Software Engineering for Internet of Underwater Things to Analyze Oceanic Data

Abdul Razzaq*[a], Aakash Ahmad[b], Asad Waqar Malik[c], Mahdi Fehmideh[d], Rabie Ramadan[e]

[a] *Ocean Technology and Engineering Ocean College Zhejiang University Zhoushan Zhejiang 316021 China 11934071@zju.edu.cn - abdull.razzaq786@gmail.com*
[b] *Lancaster University Leipzig Germany. a.ahmad13@lancaster.ac.uk*
[c] *National University of Sciences and Technology Pakistan. asad.malik@seecs.edu.pk*
[d] *University of Southern Queensland Australia. Mahdi.Fahmideh@usq.edu.au*
[e] *Cairo University Giza Egypt — University of Ha'il Ha'il Saudi Arabia. Rabie@rabieramadan.org*

Abstract

Internet of Things (IoTs) represents a networked collection of heterogeneous sensors – enabling seamless integration between systems, humans, devices, etc. – to support pervasive computing for smart systems. IoTs unify *hardware* (embedded sensors), *software* (algorithms to manipulate sensors), and wireless *network* (protocols that transmit sensor data) to develop and operationalize a wide range of smart systems and services. The Internet of Underwater Things (IoUTs for short) is a specific genre of IoTs in which data about ocean ecosystems is continuously ingested via underwater sensors. IoUTs referred to as context-sensing eyes and ears under the sea operationalize a diverse range of scenarios ranging from exploring marine life to analyzing water pollution and mining oceanic data. This paper proposes a layered architecture that (i) ingests oceanic data as a sensing layer, (ii) computes the correlation between the data as an analytics layer, and (iii) visualizes data for human decision support via the interface layer. We unify the concepts of software engineering (SE) and IoTs to exploit software architecture, underlying algorithms, and tool support to develop and operationalize IoUTs. A case study-based approach is used to demonstrate the sensors' throughput, query response time, and algorithmic execution efficiency. We collected IoUT sensor data, involving 6 distinct sensors from two locations including the Arabian Sea, and the Red Sea for 60 days. Evaluation results indicate (i) sensors' throughput (daily average: 10000 - 20000 KB data transmission), (ii) query response time (under 30 milliseconds), (iii) and query execution performance (CPU utilization between 60 - 80%). The solution exploits SE principles and practices for pattern-based architecting and validation of emerging and next-generation IoUTs in the context of smart oceans.

Keywords: Internet of Things, Software Engineering, Ocean Mining, Data Analytics, Smart Systems

---

1. Introduction

Internet of Things (IoTs) is a sensor-driven platform and an enabling infrastructure that orchestrates heterogeneous things such as systems, services, devices, and humans that coordinate autonomically in smart systems and environments [1]. Increased adoption of IoTs at a global scale is pinpointed by a recent report by Statista indicating that at the end of the year 2022, there existed a total of 13.14 billion IoT-connected devices worldwide, and the number is expected to touch 29.42 billion by 2030 (i.e., more than doubled in less than ten years) [2]. Moreover, from a commercial perspective, increased adoption of IoTs by enterprises in the context of smart healthcare, intelligent transportation, industrial automation, etc. indicates that worldwide revenue from IoT applications, platforms, and services is expected to reach $750 billion by 2025 [3]. The rapid adoption of IoT technologies in smart systems can be attributed to portable devices that unify hardware (embedded sensors), software (applications that control sensors), and wireless networking (protocols connecting sensors) that enable things to collect, process, and share contextualized data [4]. Typical examples of IoT-driven contextualized data can be health analytics or crowd-sensed traffic congestion that can be collected by pervasive and context-sensitive sensors, manipulated by software applications, and distributed over wireless networks. Software-intensive systems and services in IoTs are the backbones for data-driven smart systems initiatives across the globe such as the ones adopted by the United States [5], Europe [6], and Asia [7].

The Internet of Underwater Things (IoUTs) is a specific genre of IoTs that is designed to operate in an oceanic environment, ingesting data from underwater sources, and transmitting it to off-shore servers for data-driven intelligence and human decision support [8]. IoUTs are considered as the ears and eyes in the deep blue (sea) that provide useful insights by operationalizing scenarios such as monitoring of marine life, measuring underwater pollution, and analyzing the correlation between various factors such as impacts of temperature and acidity on water [9]. While the blooming applications of IoT in different domains (e.g., smart homes, farms) have been extensively discussed [9, 10]. A recently published report on 'Smart Ocean Technologies' indicates that despite the expected revenues of IoTs and strategic benefits of IoUTs, such sensor-driven systems entail some critical challenges [10]. These challenges include but are not limited to resource poverty of sensors, stability of wireless networks, and performance of networked things along with data security and privacy, etc. which can hinder the trustworthiness and mass-scale adoption of such IoUTs. There is a need to synergize engineering knowledge and practices from other domains of IoTs such as smart home, intelligent transportation, pervasive healthcare, etc. that can be tailored and applied in the context of IoUTs for smart oceans [8].

Research context: Engineering software applications and services for IoT systems require a multitude of software development expertise in the context of programming context-sensitive sensors and devices to operationalize and manipulate the internet of things. Specifically, Software Engineering for IoTs (SE for IoTs in short) as an emerging discipline aims to apply the methods, principles, and practices of engineering software-intensive systems to design, develop, deploy, and evolve sensors and things-driven applications effectively and efficiently [11, 12]. From a system engineering point of view, hardware and/or networking novelties are vital, however; true potential for IoT systems in general and for IoUTs in a particular lies with software systems that contain data and logic to manipulate hardware devices for offering services to end-users [12]. For example, IoT sensors and devices that mine oceanic data – collecting data via underwater sensors – rely on underlying software that contains necessary algorithms and logic to compute the correlation between oceanic variables such as temperature (°C) and acidity of the water (pH) and their impacts on marine life [10]. A recent roadmap of SE for IoTs [13] organizes experimental evidence from developing IoT applications to highlight that engineering artifacts such as software architecture, patterns, frameworks, and tool support can empower engineers and developers to architect IoT-driven systems in an automated and efficient manner. However, employing SE-specific processes and practices in IoT-based systems requires understanding the constraints and limitations of sensor-driven devices and software services [11]. Specifically, an IoUT sensor that collects underwater temperature supports portable and context-sensitive computing, however; such pervasive systems inherit limitations relating to resource poverty (limited processor and battery) and network stability that needs to be compensated [9, 10]. Moreover, the volume and velocity of oceanic data collected by the sensors require efficient processing to ensure robust performance while computing insights from the collected data [10, 12]. While prior studies [12, 9, 14] have proposed analytic solutions for underwater data, semi-automated and decision-based support of oceanic data predictive analytics is yet unnoticed in the existing literature. In recent years, software engineering processes and architectures have been gaining much attention to address the issues pertaining to the development and operationalization of software-intensive IoT systems [11, 15].

Novelty and contributions: DeepBlu project aims to unify SE and IoTs to enable architecting, developing, and validating a sensors-based solution that continuously ingests multifaceted data from underwater and processes it to provide critical insights to end-users for human decision support. A high-level view of the proposed solution is illustrated in Figure 1 which highlights the application of various SE concepts that are applied to design and develop IoUTs with complementary tool support to automate system development. As in Figure 1, a layered software architecture pattern is applied [15] to help system developers maintain the separation of concerns, i.e., layering to organize different operational aspects at different layers of the system that also enables modularization for algorithmic specifications. Precisely, the layered pattern rooted in software architecture consists of three layers (i) a *sensing layer* having sensors that ingest underwater data (ii) an *analytics layer* that processes the sensed data (iii) an interface layer that presents data for human interpretation.

We have used the ISO/IEC-9126 model for software quality [16] for qualitative and criteria-based evaluation of the solution's functionality and quality to validate the solution. In addition to a case study-based demonstration, we measure and evaluate sensors throughput (i.e., stability), query response (i.e., performance), and algorithmic execution (i.e., efficiency) for the solution. The novelty and contributions of this research are:

- Application of software engineering principles and practices to architect, implement, and validate an IoT-driven solution that systemizes the development and operationalization of IoUTs to analyze oceanic data.

