Policy Research Journal
ISSN (E): 3006-7030 ISSN (P) : 3006-7022 Volume 3, Issue 6, 2025

ADDRESSING INCONSISTENCIES OF CONTEXT IN CONTEXT-AWARE APPLICATIONS

Asadullah Kehar[*1], Ghulam Ali Alias Atif Ali Memon[2], Shereen Fatima[3], Sohail Ahmed Memon[4], Israr Ahmed Memon[5], Mashooque Ali Mahar[1]

*[*1]Institute of Computer Science, Shah Abdul Latif University, Khairpur*
[2]COPELABS, Universidade Lusofona, Lisbon
[3]Department of Computer Science, Shaikh Ayaz University, Shikarpur
[4,5]Department of Mathematics, Shah Abdul Latif University, Khairpur

[*1]asad.kehar@salu.edu.pk, [2]ghullamali@gmail.com, [3]shereen.bhatti@saus.edu.pk, [4]167uhail.memon@salu.edu.pk, [5]israr.memon@salu.edu.pk, [1]mashooq.mahar@salu.edu.pk

DOI: https://doi.org/10.5281/zenodo.15610826

Keywords
inconsistent contexts; error recovery; context quality; context-awareness; inconsistency.

Article History
Received on 28 April 2025
Accepted on 28 May 2025
Published on 06 June 2025

Corresponding Author: * Asadullah Kehar

Abstract
The advent of computer has brought drastic changes in the technology utilization in a very little time. Traditional working mechanisms are fading out with the emergence of autonomic and ubiquitous systems. Context has been foundational stone in catering the day to day seamless environmental needs. Consideration of automatic adaptation is being understood and used in the routine application development paradigms. So the need to obtain good quality context has emerged with ever increasing existence of pervasive computing environments. Escape from such environments is impossible in this era of IOT (Internet of Things) where almost everything even time and location are treated as contexts. Bad quality contexts also known as inconsistent contexts are great threat to the sustainability and persistence of pervasive computing applications. Inconsistency is generated when duplicate sensing devices report variable outputs. Such improbable results can bring contextual application crashes extending monetary as well as error recovery loss. To address this issue, the concept of constraints / rules has been proposed for consistency checking and various methods for contextual correction are proposed. But still endeavours are needed to attain optimal context inconsistency resolutions. Our work is one of the efforts to mitigate this issue of context inconsistency resolution and to attain optimal context quality so that persistence and sustainability of context aware applications may be maintained.

INTRODUCTION

Context-aware computing, a technological phenomenon has emerged as revolution for information technology infrastructure through the integration of novel technologies into every aspect of human life. It includes a wider picture of contextual information, deducing sophisticated adaptions from intrinsic contexts. In this evolution, the mobile devices have played an important role by creating automated adaptable environments which try not to involve the human intelligence for decision making [1]. However, the popularity and spread of contextual applications has also given rise to several issues [2]. These issues are mainly related to context inconsistency, which often lead to application failures and losses. To deal such problems, we propose a new framework

for resolving context inconsistencies at the stage of context acquisition, especially by reducing the processing time. There are several applications of Context-aware computing in a variety of fields, this ranges from healthcare to industries, and its use has proven to lead to pervasive and ambient environments. The collaboration between this technology and the artificial intelligence has created new opportunities for scientific innovation with least user intervention. The pervasive computing is an exciting for both business and personal uses.

Context-aware computing can offer numerous benefits to industries across various sectors. Such inconsistencies can cause application failures, leading to losses in terms of both price and system functionality. To address this issue, various strategies have been proposed, but achieving 100% optimization remains a challenge. For example, in a smart hospital room, inconsistent blood pressure readings from sensors can be life-threatening for patients if wrong treatments are administered [4]. Therefore, it is crucial to identify and resolve context inconsistencies before deploying pervasive applications in real-life scenarios. The research aims to answer key questions, including how to achieve optimal context inconsistency resolution considering accuracy and cost, and what parameters increase proficiency in context-aware applications [5]. The findings will have practical implications for designing systems in production environments, departmental stores, hospitals, and other sensitive areas. Context is the essential building block of all pervasive computing applications, and Figure 1 illustrates examples of these contexts.

Figure – 1. Examples of various types of contexts.

1. Literature Review

In 1991, Mark Weiser introduced the concept of technology that seamlessly integrates into daily life, requiring minimal attention from users. He termed this concept "ubiquitous computing" or "pervasive computing," emphasizing that the most effective technologies are those that become an unnoticed part of our everyday lives. He believed this occurs not just because of technology, but because people become accustomed to things and stop thinking about them. Weiser discussed new equipment and models that allow us to use technology without much thought. This means tasks can be performed with technology without giving explicit instructions. Every operation becomes automatic in the background without requiring any human's attention.

Currently, the "context-aware computing" is the area of attention where technology becomes independent which does not require continuous guidance. There is an automatic adoption regarding time, weather, and common practices. It

does not need the constant instructions. Such technology are surely helpful in making our lives simple and allowing us to use it effortlessly without constant guidance.

