing complex data structures.

Markup scheme modeling: models data using tags where context is stored using markup tags allows efficient data retrieval. Validation is supported through schema definitions using validation tools such as XML. Range checking is also possible up to some degree for numerical values. In contrast, do not allow reasoning. Due to lack of design specifications, context modeling, retrieval, interoperability, and re-usability over different markup schemes can be difficult.

Graphical modeling: models context with relationships. Some examples of this modeling technique are Unified Modeling Language (UML).

Object based modeling: object based modeling is suitable to be used as an internal, non-shared, code based, run-time context modeling, manipulation, and storage mechanism. Do not provide inbuilt reasoning capabilities. Validation of object oriented designs is also challenge due to the lack of standards and specifications.

Logic based modeling: rules are mainly used to express policies, constraints, and preferences. It provides much more expressive prosperity compared to the other models. Reasoning is possible up to a certain level. Lack of standardization reduces the re-usability and applicability.

Ontology based modeling: context is organized into ontology using semantic technologies. A number of various standards (RDF, RDFS, OWL) and reasoning capabilities are supported as per the requirement. Various development tools and reasoning engines are also available. However, context retrieval can be computationally rigorous and time consuming as the amount of data is increased.

Context reasoning decision models

Context reasoning is method of inferring new knowledge, and understanding better. It is based on the existing context as a process of giving high-level context deductions from a set of contexts. Reasoning has emerged due to imperfection (i.e. unknown, ambiguous, imprecise, or erroneous) and uncertainty characteristics of raw context. Reasoning performance can be measured using efficiency, soundness, completeness, and interoperability metrics.
There are ample numbers of context reasoning decision models available. Popular decision models are *decision tree*, *naive Bayes*, *hidden Markov models*, *support vector machines*, *k-nearest neighbor*, *artificial neural networks*, *Dempster-Shafer*, *ontology-based*, *rule-based*, *fuzzy reasoning* etc. These are originated and are employed in the fields of artificial intelligence and machine learning.
Generally context reasoning techniques are categorizedinto*supervised learning, unsupervised learning, rule based, fuzzy logic, ontological reasoning* and

*probabilistic reasoning.*Supervised learning aims to first collect training examples and labeled according to the expected results. Then a function is derived which can generate the expected results using the training data. This technique is widely used in mobile phone sensing and activity recognition. *Decision tree* is a supervised learning technique which builds a tree out of a dataset that can be used to classify data. Artificial neural networks technique's typical implementation is to modelcomplex relationships betweeninputs and outputs. Also it aims to obtain patterns in data. Pervasive healthcare monitoring system is best suit using body sensor networks domain experimentation. For context-aware reasoning clustering techniques such as K-Nearest Neighbor is prevalently employed. Precisely mentioning, clustering is used in low-level sensor network operations mainly for routing and indoor, outdoor positioning and location aware systems.

Discussion

Challenges in Context aware Middleware Implementations:

In developing social context-aware applications correct detection of user intention based on the situational knowledge requires further exploration. Conflict resolution among the data sensed by multiple sensor nodes is major apprehension for research. Storage of contextual data used in pervasive applications is also important issue. Reliability in service management operations for multi-application service provision is absolutely vital for context aware environments. In social context to analyze profiled user's personal and social behavior to predict future actions using context aware tools and technologies from social networks can entail major challenges, such as devising appropriate algorithms to match large scale data management for supporting data driven adaptability, managing user's privacy and security.

Acknowledgments

We express our sincere gratitude to all the authors for inspiring research attempts by virtue of their publications. We are thankful to all the authors whose papers are referred for preparing this article.

References

[1] Mark Weiser, "The Computer for the 21st Century", Scientific American Ubicomp, 1991.

[2] M. Satyanarayanan, Pervasive computing: vision and challenges, IEEE Personal Communications, 2001.

[3] "Context Aware Computing for the Internet of Things: A Survey", IEEE Communications Surveys & Tutorials, 2013.

[4] Vaskar Raychoudhurya, Jiannong Caob, Mohan Kumarc, Daqiang Zhangd ,"Middleware for pervasive computing: A survey", Pervasive and Mobile Computing,2013.

[5] Daniel Schuster et.al., "Pervasive Social Context: Taxonomy and Survey",ACM Transactions on Intelligent Systems and Technology, 2013.

[6] Christian Becker et.al.,"PCOM–A Component System for Pervasive Computing", IEEE Annual Conference on Pervasive Computing and Communications (PERCOM'04),2004.

[7] ManuelRomán et. al. , "A Middleware Infrastructure for Active Spaces",IEEE ,Pervasive computing,2002.

[8] Christian Becker et. al., " BASE - A Micro-broker-based Middleware ForPervasive Computing", IEEE International Conference on Pervasive Computing and Communications,2003.

[9] VerenaMajuntke et.al,"A Coordination Framework for Pervasive Applications in Multi-User Environments", IEEE Sixth International Conference on Intelligent Environments, 2010.

[10] J.P. Sousa, D. Garlan, "Aura: an architectural framework for user mobility in ubiquitous computing environments", Proc. of the 3rd Working IEEE/IFIP Conference on Software Architecture, 2002.

[11] D. Garlan, D.P. Siewiorek, A. Smailagic, P. Steenkiste, Project Aura: toward distraction-free pervasive computing, IEEE Pervasive Computing 2002.

[12] H. Truong, S. Dustdar, "A survey on context-aware web service systems", International Journal of Web Information Systems, 2009.

[13] VerenaMajuntke et.al. , "COMITY: Coordinated Application Adaptation in Multi-Platform Pervasive Systems", IEEE, (PerCom),2013.