- Development of a layered software architecture that modularizes the

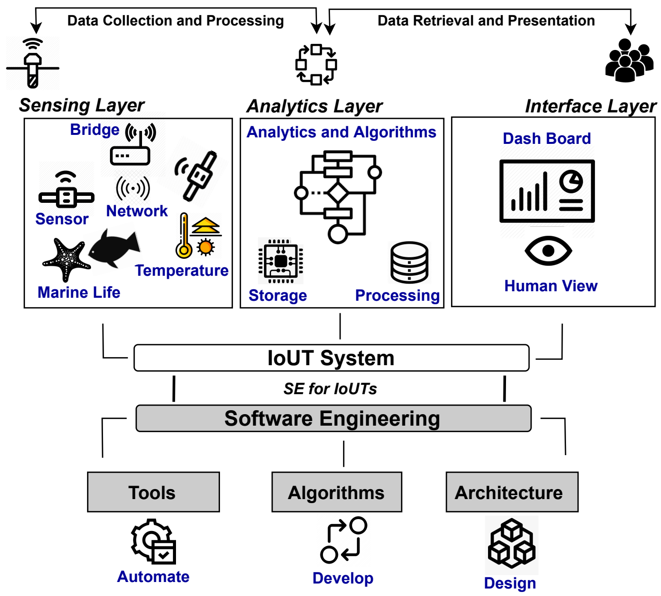

Figure 1: Overview of the Proposed Solution (SE for IoUTs)

solution, supports patterns as best practices of development and provides a set of algorithms that enable automation and parameterized customization of the solution.

- Validation of the solution based on a real-world case study that provides a scenario-driven approach to evaluate the quality of the solution in terms of sensors' throughput, query response, and algorithmic execution time.

The solution overview as in Figure 1 pinpoints architecting and developing IoUTs - synergizing SE practices with IoTs system development - as a specific genre of the IoTs [9, 11]. This research aims to provide a solution and set of guidelines that can help IoT researchers and practitioners engineer emerging and next-generation (software-intensive) IoT applications in the context of smart ocean systems.

Structure of the paper: This paper is organized as follows. Section 2 presents background details and related work. Section 3 presents the research methodology and architectural design. Section 4 details algorithmic details and solution implementation. Section 5 discusses case studies, evaluations, and validity threats. Section 6 concludes the paper and highlights the need for future research.

2. Background and Related Work

This section presents the background details to contextualize the building blocks of IoUT systems and elaborates on software engineering approaches to develop IoUTs (Section 2.1). We also review and compare the most relevant existing research to justify the scope and contributions of the proposed solution (Section 2.2). The concepts and terminologies introduced in this section are also used throughout the paper.

2.1. Smart Oceans and the Internet of Underwater Things

In the broader context of smart systems, the concept of smart oceans is a relatively new term and it represents a promising paradigm for research and development in areas including but not limited to maritime monitoring, oceanography, emergency search and rescue, and protection of marine life [17]. Developing tools and technologies that support smart ocean requires a synergy between pervasive systems and context-sensitive applications to sense, monitor, and identify underwater objects connected wirelessly to transmit oceanographic data. Traditional (land-based) IoT systems may lack capabilities in terms of sensors' configurations, their deployment, and software modules that have the capability to compute data ingested from the ocean [18]. Figure 2 conceptualizes the building blocks smart ocean in terms of data sensing, data analytics, and data presentation. To operationalize smart ocean systems, a number of wirelessly connected sensors from the *underwater things* are connected to a bridge node, i.e., a Scientific Instrument Interface Module (SIIM), for coordinating data to the backend server, expressed as $(\mathsf{S_A}, \mathsf{S_B}, \ldots, \mathsf{S_N})$ located at different locations. In addition to the oceanic data, each sensor [S] also adds information about the sensor's identity and location. The information is sent to the bridge so that it can be analyzed. In order to be able to aggregate all of the data from all sensors, the bridge acts as an intermediary between the sensors (i.e., a data collector and a data store) and the server (i.e., a data repository). Examples of oceanic data include a multitude of information such as types and levels of contamination, sunlight, temperature, dissolved oxygen, and water acidity. For example, assume the sensor having an identity $\mathsf{S_T}$ captures the value of temperature (°C) at specific time intervals and transmits the required information to the sensors' bridge. The bridge is responsible for collecting all sensor data and transmitting it to the backend server, where it is then processed. As in Figure 2, the server is managed as a cloudlet to send the data for offshore processing and storage. A large part of the offshore processing takes place in order to pull together the necessary data from various sources of oceanic information in order to analyze the correlation between various types of oceanic data such as the impact of temperature and acidity on marine life. Finally, the analytics results are presented to end-users for necessary actions and human decision support in the context of smart oceans.

2.1.1. Designing IoUT Systems

The software engineering standard (represented as ISO/IEC 12207:2008) provides a structured approach and process life-cycle to engineer software-intensive systems. Moreover, the architecture model (i.e., ISO/IEC/IEEE 42010:2011) provides a standardized approach to architect, develop, and evolve software services and applications effectively and efficiently [19]. The architectural models for IoTs are designed to abstract the complex implementation specific details (i.e., source code modules and procedural calls) with a high-level (component and connector) view of the system. Specifically, modules can be abstracted and represented as architectural components and architectural layers to conceptualize a system model [20]. There has been an increased focus on exploiting architectural models to design, develop, and evolve IoT systems in the context of smart homes, transportation, healthcare, and urban services [15]. As in Fig. 2, a simplified view of the architectural layers for IoUT is presented based on layered architecture pattern [12, 15].

Each later the concept of IoUT in terms of data and its computation. Sensor-based data sensing refers, for example, to the capture and representation of data that is obtained from underwater objects and sent to a server via sensors. The architectural modeling facilitates developers to design the system landscape while abstracting away from the implementation-centric and technical details (i.e., algorithm spec), which can be operationalized in later stages. For example, a recently proposed solution named ThingsML provides an architecture for high-level modeling of things in IoTs where the low-level executable specification can be generated in an automated manner using model-driven software engineering [20].

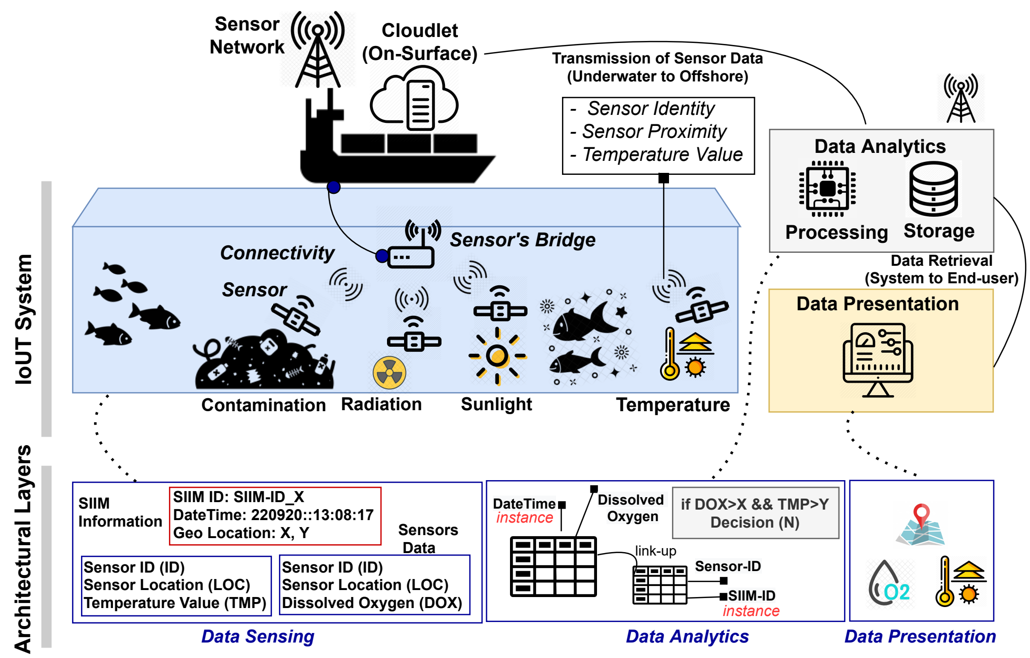

Figure 2: Building Blocks and Architecture Layers of IoUT for Smart Ocean Systems

2.1.2. Challenges for IoUT System Design

Despite the strategic benefits of IoUTs, architecting, operationalizing, and deploying underwater sensors remains a challenging task. The adoption of smart ocean technologies highlights that pervasive sensors entail several hardware, network, and software limitations [9, 10]. In terms of hardware, there is a lack of computation, storage, and energy resources, rooted in the pervasive and mobile nature of the sensors. The network instability leads to frequent disconnections and deteriorating sensor throughput can lead to anomalous data transmission. Further, from the software point of view, the performance of IoT data analytics in terms of algorithmic efficiency and query processing are among the primary challenges to be addressed [12, 13]. Notably, the adoption of software engineering life-cycle can enable architects and developers to (i) design IoUTs using patterns for an incremental and reusable development [12], (ii) develop parameterized algorithms to customize the solution [9], (iii) utilize software tools and technologies to automate the solution [20], and (iv) evaluate system functionality and quality based on standardized criteria (i.e., ISO/IEC-9126 model) for software validation [16].