Despite having many advantages, this technology can meet various issues, something like crashing or malfunctioning, particularly when it misinterprets a situation. These problems create some strange situations such as increased costs, reduced reliability, and decreased accuracy. The main purpose of investigating such issues it to help people in understating that how technology can be incorporated effectively in our daily lives and to identify corners for improvement.

In context-aware and ubiquitous computing, the "context" is considered important and is the focus. It was derived from Latin words meaning "to weave together," the context is more than just a profile. This is a process where a link is established between the experiences and surroundings of the individuals to understand the world. It is like weaving a narrative with elements around us [6].

To understand the management of context, it is essential to define a four-stage process of the context lifecycle. Information related a surrounding environment is called the context. On the first stage, there comes the Context Acquisition where the information is acquired. Secondly, there comes the stage of the Context Modelling, this phase is comprised of organization and structuring. The modelled context is moved to two more stages for further processing. The decision-making and problem-solving take place at the third stage called the Context Reasoning. Finally, the modelled context to distributed to other systems and this stage is called the Context Distribution [7].

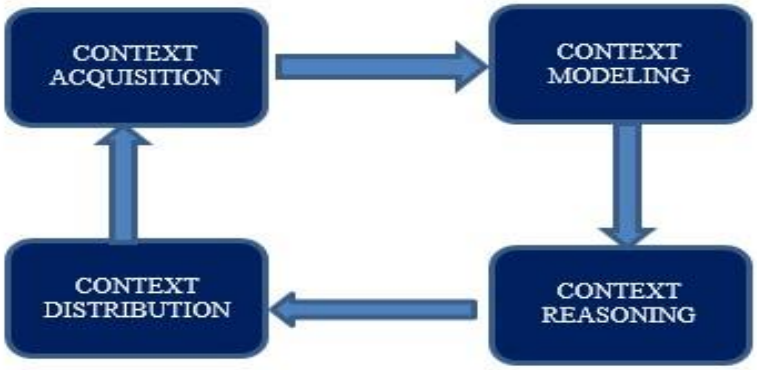


Figure – 2. Context Life Cycle

It has been observed the development of the applications based on context-aware can be complex, and the main issue is to determine the representation of an appropriate context. There are several methods for identifying the such appropriateness. Before learning and understanding such techniques, it is necessary to understand the functions of context representations. The Context representation is adapted for certain needs which involve combining different components, ensuring accurate and sufficient data, dealing any ambiguity and incompleteness, incorporating formality levels, and examining the compatibility with existing scenarios [8, 9].

The simple model is user-friendly and suitable for basic information which holds key-value pairs. Being simple, it lacks the ability to handle complicated data structures and is not optimized for fast data traversals. Mainly, such model is best used for preserving historical data, critical information, and configuration settings [10, 11].

In a technique, the tags are used for storing data [11, 12]. These tags are like containers which have the capability of holding other tags. The XML is mainly used for this purpose. Such model resembles to organizing physical objects with boxes. The examples of such profiles are the Composite Capabilities / Preference Profile (CC/PP).

It has been observed that the UML is utilized for visualizing models [13]. This approach is simple and suitable for context modelling. On the other hand, there are various tools such ORM, NoSQL databases, and XML which be incorporated for this purpose.

Object-oriented models can best be considered for the adoption of the real-world entities [14], these models are based on classes and objects and the properties like encapsulation. The technique of encapsulations enriches the organization and isolation of data and behavior but due to its adherence to certain rules, it increases the complexity of verification.

Thus, this modeling approach employs conditional logic to construct representational models. By establishing rules grounded in established knowledge and underpinned by logical principles, the framework is constructed. Within this model, context is encoded through factual components, sequential procedures, and logical precepts. The genesis of this modeling methodology can be attributed to McCarthy and his colleagues at Stanford [15, 16].

The adapted modeling approach focuses on representing spatial relationships and the environment. It incorporates elements such as sensors, locations, and entities. Spatial positioning is achieved using coordinates and elevation data. The model accommodates various levels of spatial granularity [17]. The modelling approach integrates diverse contextual representation methods to optimize performance. It combines various models to align with specific requirements. Illustrative examples include the integration of facts and ontologies, as well as the integration of tags and ontologies [18, 19]. The quality of contextual information is paramount for the effective operation of context-aware applications. A decline in context quality can lead to application malfunctions and subsequent system disruptions. The different techniques for tackling context awareness problems are illustrated in Table 1.