2.2. Related Work

This section overviews the most relevant related work to analyze existing solutions, their underlying techniques, and limitations that justify the scope and contributions of the proposed solution. Table 1 acts as a structured catalogue to objectively compare and summarise the proposed solution in the context of the most relevant existing work.

2.2.1. Software-Intensive IoT-driven Systems

In recent years, many research and development initiatives have been put forward that advocate software engineering methods for the development of IoT-driven applications [11, 13]. Precisely, the initiative in [11] aims to organize IoT researchers' and practitioners' communities that can leverage academic research on software engineering for IoTs and its application and validation in an industrial context. Several similar efforts aim to establish the foundations that unify state-of-the-art software engineering principles with emerging and futuristic challenges of IoT systems [21, 22]. The study in [21] organizes key concepts and develops abstractions that revolve around the design and development of IoT systems to start shaping-up the guidelines of a new IoT-oriented software engineering discipline [20]. Some of the pioneering studies [11, 13, 20] laid the foundations for later work that goes beyond academic researchers to analyze practitioners' views and industry-specific processes for IoT systems. For example, an empirical study in [23] conducted a survey on IoT systems and practitioners from 35 countries across 6 continents with 15 different industry backgrounds. It can be considered a pioneering work on analyzing practitioners' views on key tasks, challenges, and software engineering methods for software-intensive IoT systems. In a similar work [12], the authors analyze multiple software engineering processes and practices that are used in industrial systems for IoT-driven data analytics. The study's results highlight the critical tasks, most relevant challenges, and recommended practices for developing IoT-driven systems for industrial analytics.

Usually, software architecture-centric techniques have been used to model and develop IoT applications. A recently conducted mapping study in [15] reviews qualitatively selected research studies to identify the challenges, architectural solutions, patterns, and areas of emerging research in software-defined IoTs. In a similar work, the authors have presented Things ML [20] as a model-driven, architecture-centric approach that empowers architects and developers via a model-driven software engineering approach to implement

IoT systems iteratively. The existing research and development on IoT-driven applications in [12, 15, 20, 23] complement a recently proposed roadmap for software engineering of IoTs that streamlines the essential aspects related to specification, design, and implementation of software-intensive IoT systems and applications [13].

2.2.2. IoT-driven Oceanic Data Mining

In the context of the smart system, there is a growing interest in tailoring IoT solutions for intelligent computing [1, 5, 7] (e.g., smart healthcare, intelligent transportation, home automation) that can be applied to sensor-driven mining of oceanic data [8, 9]. A new generation of satellites has provided oceanographers with a new means to acquire synoptic observations of ocean surface conditions at unprecedented time and space scales. This depends on the usage of satellites; see Halpern 2000 Satellites for more information [24]. Oceanographers have gathered information on critical parameters over time, including sea surface, temperature, worldwide high spatial, high accuracy, chlorophyll, and sea surface height (SSH). To allow a web-based platform for data collection, retrieval, interpretation, and visualization of oceanic data, Osen et al. [25] suggest a method to coordinate IoT data. The web-based framework has been introduced to provide distributed and interactive real-time data streams for detecting underwater oil detector resources. However, the underwater marine environment has increased the privacy and security issues of data collection, transmission, and retrieval. The authors in [14] have proposed an IoT-based architecture for a secure and efficient data compression algorithm to address these issues.

IoUTs are characterized as a worldwide network of intelligent, interconnected underwater objects that allow large unexplored water areas to be monitored [9, 10]. However, to process the collected oceanic data, such data must be transmitted to the storage infrastructure. In [26], the authors highlight some challenges related to storing and analyzing the data highlighting the fact that monitoring of ingested data from underwater sensors is still an open challenge for the research community. In [27, 28], authors have proposed a Deep learning-based approach called UIoT (underwater IoT) with an improved stability method to categorize acoustic sounds to automate marine sound processing in large data architectures [29]. The proposed UIoT architecture discusses the different scenarios to address critical challenges, including the increasing problems of long-distance underwater communication [9]. Some of the challenges mentioned above and alike constraints hamper the widespread adoption of IoUT systems and enable engineers and developers to architect emerging and futuristic challenges of ocean data mining. By deploying tens of thousands of inexpensive, intelligent floats that serve as a distributed sensor network over huge ocean regions, the Ocean-of-Things program [30], aims to close the marine knowledge gaps. In [31], the authors discussed maritime awareness and a cost-effective way of predicting ocean circulation and marine mammal tracking. The technology is based on consumer electronics off the shelf and involves a central controller with various sensors connected to it [32].

Critical Challenges for Oceanic Data Mining and Analytics – The design and implementation of efficient data mining algorithms represent one of the critical challenges due to the operational environment and coordination between data collected from IoT sensors and devices. However, to enhance system performance, marine organizations explore machine learning systems that can adapt to the complex environment [33]. Notably, data analytics techniques are mostly used to develop dynamic models through data interactions leveraging several layers of information received through IoT networks [34]. The analytics based on deep learning techniques are used for data extraction, transformation, classification, and pattern analysis [35, 36]. However, marine data poses unique challenges, including but not limited to the incompleteness of data, multi-sourced data ingestion, and complexity of analytics. The existing techniques focus primarily on quantifying data efficiently and reliably [37]. The recent studies explore problems such as infrastructure [38], storage [39], security [40], analysis [41], etc. to manage IoT data. Data mining approaches need further exploration to overcome challenges that include but are not limited to multi-source data streaming, complex marine data, performance enhancement [42], identifying trends from ambiguous data [43, 44], multi-source data mining algorithms [45], and advanced data mining and stream data processing methods [46].

2.2.3. Conclusive Summary

Table 1 provides a structured catalog to document criteria-based comparison of existing vs proposed solutions to present the scope and contributions of the proposed work objectively. We adopted the guidelines for classifying the existing research (IoT-driven data analytics and IoUTs [10, 12]) to shortlist five criteria that include (1) engineering method applied for IoT systems, along with the capability of the systems to exploit IoTs for (2) real-time data collection, (3) data analytics, (4) data mining and analytics, and

Table 1: Criteria-based Comparative Analysis of Existing Research vs Proposed Solution

Study Reference	**Engineering Method/Solution**	**Real-Time IoT Data Collection**	**Data Analytics**	**Forecasting Analytics**	**Mining Insights & Intelligence**	**Publication Year**
Waterston et al. [31]	✓	×	×	×	×	2019
Tziortzioti et al. [47]	×	✓	✓	×	×	2019
Hu et al. [14]	✓	✓	×	×	×	2020
Qiu et al. [9]	✓	×	×	×	×	2020
Ahmad et. Al. [12]	✓	×	✓	×	×	2021
Our Scheme	✓	✓	✓	✓	✓	

(5) predictive analytics. The year of publication is complementary information to highlight the years in which a particular solution was put forward. For example, the existing solution (Ahmad et. al. [12]) focuses on IoT-driven data analytics for industrial systems. The solution follows SE process life-cycle to develop and evaluate IoT systems that support real-time data collection and analysis. However, this solution lacks data mining and predictive analysis. Considering the overall comparison criteria in Table 1, we can conclude that a number of research studies have exploited SE for IoT-based data mining in the context of smart systems. However, there does not exist any solution that leverages SE methods and techniques (e.g., architecture, algorithms, patterns, tool support) for a systematic engineering and development of IoTs in the context of IoUTs. There is a need for solutions where software algorithms and applications can orchestrate the deployed sensors to ingest and analyze underwater data [25, 8, 9] to support software-defined IoTs. Technical details of the proposed solution are discussed in subsequent sections of this paper.

3. Research Method and Software Architecture

We now present research methodology (Section 3.1) that follows details of the software architecture for the proposed solution (Section 3.2).

3.1. Research Methodology

An overview of the research method is presented in Figure 3 that consists of four steps, following an incremental approach to analyze, design, implement and validate the solution, as detailed below.

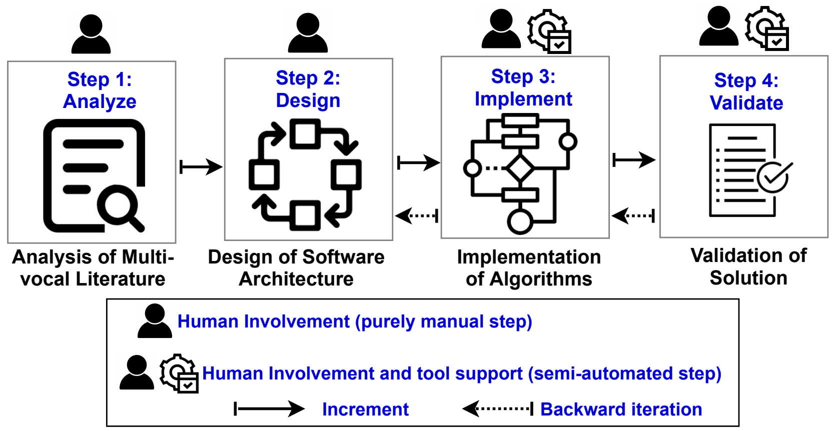

Figure 3: Overview of the Research Methodology

- **Step 1 – Analysis of Multi-vocal Literature** focuses on critical analysis of a diverse collection of existing literature (e.g., peer-reviewed published research, technology road maps, technical reports, etc.) [1, 9, 10, 11, 21] to pinpoint existing solutions and their limitations. We followed the guidelines to conduct the systematic literature review [48] to review the most relevant existing research (detailed in Section 2.2). Analysis of existing research and development solutions helped us to streamline the needed solutions and define the scope for this research (Table 1).

- **Step 2 – Design of Software Architecture** represents the design phase of methodology that aims to model the solution before its implementation. We followed the guidelines and recommendations to model IoT systems from [15] and adhered to ISO/IEC/IEEE 42010:2011 standard for architecting software systems to design the proposed solution [6]. A layered software architecture is developed that acts as a blueprint to implement the solution (detailed in Section 3).

- **Step 3 – Implementation of Algorithms** represents an implementation of the solution in the form of computation and storage-intensive steps. An algorithmic solution represents a modular decomposition of a solution that can be customized based on parameterized inputs by the users. Algorithmic details and underlying source code produce executable specifications for the architecture (detailed in 4).

- **Step 4 – Validation of Solution** is the last step that aims to evaluate the functionality and quality of the proposed solution. We have used the ISO/IEC-9126 [16] model to evaluate system quality. Specifically, we focus on measuring various aspects of system usability and efficiency based on a number of well-established evaluation metrics (detailed in 5).

  As in Figure 3, the initial two steps are purely manual activities requiring human intellect and decision support for their completion. In comparison, the last two steps involve human intervention and tool support to (semi-) automate the solution development. Iterations between steps (Step 4, 3, 2) may be required, in case there is any needed refinement(s) of the previous step. For example, Step 4, i.e., solution validation may suggest the refinement of algorithms to increase their efficiency or alter their functionality.

3.2. Architectural Representation for the Proposed Solution

We now present software architecture that represents a blueprint - comprising of the building blocks - for the overall solution. As discussed earlier, architecture for software-intensive systems is represented as an IEEE standard [19] that abstracts complex implementation-specific details of the system represented as architectural components (e.g., computational components or data stores) and connectors (i.e., component interconnections). The architectural view of the proposed solution is presented in Figure 4. Architectural specifications are presented independently of specific tools and implementation technologies to generalize the solution. Tools and technologies for architectural implementation are discussed later once the algorithms have been presented. The architecture model as in Figure 4 highlights:

- *Layered solution* that employs 3-layered architecture pattern [19, 21] to support the separation of functional concerns based on (1) sensing layer, (2) data analytics layer, and (3) data presentation layer, each detailed below. In the software development life-cycle, the separation of functional concerns, a.k.a. divide and conquer, allows architects and developers to engineer and develop a specific concern (e.g., data analytics or data presentation) in a parallel way.

- *Modularization of solution* represented as implemented algorithms, where each of the architectural layers can be represented as an individual module of the solution that supports customization based on user-specified input to the algorithms.

- *Architectural pattern* as reusable knowledge and recommended best practices provide a frequent design solution to recurring problems during the system development phase [15]. Pattern-based architecture enabled the reuse of design decisions, maintained the separation of concerns during development, and enhanced system extensibility and maintainability.

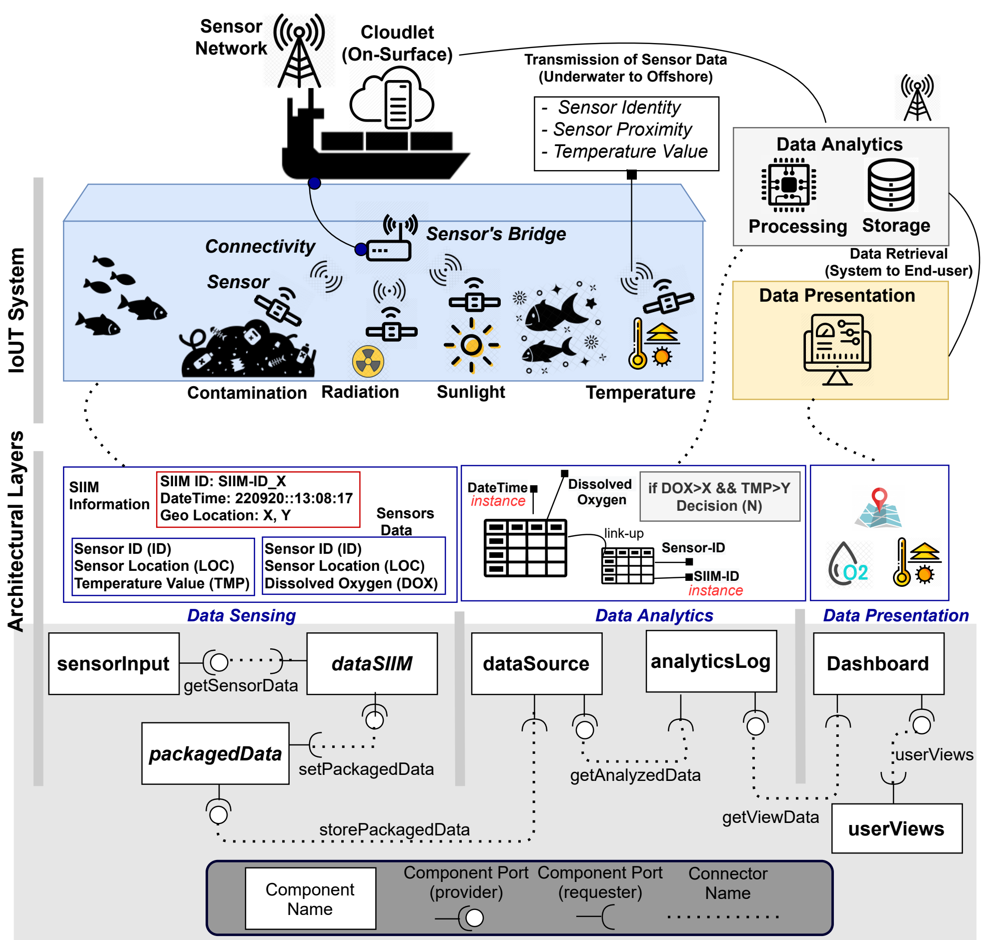

Figure 4: Overview of the Layered Software Architecture for the Solution

Figure 4 shows two views (a) domain view and (b) architecture view across three layers of the solution. Specifically, Figure 4 (a) domain view highlights a real-world representation of the system in terms of different functional aspects and building blocks. In contrast to the domain view, Figure 4 (b) highlights the system's component and connector-based architectural view of the system based on the UML component diagram [19]. Components of the architecture represent computation and storage-intensive units, whereas connectors represent the interconnection between components. For example, the component named packagedData gets accumulated sensor data from another component named dataSIIM using a connector storePackagedData at *Data Sensing* layer. In an architectural concern, layers only represent a logical separation of different functional aspects of the solution each of which is detailed below.