Table –1. Reasoning Approaches for Context Awareness [13]

Context	Reasoning Approaches
Availability (Busy, Free)	**Rules, Decision Tree**
Activity (Eating, Driving, Running etc.)	**Hidden Markov Models, Decision Tree, Bayesian Networks**
Environmental (Temperature, Humidity etc.)	**Direct from sensors**
Social Context (Who's nearby? What's their availability?)	**Logic, Rules**
High level identity (family member, neighbour, friend etc.)	**Social ontology, rules**
High level location (Coordinates, GPS etc.)	**Direct from device**
Mobility	**Hidden Markov Models, Dynamic Bayesian Networks**
Proximity (close, near, far etc.)	**Fuzzy logic**
Physiological Biometric (Blood Glucose, Oxygen saturation, Pulse etc)	**Direct from Bio Sensors**
Temporal (Time, Day, Year etc.)	**Direct from Time Sensors**

Context-aware computing empowers applications to dynamically adapt their behavior based on real-world environmental factors. This paradigm involves employing devices to capture contextual data, integrating it into the application, and leveraging it to inform decision-making processes. Applications can utilize this contextual information to execute diverse tasks and modify their operational modes accordingly. However, the accuracy of contextual information may be compromised by environmental noise or device malfunctions [20-23].

Due to this issue, the applications, which employ context, result in inaccuracies and can lead to erroneous application behavior. To address this challenge, researchers have proposed

methodologies for context data cleaning and refinement [24-27].

The comparison of inconsistency resolution techniques based on their strengths and weaknesses is presented in Table 2.

Table –2. Comparison by Pros and Cons

Method Name	Pros	Cons
Drop latest one	Simple and efficient	Insufficient may not give desirable resolution
Drop all	Simple and efficient	Deleted more contexts than required
User preferred	User defined and flexible	Quality of results rely on rules set by the user
Drop bad ones	Can support effective inconsistency resolution	Needs extra computing and judgements
Effect	Flexible and effective	It is expensive in terms that it needs to compare with all the solutions
Hybrid solution	Increased resolution accuracy can be achieved	Hard to decide the postponed time for resolution

2. Proposed Framework (Methodology).

3.1 Overview:

This section explores the significance of context in daily life and its application within context-aware computing applications. While context is indispensable for comprehending situations, it can also introduce inconsistencies. Resolving these inconsistencies is imperative for ensuring the reliability and accuracy of context-aware applications.

This study proposes a novel framework for enhancing the reliability and accuracy of context-aware applications through robust context acquisition and processing. The system comprises several interconnected components designed to collect, pre-process, and refine contextual information before integration into the application.

The flowchart presented in Figure 3 outlines a system for collecting, processing, and utilizing contextual information for context-aware applications. The flowchart in Figure 3 depicts the complete methodology and framework which emphasizes data quality and reliability through sensor redundancy and error correction mechanisms.

To develop a robust system that accurately captures, processes, and utilizes contextual information for context-aware applications by addressing potential issues like sensor errors and data inconsistencies. The subsequent text will elaborate on a framework comprising five essential components (as shown in Figure 4(a)) designed to effectively address and rectify context inconsistencies.