3.2.1. Layer 1 - Data Sensing

This layer deals with collecting data from the deployed sensors in the sea. Figure 4 illustrates a typical example of data sensing with deployed sensors (Sensor-ID) and their data collection. Each of the deployed sensors has a unique identifier referred to as Sensor-ID_A or Sensor-ID_B, shown in Figure 4, to collect different types of oceanic data, as highlighted in Table 2. Table 2 provides the kind of information that was gathered, the unit used to represent the data, and the specific sensor that was used to acquire the data. For instance, the sensor known as Sensor-ID_A gathers information known as DO (dissolved oxygen), which aids in measuring the amount of dissolved oxygen in the water. Scientific Instrument Interface Module (SIIM), which serves as a conduit between sensors (data collection) and data servers, supports sensor deployment and data collecting (data management). As a link between the two levels, SIIM unifies hardware and its control software to gather data from deployed sensors, package it with SIIM data, and send it to the server. For instance, in Figure 4 (a), the SIIM gathers data from Sensor A such as dissolved oxygen at a specific time (Dissolved Oxygen (DO): 7.93, DateTime: 22-09-20::13:05:37) and packages it with the SIIM's identity and the geolocation of the data collected (SIIM-ID X, GeoLocation). Periodically, based on a minute-based time interval, the data collecting and transmission procedure occurs. Figure 4 (b) illustrates how the packagedData component at the *Data Sensing* layer transmits data to the dataStore component at the *Data Analytics* layer via the storePackagedData connection.

3.2.2. Layer 2 - Data Analytics

This layer primarily focuses on managing and analyzing the data collected from the underwater sensors (i.e., data transmitted from SIIM). First of all, sensors' data packaged in a predefined format is stored in respective data stores. The data relating to sensors and SIIM identification (SIIM_X, Sensor_A, Sensor_B) is stored separately compared to other data such as geolocation, time-stamp, and DO. Furthermore, this layer supports data analytics such as DO oxygen patterns at a specific time, various sea levels, and their impacts on the acidity of water. Two main types of analytics are performed based on the types of data including (i) historical data determined by specific data collected between two-time intervals, and (ii) current data. Moreover, a correlation between two or more data items, as in Table 2, is computed such as the effects of temperature (°C) on the acidity of the water (pH) at a specific time. Analytics is performed through data processing and computations (further elaborated in the next section).

3.2.3. Layer 3 - Data Presentation

The final layer of the architecture presents key insights and results of data to the end-users. An end-user is an interested party who is interested in analyzing oceanic data, such as ocean explorers or marine scientists, etc. In this layer, which is also known as the user interface layer, a customized report is generated and various statistics are visualized in order to empower end-users (stakeholders and decision-makers) to make informed decisions. It is possible to see, as an example, over a specific period of time, the dissolved oxygen content, the underwater temperature, and the acidity of the water using the visualizations given here. At this layer (b), according to Figure 4, the architectural component named Dashboard can provide customised views to the users via the component named userViews.

4. Algorithms and Technologies for Solution Implementation

This section discusses the underlying algorithms and the technologies to modularise and implement the architecture-centric solution. By highlighting the tools and frameworks that are accessible to software and system developers, the topic of implementation technologies is introduced to support the algorithmic requirements.

Table 2: List of Data Collected by the Sensors

Data	**Unit of Measurement**	**Intent**
Temperature (°C)	°C Degree Celsius	To read the current temperature
Dissolved Oxygen (mg/L) (DO)	measure the amount of dissolved oxygen	To measure the dissolved oxygen in the water
pH (moles/L)	moles per liter	To determine the water's acidity
Salinity (ppt)	Parts per Thousand	to gauge how "salty" saltwater is
Turbidity (NTU)	Nephelometric Turbidity unit	To quantify the quantity of light dispersed by water's suspended solids.
Chlorophyll (mg/L)	mg chlorophyll per liter of water wavelength.	Chlorophyll levels in water are measured by the fluorometer.
Sea Level (m)	measurement in meters	To measure the depth.

4.1. Algorithms for IoUT-based Ocean Data Mining

Interpretation of the Algorithms: Figure 5 illustrates the algorithms' computational steps, data storage activities, and flow. The consistency between the proposed architecture (Figure 4) and algorithmic specifications (Figure 5) is maintained by mapping the architectural components with algorithmic steps across three layers. For example, the architectural component named *DataPackaging* inside the sensor layer in Figure 4 is mapped with the *Data Packaging* activity in Figure 5, implemented in Algorithm 1 - Send($\mathcal{F}, \mathcal{B}$) at Line 09. To facilitate customization and user input, the oceanic data items from Table 2 serve as parameterized input to algorithms. For instance, the sensor identification $\mathcal{S}$ is supplied as a parameter to Algorithm 1 Input: $\mathcal{S}$ at Line 01 in order to start data collection from that particular sensor. As shown in Figure 5, the accumulated data (Sensor ID, Sensor Data, Time and Location) from the sensor layer is packaged for its transmission to the data analytics layer, i.e., packagedData component in Figure 4. After performing the analytics, the data (Sensor ID, Location and List of Sensors, and Data Correlation from Multiple Sensors) is unified into the Analytics Log for its transmission to the interface layer, i.e., analyticsLog component in Figure 4. The inputs, processing, and outputs for each of the three layers of algorithms are presented in the remaining paragraphs of this section. Comments are made in an effort to clarify and make the text easier to understand by elaborating on certain algorithmic processes.

4.1.1. Algorithm 1: Sensors' Data Collection

This section explains the sensors' data collection mechanism as listed in Algorithm 1. In this algorithm, the data is packed in a specific format before forwarding to the server for processing. As stated before, the data collec-

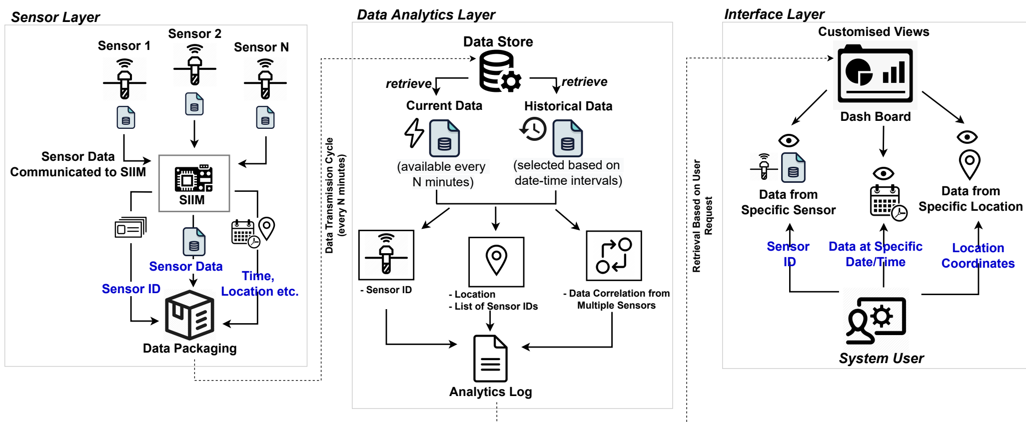

Figure 5: A Visual Overview of the Algorithms

tion and processing module contains a set of deployed sensors to perceive the environmental conditions and send their measurements. The SIIM is the controller module, responsible for data packaging and transmitting it wirelessly. It sends sensors' data to the backend server, where the data is processed. The data packaging is performed to consolidate a diverse set of data having ID, the value of oceanic data, sensor location, and date/time as a unified record for necessary analytics.