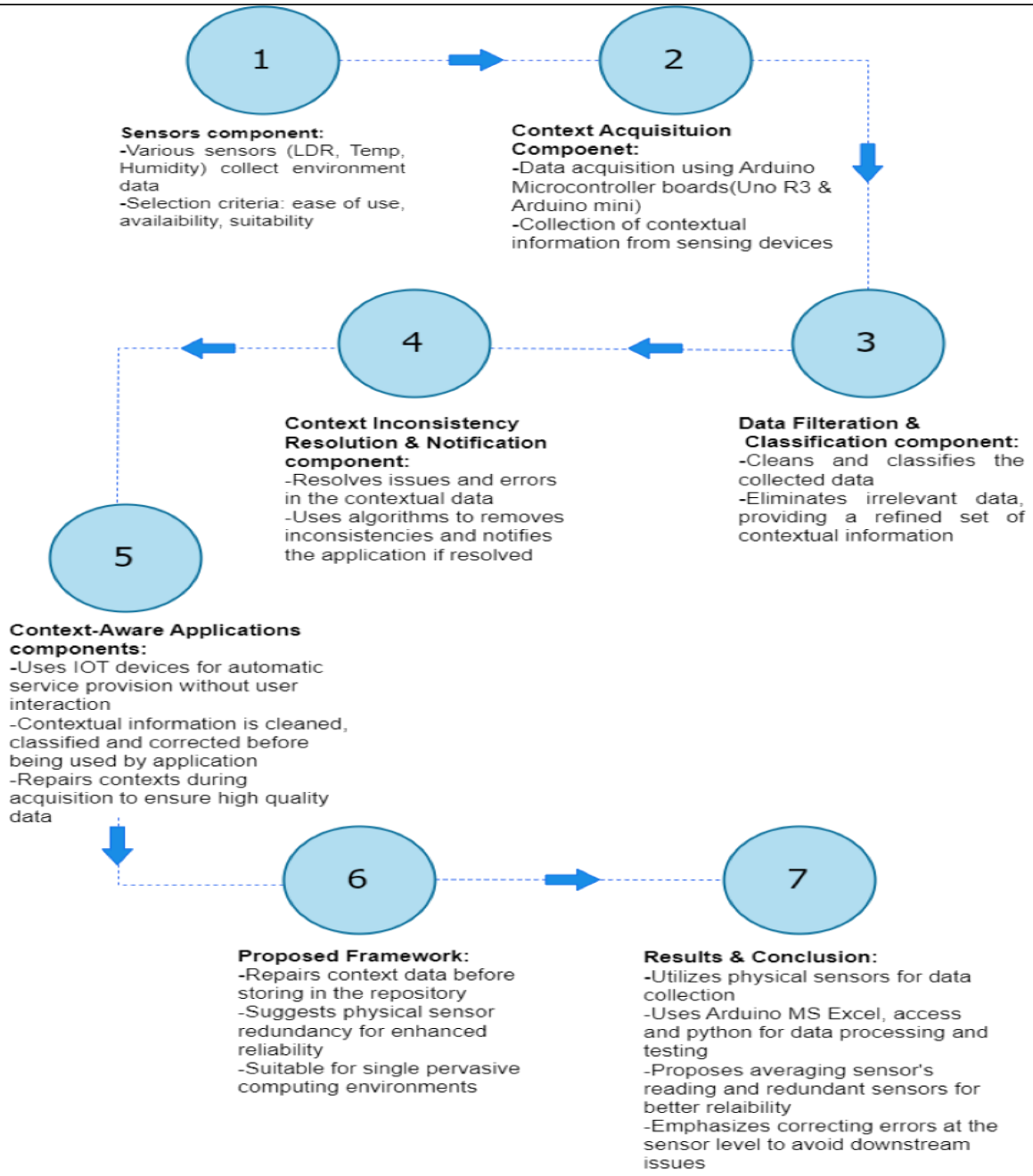


Figure –3: Framework for Contextual Information Acquisition, Processing, and Utilization in Context-Aware Applications.

3.2 Sensors Component:

This component addresses a component that provides information for a smart computing environment. This information is used for further processing. Various sensors are used to detect atmospheric conditions in automated systems. These sensors make it easy to collect and store atmospheric data in digital form. The researchers tested different sensors like light dependent resistor (LDR), temperature sensor, and humidity sensor. Three main factors are considered while choosing these sensors: (1) ease of use, (2) availability and accessibility, and (3) suitability for their specific research.

3.3 Context Acquisition Component:

The context acquisition component is an integral part of the Context Acquisition and Processing Manager, responsible for collecting contextual information. It aggregates data from sensory devices. In this study, Arduino microcontroller boards (Arduino Uno R3 and Arduino mini) were employed for data acquisition. The acquisition of contextual information is contingent upon this component.

3.4 DATA FILTRATION & CLASSIFICATION COMPONENT:

This component serves a critical role in pre-processing and classifying contextual information. By refining raw data and applying classification rules, it generates a condensed set of relevant context data and eliminates irrelevant data to obtain relevant context information.

3.5 CONTEXT INCONSISTENCY RESOLUTION AND NOTIFICATION COMPONENT:

This component performs two main tasks:

i. It takes the filtered and classified contextual data and fixes any issues or errors in the context.

ii. It identifies any inconsistencies in the context and uses a known algorithm to remove them before sending the corrected context to the context-aware application.

If unable to rectify inconsistencies, the module notifies the context-aware application of the discrepancy.