- *Input(s):* The input to the algorithm is the identity of a specific sensor being used to trigger the data collection (Input: $\mathcal{S}$ - Line 01).
- *Processing:* Data is ingested through sensors iteratively which is processed on a time slot basis, repeated frequently (Line 4, 5). However, four data categories are fed into the algorithm 1, sensor ID, SIIM ID, sensing value, and date time (Line 6). Initially, the data is buffered and later packaged (DPDATA_PACKEGE). The DPDATA_PACKEGE is further sent to the IoT server (Line 9) and the timer is reset (Time_Reset ()) to start a new interval to repeat the process (Line 8).
- *Output:* The output of the algorithm is the packaged data that has to be transmitted to the server (Output: $\mathcal{B}$ - Line 02)

4.1.2. Algorithm 2: Data Analytics

The data analytics module comprises multiple algorithms that include the Time Series, and Random Forecast Model applied to custom data sets. The

Algorithm 1 Data Sensing Algorithm

1: Input: $\mathcal{S}$ ▷ sensor data
2: Output: $\mathcal{B}$ ▷ data block on time interval
3: **procedure** DataPacking
4:     **while** true **do**
5:         $\mathcal{S}_i \leftarrow$ Read() ▷ read sensor data
6:         $\mathcal{B} \leftarrow$ AddBlk(ϕ_{id}, $\mathcal{S}_i$, t) ▷ develop data block
7:         **if** $t < t_p$ **then**
8:             $t \leftarrow$ Reset() ▷ Reset timer for next interval
9:             Send($\mathcal{F}$, $\mathcal{B}$)
10:             $\mathcal{B} = null$
11:         **end if**
12:     **end while**
13: **end procedure**

data mining algorithms and data storage is placed on-premises i.e. backend server. The data processing module is invoked on user-defined custom criteria in the data analytics part. The user can select one or more sensors to see the useful insights and make a correlation with other sensors to observe the impact of sensors. The output of sensors presents data from individual sensors or a correlation of data among more than one sensor, detailed below.

- *Input(s):* The algorithm takes five parameters as input (Input: σ, ψ, ϑ, ρ, $\mathcal{L}$) - Line 01). These parameters include a specific sensor, type of data sensed, date, time, and location of the sensor.

- *Processing:* The model is trained to provide insights and predictions using packaged data. It can be seen in Algorithm 2, the inputs are the Selected Type (selected_type), Sensor ID (s_id), Correlation Sensor IDs (co_ids), Location (location), and Date Range (date_range). The data is retrieved from the database server and processed according to the selected requirements based on the defined custom selection.

- *Output:* The output of the algorithm is the trained data model for data insights and predictions (Output: P_{set} - Line 02). The algorithm output is a set of values ($P_{PREDICTIONS}$).

4.1.3. Algorithm 3: User Interface

The user interface functionality is illustrated in algorithm 3 and specified in this section. The interface is used to highlight data insights based on

Algorithm 2 Data Analytics Algorithm

```
Input: $\sigma$, $\psi$, $\vartheta$, $\rho$, $\mathcal{L}$          ▷ sensor, data type, date, time, location
Output: $\mathcal{P}_{set}$                                          ▷ prediction set
procedure DataAnalytics($\psi$, $\sigma$, $\vartheta$=Null, $\rho$=Null)
    if $\psi$ == C || $\psi$ == H then                               ▷ Analytic on Streaming OR Historical data
        if $\sigma_l > 0$ then
            if $\mathcal{Q}.\sigma_l > 0$ then                       ▷ Correlation is not null
                if $\mathcal{L}$ != NULL then                        ▷ location is not null
                    while j < $\sigma_l$ do
                        while i < $\mathcal{Q}.\sigma_l$ do
                            $\mathcal{P} \leftarrow$ GetValue($\sigma_l$[j], $\mathcal{Q}$[i], $\mathcal{L}$, $\vartheta$, $\rho$)
                            if $\mathcal{P}$ != null then
                                $\mathcal{R} \leftarrow$ GetImpact($\mathcal{P}$)
                            end if
                            i++
                        end while
                        j++
                    end while
                else
                    while j < $\sigma_l$ do
                        while i < $\mathcal{Q}.\sigma_l$ do
                            $\mathcal{P} \leftarrow$ GetValue($\sigma$[j], $\mathcal{Q}$[i],$\vartheta$,$\rho$)
                            if $\mathcal{P}$ != null then
                                $\mathcal{R} \leftarrow$ GetImpact($\mathcal{P}$)
                            end if
                            i++
                        end while
                        j++
                    end while
                end if
            else
                while j < $\sigma_l$ do
                    $\mathcal{P} \leftarrow$ GetValue($\sigma$[j], $\mathcal{L}$, $\vartheta$,$\rho$)
                    j++
                end while
            end if
        end if
        return $\mathcal{P}$
```

given user input. The category of data insight includes *current data type*, and *historical data type* with a set of input variables (Selected Type, Sensor ID, Correlation of Sensor ID, Date & Time, Location.

Algorithm 3 Data Presentation Algorithm

1: Input: $\mathcal{U}$ ▷ user selection
2: Output: $\mathcal{R}$ ▷ Display analytics
3: **procedure** INTERFACEMODULE ▷ Event based function
4: $\quad\psi \leftarrow$ UserSelection()
5: $\quad$**if** $\psi == \mathcal{S}$ **then**
6: $\quad\quad\sigma \leftarrow$ Analytics($\mathcal{S}$) ▷ call analytical module on sensor type
7: $\quad$**end if**
8: $\quad$**if** $\psi == \mathcal{D}$ **then**
9: $\quad\quad\sigma \leftarrow$ Analytics($\mathcal{D}$) ▷ call analytical module on date specific
10: $\quad$**end if**
11: $\quad$**if** $\psi == \mathcal{L}$ **then**
12: $\quad\quad\sigma \leftarrow$ Analytics($\mathcal{L}$) ▷ call analytical module on location specific
13: $\quad$**end if**
14: $\quad\mathcal{R} \leftarrow$ UpdateDashboard(σ) ▷ Update analytics on user screen
15: **end procedure**=0

- *Input(s):* The input to the algorithm is used to retrieve the data based on the required data type selection (Input: $\mathcal{U}$ - Line 01).

- *Processing:* The analyzed data is stored on the server. The stored data is used for training the data model. This process is repeated frequently; however, for the incoming data, we trained the number of models in accordance with the number of correlation sensors. The output of the algorithm is data insights. Further, the variable data categories that are fed into algorithm 3 with Selected Type (Current Data, or Historical Data), Sensor ID, Correlation of Sensor ID, Date & Time, and location.

- *Output:* The output of the algorithm is the data insights and predictions that are to be transmitted to the user interface server (Output: $\mathcal{R}$ - Line 02)

4.2. Tools and Technologies for Algorithmic Implementation

This section summarizes the complementary role of relevant tools and technologies to implement the algorithms. The intent of the discussion here is to contextualize the tools and technology perspective used to implement the algorithms that realize the IoUT architecture. The tools and technologies are layered as in Figure 6. For example, the accumulated data at the sensor layer from SIIM, implemented as *Raspberry Pi*, is packaged as a *CSV* (Comma-separated values) file. The CSV file is transmitted to the server for analytics, where data is stored and managed using *MS-SQL* (Microsoft SQL) server. From a technical perspective, a direct SQL SERVER on-premises machine using Windows Operating System (OS) is utilized. A server-side application is developed using *.NET* platform (Visual Studio 2019) for user authentication. PyCharm IDE is used to perform server-side data analytics and Jupyter Notebook is for making the environment of data training. Python language is used to train the data model including these libraries (Numpy, Pandas, Matplotlib, Sklearn). Similarly, the user interface is implemented with a client-side scripting language like *JS* (Java Script). The data for the user interface layer is queried and managed via server-side scripting language C# (C-Sharp) and Python. As in Figure 6, the data packaging at the sensor layer and analytics log at the analytics layer are managed as CSV files that are processed and transmitted using C#

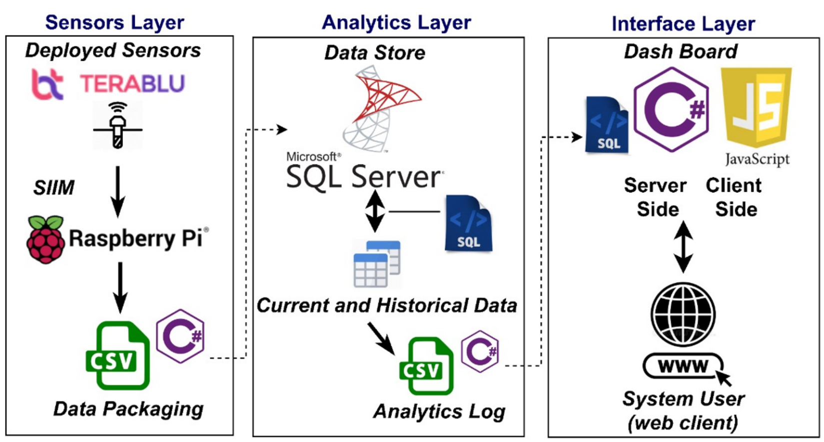

Figure 6: Abstract view of the proposed system with tools and technology interactions.