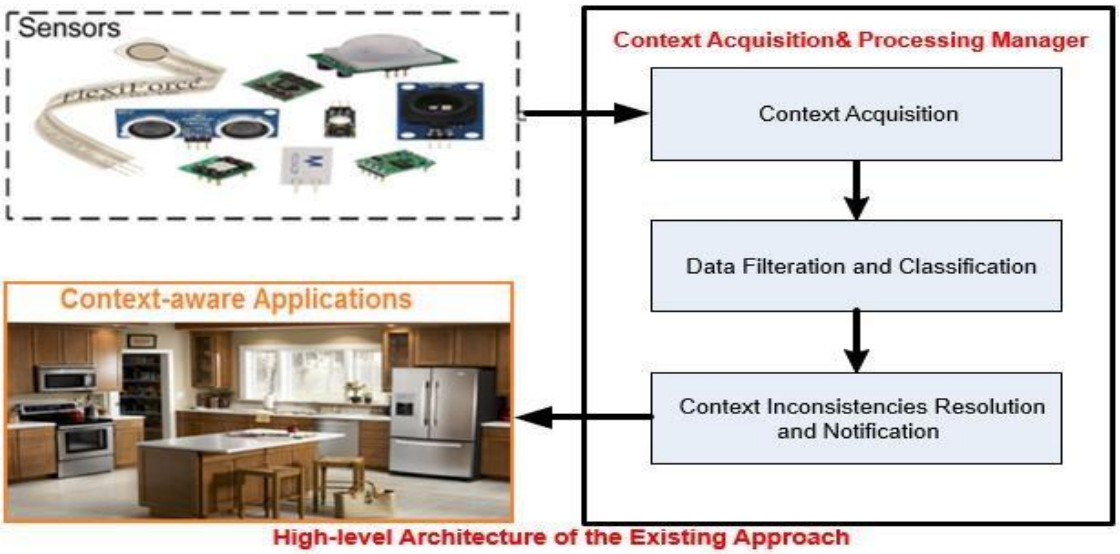


Figure –4 (a) Formal Framework for Context Aware Applications

3.6 CONTEXT-AWARE APPLICATIONS COMPONENT:

The context-aware applications that leverage IoT devices to provide programmed services without user intervention. Contextual information is pivotal to these applications, and various techniques are employed to acquire optimal data. This data undergoes cleaning and classification processes before being transmitted to the Context Inconsistency Resolution and Notification component. This component is tasked with repairing acquired contexts, detecting inconsistencies using predefined rules, and applying resolution strategies to rectify contexts prior to integration into the context-aware computing environment. Our methodology proposes a context restoration mechanism implemented during the context acquisition phase, as visually illustrated in Figure 4(b).

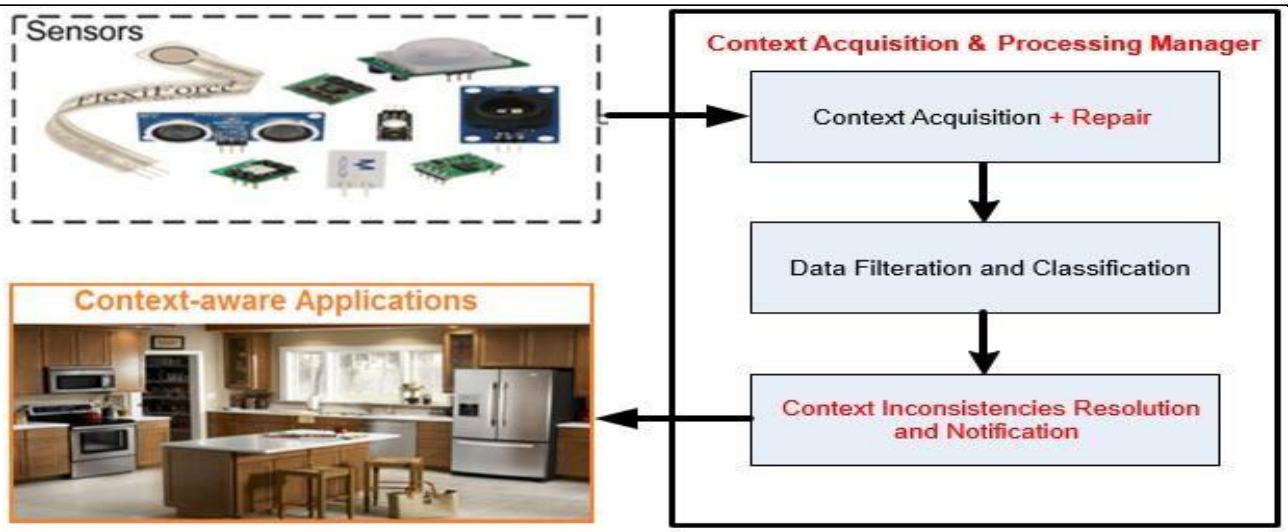


Figure –4 (b) High level Architecture of the proposed framework

The proposed approach enhances the performance of context-aware applications by expediting context repair during acquisition, as illustrated in Figure 4(c). This methodology advocates for context remediation prior to repository storage and suggests employing physical sensor redundancy (high availability) within the sensory environment. The efficacy of this approach is optimized within single pervasive computing environments.

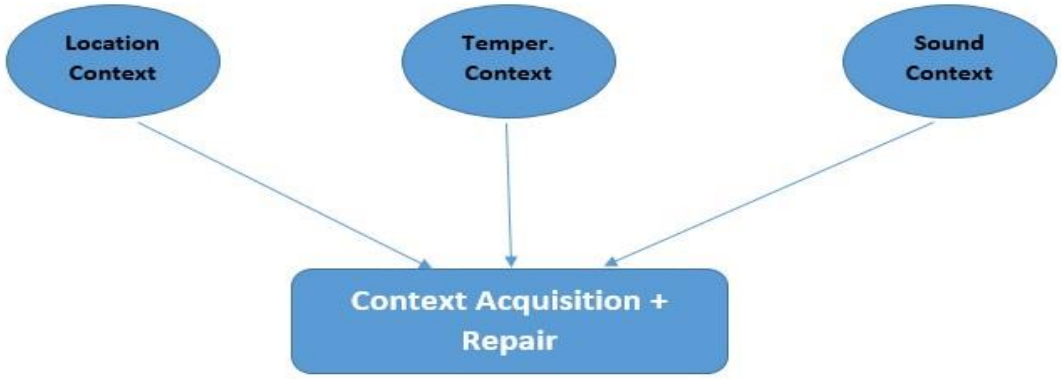


Figure –4 (c). Illustration of the working mechanism of the proposed system

3. Results and Conclusion

4.1 Hardware tools used:

Due to the inherent complexity and challenges associated with software simulators, particularly in terms of scenario validation and the requirement for specialized programming skills, the researchers opted for a physical sensor-based approach. The sophistication and lack of transparency in most context simulation tools further reinforced this decision. Consequently, the study employed physical sensors to generate contextual data and subsequently detailed the specific hardware components utilized for this purpose.

- **Arduino Uno R3 and Arduino Mini :**

Arduino is a tool for developing and constructing automated systems. It incorporates a microcontroller, a small computer, and software for coding and uploading programs. Arduino can be connected for communication with a computer via a USB cable. Renowned for its affordability and compatibility with various electronic modules

such as wireless, ultrasonic, and motion sensors, Arduino is a widely-utilized tool.

4.2 Software Tools Used:

A range of software tools were employed throughout the research process, encompassing Arduino software, Python, MS Access database, and spreadsheets. Arduino, utilizing a C-like programming language, was instrumental in developing concise programs, or "sketches," to capture data from the deployed sensors. The acquired data was initially collated within spreadsheets before being migrated to an MS Access database for subsequent retrieval and query operations. This facilitated the seamless transfer of data to subsequent processing stages. Python served as the unifying element, integrating the system's various components.

4.3 Working Mechanism and Results:

This section discusses the issue of minor variations in sensor readings that can lead to errors and inconsistencies in data. To mitigate this issue, the authors advocate for calculating and preserving the mean value derived from multiple sensor measurements. Additionally, they propose employing two distinct sensors sets at each designated location to enhance system robustness and facilitate maintenance procedures. In an experiment the authors collected a large amount of data and found that some readings had small variations, necessarily to be corrected during data collection. The authors emphasize the criticality of rectifying errors at the sensor level during data collection to prevent subsequent issues. The severity of error impact is contingent upon the specific application, with potential for severe repercussions in certain instances.

Repairs are done when sensors produce such minute variations in their output. For example: Sensor 1 (t1) is producing 17 degree Celsius and Sensor 2 (t2) is producing 17.2 degree Celsius, the difference causes inconsistency. In this situation, the context shall be adjusted and stored as a mean of both fields i.e. 17.1, hence repairing the context at the time of acquisition.

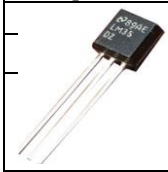
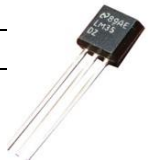
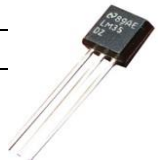
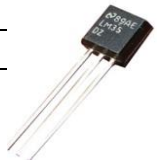
As illustrated in Figure 5, discrepancies in temperature readings were observed between sensors t1 and t2 during simultaneous data acquisition.

t1	t2
13:12:02.072 -> TEMPRATURE = 25.39*C	13:12:02.072 -> TEMPRATURE = 25.39*C
13:12:03.073 -> TEMPRATURE = 24.90*C	13:12:03.073 -> TEMPRATURE = 24.90*C
13:12:04.076 -> TEMPRATURE = 25.39*C	13:12:04.076 -> TEMPRATURE = 25.39*C
13:12:05.080 -> TEMPRATURE = 25.39*C	13:12:05.080 -> TEMPRATURE = 25.39*C
13:12:06.068 -> TEMPRATURE = 24.90*C	**13:12:06.068 -> TEMPRATURE = 24.94*C**
13:12:07.064 -> TEMPRATURE = 25.39*C	13:12:07.064 -> TEMPRATURE = 25.39*C
13:12:08.086 -> TEMPRATURE = 25.39*C	13:12:08.086 -> TEMPRATURE = 25.39*C
13:12:09.078 -> TEMPRATURE = 25.39*C	13:12:09.078 -> TEMPRATURE = 25.39*C
13:12:10.084 -> TEMPRATURE = 25.39*C	13:12:10.084 -> TEMPRATURE = 25.39*C
13:12:11.083 -> TEMPRATURE = 25.39*C	13:12:11.083 -> TEMPRATURE = 25.39*C
13:12:12.080 -> TEMPRATURE = 25.39*C	13:12:12.080 -> TEMPRATURE = 25.39*C
13:12:13.095 -> TEMPRATURE = 25.39*C	13:12:13.095 -> TEMPRATURE = 25.39*C
13:12:14.090 -> TEMPRATURE = 25.39*C	13:12:14.090 -> TEMPRATURE = 25.39*C
13:12:15.088 -> TEMPRATURE = 25.39*C	**13:12:15.088 -> TEMPRATURE = 25.41*C**
13:12:16.084 -> TEMPRATURE = 25.39*C	13:12:16.084 -> TEMPRATURE = 25.39*C
13:12:17.090 -> TEMPRATURE = 25.39*C	13:12:17.090 -> TEMPRATURE = 25.39*C
13:12:18.090 -> TEMPRATURE = 25.39*C	13:12:18.090 -> TEMPRATURE = 25.39*C
13:12:19.075 -> TEMPRATURE = 25.39*C	13:12:19.075 -> TEMPRATURE = 25.39*C

Figure 5: Real Time Temperature Context captured showing Anomalies

Furthermore, it is proposed that redundant sensor sets be deployed at each monitoring location. This strategy is economically viable given the relatively low cost of sensors compared to the potential financial consequences of application failure.