5. Evaluation of the Solution

First, we present the case study in Section 5.1 and then discuss the environment and evaluation dataset in Section 5.2. Afterward, we perform criteria-based evaluation by evaluating sensors' throughput (i.e., stability in Sections 5.3), query response (i.e., performance in Section 5.4), and algorithmic execution (i.e., efficiency in Section 5.5). The evaluation criteria are based on the ISO/IEC-9126 model used to evaluate software-intensive systems' quality [16].

5.1. Case Study

We now present the validation of the solution using a case study that is based on capturing ocean data for analysis from two distinct locations including (i) the Red sea, and (ii) the Arabian sea as shown in Figure 7. The case study is limited to data collection from two oceanic sites, however; as part of extending the basic proof-of-concept, in the future we plan for more diverse data to further validate the solution. Case study-based validation provides a practical context for scenario-based validation of the solution (see Figure7). Figure 7 illustrates a simplified view of the interface that allows the users to select three parameters (a) a specific sensor from the available list, (b) available locations for ocean data, and (c) correlation value/sensor. For example, the user selects the temperature and selects Arabian ocean and the pH value and the system shows the pH value of the data.

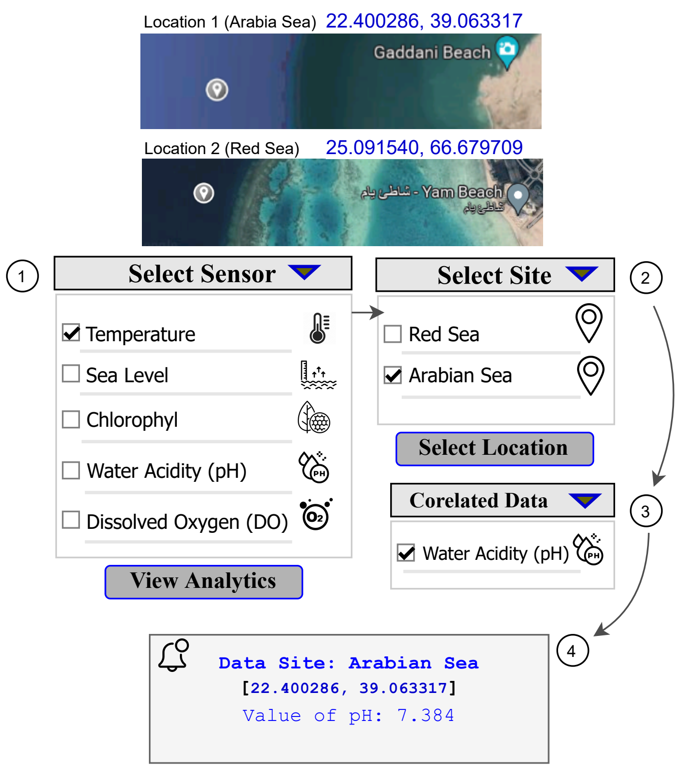

Figure 7: An Overview of Collected Data and Solution Interface

5.2. Evaluation Environment and Dataset

The evaluation environment refers to a collection of hardware and software resources used to execute the solution and measure various execution steps and outputs. Specifically, from the *hardware* perspective, evaluation experiments have been conducted using TeraBlu sensors, and Raspberry Pi (SIIM) with data analytics experiments performed on the Windows Platform (core i7 with 16 GB of runtime memory). From *software* perspective, execution evaluation also referred to as evaluation scripts, automates system testing. Such scripts have been written in Python and executed in Jupiter Notebook. A number of existing libraries, including but not limited to Matplotlib and Sklearn are also used during the evaluation process. For example, the script written in Python is used to measure CPU utilization while computing the correlation of data among various sensors, i.e., Algorithm 2, and evaluation results are visualized using Matplotlib.

The *dataset* used for evaluation consists of data collected from deployed sensors. The dataset includes a collection of data related to a list of sensors, current data (sensor throughput), historical data (query processing), and sensor correlation (system performance) along with the location and date/-time range of collected data. Further details about the dataset(s) being used are detailed in individual subsections below.

5.3. Evaluations of Sensors' Throughput

The evaluation, following are types of sensors used for the evaluation, temperature, dissolved oxygen (DO), pH, salinity, turbidity, chlorophyll, and sea level (as listed in Table 2). The temperature sensor is used to determine the ocean's climate; the DO sensor measures oxygen in the water, pH is used to determine the acidity of the water, salinity is used to determine the saltiness of the water; turbidity determines the amount of light in the water and sea level is used to determine the depth of the ocean.
At the data sensing layer, we evaluate the sensor's throughput to analyze the stability of data transmitted out of each sensor, as in Figure 8. Measuring the throughput can help identify if there are any disruptions or significant variations in the transmission over a period of time. Specifically, as per the plotting in Figure 8, the vertical axis represents the volume of transmitted data in Kilobytes (KBs), whereas the horizontal axis represents the time duration (number of days) for data collection. The throughput for each of the seven sensors is represented as an individual graph plot that moves along both axes such that fluctuation on the vertical axis represents data being

transmitted, while progression on the horizontal axis represents successive days. For evaluation purposes, the throughput is measured for a period of 60 consecutive days. Sensors send their collected data. The gateway at the sensing layer (i.e., SIIM in Figure 4) collected this data every 10 minutes and logged it into the server. Figure 8 highlights only the average of collected data per day. The results highlight the relative stability of the sensor's throughput with occasional fluctuations. For instance, the data about Sea Level goes up on day 15 and down on days 26, 39, and 47.