Table 3: Set of Sensors used for data acquisition

Temperature Set - 1		Temperature Set - 2	
			
T1	T2	T3	T4

If a sensor in Temperature Set – 1 or Set – 2 reports the same readings then the equal set readings shall be recorded in the context repository. Set of Sensors used for data acquisition are shown in Table – 3.

This approach gives us two benefits.

i It provides high availability with regards to context acquisition.

ii It provides ease with context repairs.

Additionally, we collected a total of 5,850 physical contexts at sequential intervals. Among these, 282 contexts exhibited minute reading variations, as illustrated in Figure 6.

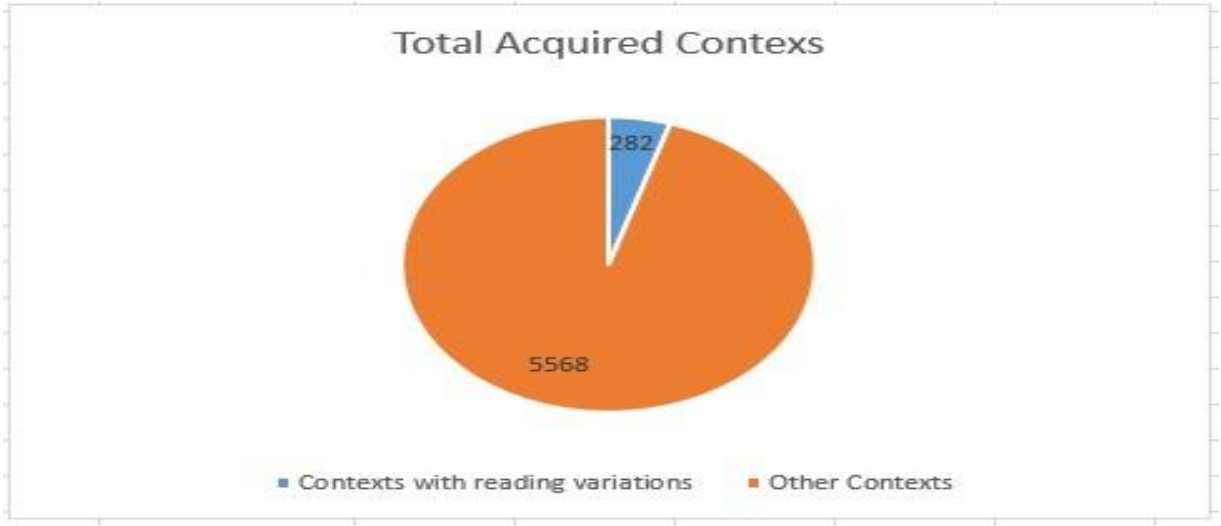


Figure 6: Total Acquired Contexts with Reading Variations

Such variations in sensor readings necessitate a repair mechanism to prevent the inconsistencies, which would otherwise require additional computational resources for context inconsistency resolution and notification. Furthermore, instances of sensor malfunction due to external factors, such as overheating, can lead to erroneous data. Our approach addresses these issues by implementing a sensor error correction mechanism at the context acquisition stage. Therefore, in our approach the sensor level errors are repaired during the context acquisition.

The cost of recovering from application errors and the overall system accuracy are contingent upon a multitude of factors and exhibit variability across different domains. In the realm of healthcare, context-aware applications carry a heightened risk, as system failures can potentially result in fatalities. Conversely, within warehouse environments, such failures may manifest as monetary losses, leading to partial or complete application disruptions.

4.4 Conclusion

This research provides a novel approach for resolving context inconsistencies in context-aware applications by introducing context repair at the time of context acquisition.

- By adjusting repair component in acquisition part, burden on inconsistency

resolution component is reduced enabling resolution component more effective.

- Also this modular approach reduces algorithmic complexity of the inconsistency resolution component, thus minimizes resolution time.

The future prospects consist of the following three aspects:

- Deploying and combining the suggested framework in large-scale, real-world pervasive applications.
- Performing testing and evaluation in real-life pervasive situations.
- Utilizing various statistical methods for the system's repair component.

REFERENCES

Ponce, Victor, and Bessam Abdulrazak. "Context-aware end-user development review." Applied Sciences 12.1 (2022): 479.

Kayes, A. S. M., et al. "A survey of context-aware access control mechanisms for cloud and fog networks: Taxonomy and open research issues." Sensors 20.9 (2020): 2464.

Tzevelekakis, Konstantinos, Zinovia Stefanidi, and George Margetis. "Real-time stress level feedback from raw ecg signals for personalised, context-aware applications using lightweight convolutional neural network architectures." Sensors 21.23 (2021): 7802.

Kavitha, D., and S. Ravikumar. "IOT and context-aware learning-based optimal neural network model for real-time health monitoring." Transactions on Emerging Telecommunications Technologies 32.1 (2021): e4132.

Xu, Jie, et al. "A new context correctness measure CMoC and corresponding context inconsistency elimination algorithm." Information Sciences 649 (2023): 119532.

C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca, "A dataoriented survey of context models," SIGMOD Rec., vol. 36, no. 4, pp. 19–26, 2007, doi: 10.1145/1361348.1361353.

D. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, "Context aware computing for the internet of things: A survey. Communications Surveys Tutorials," Ieee, vol. 16, no. 1, pp. 414 – 454, 2014.

T. Strang et al., "A Context Modeling Survey," Work. Adv. Context Model. Reason. Manag. UbiComp 2004 - Sixth Int. Conf. Ubiquitous Comput., vol. 16, no. 4, pp. 414–454, 2006, doi: 10.1.1.2.2060.

Sahito, M. Afzal, and Asadullah Kehar. "Dynamic content enabled microservice for business applications in distributed cloudlet cloud network." International Journal 9.7 (2021): 1035-1039.

O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-Aware Computing, Learning, and Big Data in Internet of Things: A Survey," IEEE Internet Things J., vol. 5, no. 1, pp. 1–27, 2018, doi: 10.1109/JIOT.2017.2773600.

Talpur, Mir Sajjad Hussain, et al. "Smart Talking Plant Based on IoT with Social Media."

C. Bettini et al., "A survey of context modelling and reasoning techniques," Pervasive Mob. Comput., vol. 6, no. 2, pp. 161–180, 2010, doi: 10.1016/j.pmcj.2009.06.002.

X. Li, M. Eckert, J. F. Martinez, and G. Rubio, "Context aware middleware architectures: Survey and challenges," Sensors (Switzerland), vol. 15, no. 8, pp. 20570–20607, 2015, doi: 10.3390/s150820570.

P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, "A survey of context data distribution for mobile ubiquitous systems," ACM Comput. Surv., vol. 44, no. 4, 2012, doi: 10.1145/2333112.2333119.

J. McCarthy, "Notes on formalizing context," Science (80-.)., vol. 13, pp. 555– 560, 1993, [Online]. Available: http://cogprints.org/418/.

J. McCarthy and S. Buvac, "Formalizing Context (Expanded Notes)," Theory Psychol., vol. 15, no. 1, pp. 128–131, 1997, [Online]. Available: http://cogprints.org/419/.

A. U. Frank, "Tiers of ontology and consistency constraints in geographical information systems," Int. J. Geogr. Inf. Sci., vol. 15, no. 7, pp. 667–678, 2001, doi: 10.1080/13658810110061144.

K. Henricksen, S. Livingstone, and J. Indulska, "Towards a hybrid approach to context modelling, reasoning and interoperation," Adv. Context Model. Reason. Manag., pp. 54–61, 2004, [Online]. Available: http://henricksen.id.au/publications/UbiCompWorkshop04.pdf.

A. Agostini, C. Bettini, and D. Riboni, "Hybrid reasoning in the CARE middleware for context awareness," Int. J. Web Eng. Technol., vol. 5, no. 1, pp. 3–23, 2009, doi: 10.1504/IJWET.2009.025011.

Kandhro, Irfan Ali, et al. "Detection of Real-Time Malicious Intrusions and Attacks in IoT Empowered Cybersecurity Infrastructures." IEEE Access 11 (2023): 9136-9148.

Y. Bu, T. Gu, X. Tao, J. Li, S. Chen, and J. Lu, "Managing Quality of Context in Pervasive Computing *," 2006.

I. Park, D. Lee, and S. J. Hyun, "A Dynamic Context-Conflict Management Scheme for Group-aware Ubiqui-tous Computing Environments," 2005. [Online]. Available: http://diana.icu.ac.kr/owl/service#.

S. Ryan, J. Minos, G. M. Franklin, S. R. Jeffery, M. Garofalakis, and M. J. Franklin, "Adaptive Cleaning for RFID Data Streams," 2006. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-29.html.

C. Xu and S. C. Cheung, "Inconsistency Detection and Resolution for ContextAware Middleware Support," 2005.

D. Heckmann, "Situation modeling and smart context retrieval with semantic web technology and conflict resolution," CEUR Workshop Proc., vol. 146, pp. 128–132, 2005.

Khoso, Fida Hussain, et al. "Proposing a novel iot framework by identifying security and privacy issues in fog cloud services network." Int. J 9.5 (2021): 592-596.

C. Xu, S. C. Cheung, W. K. Chan, and C. Ye, "Heuristics-based strategies for resolving context inconsistencies in pervasive computing applications," in Proceedings - The 28th International Conference on Distributed Computing Systems, ICDCS 2008, 2008, pp. 713–721, doi: 10.1109/ICDCS.2008

.