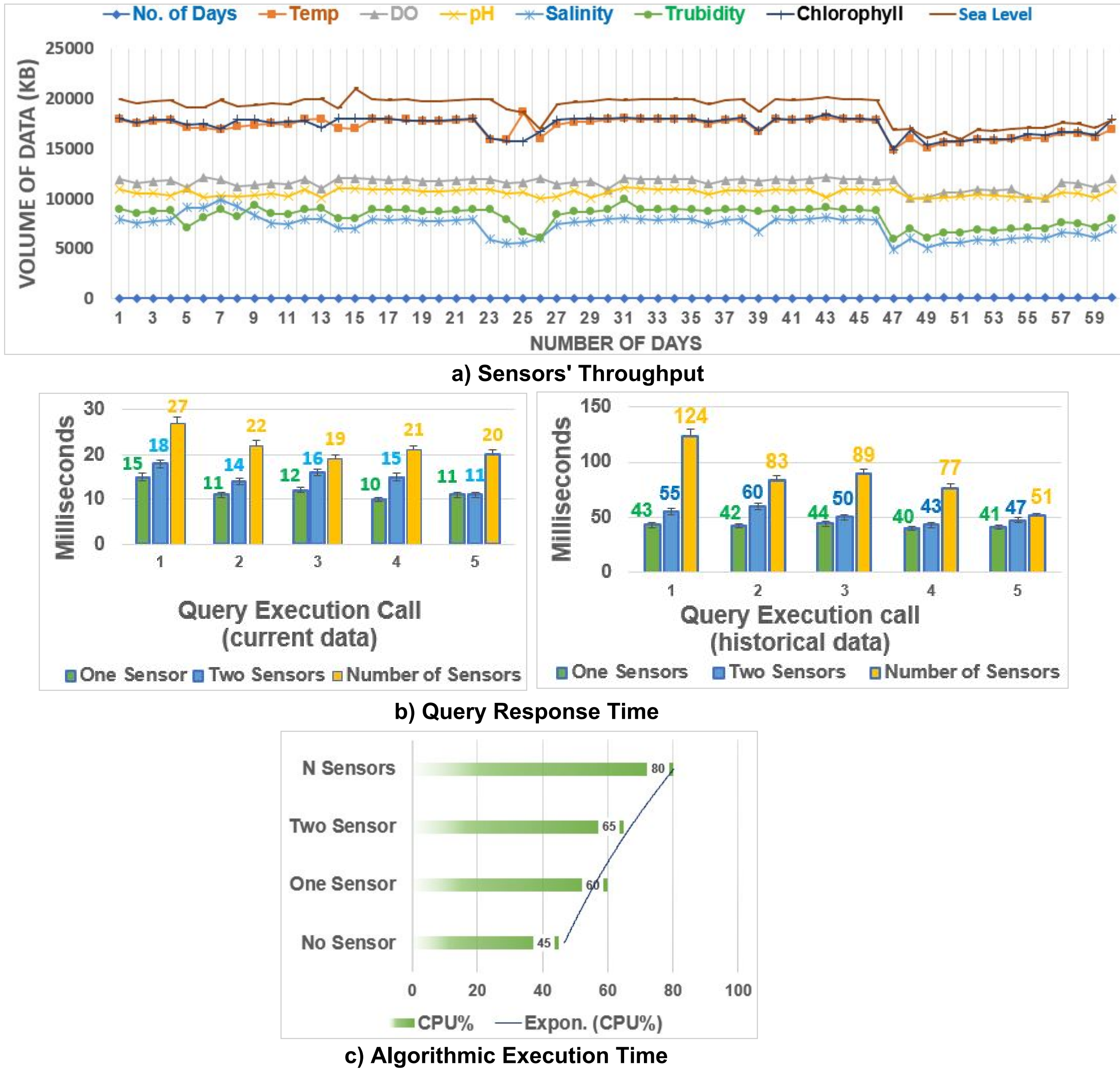

Figure 8: Overview of Evaluation Results

5.4. Evaluations of Query Response Time

Data querying is fundamental to retrieving oceanic data from the server for desired analytics. Evaluating the query response time helps to analyze the solution performance in terms of retrieving the recent and historical data from the server. Figure 8 highlights two views, i.e., (a) query response time for current data and (b) query response time for historical data. Specifically, Figure 8 represents average values for a total of 50 trials (average of 10 trials presented as 1 instance), where the vertical axis highlights the response time in milliseconds and the horizontal axis highlights the number of trials for 1, 2, N sensors respectively. For the purpose of analytics, historical data could be essential to be queried. Besides, data analytics could be done on a customized set of fields or ranges. Therefore, the query could differ based on the system requirements. For example, historical data could be divided further based on a date range and a date range time combination. As per Figure 8 (b), based on the number of trials, query response time increases due to the number of accumulated queries to the deployed sensors.

5.5. Evaluations of Algorithmic Execution

One of the critical measures, especially in IoT, is the algorithm's CPU usage. As shown in Figure 8(c), the CPU usage is computed for none of the sensors, one sensor, two sensors, and a number of sensors. The results indicate that in a normal case without executing the proposed algorithm, the CPU usage is 45% which means the board operating system and some other functionalities that are not related to the proposed algorithm are already consuming 45% of the CPU.

As the system becomes more advanced with the addition of multiple sensors, the CPU usage increases significantly. The increase CPU processing demands is due to a number of factors, including the need for data processing and communication protocols. These protocols are necessary to keep track of each sensor's state and activate the buffer for incoming data. Unfortunately, this increase in processing demands can reveal limitations in the system's ability to scale. The existing board may not have sufficient computing power to handle a large number of sensors, which could restrict the system's scalability. As a result, it's important to carefully consider the number and type of sensors to be added to the system and to ensure that the CPU is capable of handling the increased demands.

Key-points for Evaluation

Sensors' Throughput is evaluated to assess the stability of data ingested and transmitted from the deployed sensors. The results show that the sensors' readings are relatively constant with the other parameters, such as temperature, pH, or sea level. The average data transmission day remains at 20056 KB per day.

Query Response Time is used to understand the time it takes for data retrieval from the deployed sensors. The maximum time required for a query on the current data for one sensor is 15ms, while the maximum query time for two sensors is 18ms. At the same time, the maximum query time for a number of sensors is 27ms. Similarly, the query time in the historical data doe one sensor, 2 sensors, and a number of sensors are 43ms, 60ms, and 124ms, respectively. Our conclusion is that the query execution time is reasonable in both current and historical data cases.

Algorithmic Execution Time shows the performance of computation based on CPU utilization. The results of the evaluation highlight that With a single sensor, the CPU utilization reached 60%, while in the case of two sensors and a number of sensors, the percentage reached 65% and 80%, respectively. This indicates that the proposed algorithm takes at max 35% of the CPU time, on average.

Summary of Comparison: Table 3 complements the results of evaluation in Figure 8 with a brief comparative analysis of the most relevant algorithmic solutions on IoT-driven data analytics in smart oceans context. The comparative analysis is based on four main points that include (i) criteria for evaluation as the main objectives of solution validation, (ii) total numbers of IoT sensors for data collection, (iii) use case as the scenarios of evaluation, and (iv) environment being used for evaluation. For example, the algorithmic solution in [1] aims to evaluate energy efficiency and data transmission rate for underwater sensors. The number of sensors is not explicitly mentioned. The use case is digital twins in smart cities and more specifically smart oceans context. The algorithmic evaluation is performed using MATLAB simulations. In conclusion, the majority of existing algorithmic solutions have concentrated on using machine learning models for predicting data in specific applications. These solutions which are indicated in Table 3 on datasets assuming that the data is obtained from sensors.

Table 3: Comparison Summary of the Algorithmic Solutions for IoT-driven Data Analytics

Algorithmic Solutions	**Criteria for Evaluation**	**Number of IoT Sensors**	**Usecases of IoT Analytics**	**Evaluation Environment**
Yang et al. [49] 2020	- Computation Efficiency - Temperature - Accuracy	Not available	Predict Ocean blind zone information	Custom developed
Deng et al. [50] 2020	Number of barrier paths	1	Ocean border environmental surveillance	MATLAB
Hu et al. [51] 2020	- Computation Efficiency - Data Size	5	Secure, efficient, collection, transmission, and data storage for IoT in smart ocean	Python Simulations
Wang et al. [52] 2021	- Gliding motion elucidates, - Locomotion	3	Under-water fish bot	Custom developed
Li et al. [53] 2022	- Energy efficiency, - Data transmission rate	Not available	Smart city and Digital Twin	MATLAB
Proposed Solution	- Sensors Throughput - Query Response Time - Processor Utilisation	6	Underwater Data Analytics	Custom developed

In contrast, our work is primarily focused on not only data collection but analyzing useful insights of the data ingested by 6 deployed sensors. However, collecting data through sensors with limitations presents a significant challenge. In our work, we propose a framework to benchmark underwater data collection from various sensors. This approach enables us to address the challenge of data collection before preparing a predictive model that is mostly lacking in solutions on ocean data analytics. Based on the results of the evaluation for the proposed solution, we conclude that:

- Based on the amount of data collected consecutively, 60 days from 6 sensors, the sensors' throughput indicates consistency, completeness, and validity of IoUT data.
- The query execution time for one, two, and a number of sensors for both current and historical data are reasonable for IoUT-driven data analytics.
- The algorithm execution and CPU utilization are relatively stable, i.e.,

consuming only 35% of the CPU, on average, in case a number of sensors are used. In the case of one and two sensors, only 20% of the CPU time is needed. Therefore, the proposed system and algorithms are evaluated as computationally efficient to operationalize the IoUT systems. The proposed system and algorithms are valid to be used and implemented in other systems as well.

The query response time and CPU utilization are proportional to the number of sensors. As part of future work, i.e., incorporating additional sensors would result in increased query response time and CPU usage.

5.6. Threats to Validity

We briefly mention some potential threats to the validity of this research. Validity threats refer to some limitations or constraints that impact the solution's design, implementation, and validation. Validity threats need to be minimized as part of future work to optimize the solution and its implications.

- *Threats to Internal Validity:* relate to any restrictions or limits that could have an effect on the development and use of the suggested solution. For instance, the number of sensors used to gather oceanographic data, the number of trials needed to assess sensor correlation, and the platform used to assess the solution may all contribute to internal validity (see Section 5). This means that calculating the correlation between more sensors, increasing the number and magnitude of trials, or using different platforms can produce different results in the evaluation. Future work needs to consider the diversity of sensors, significantly large datasets and validation on different platforms can help with minimizing the internal validity.

- *External Validity:* It relates to the validation of solution on different related systems and case studies. As in the research method (see Figure 3) and evaluation (Section 5), we have adopted a case study-based approach to demonstrate and validate the solution. However, a single case study may limit rationalizing the generalization of the solution and the rigor of its validation. Future work is required to accommodate more case studies and different systems to minimize the impacts of external validity.

6. Conclusions and Vision for Future Work

Internet of Things (IoT) enable context-sensitive and pervasive computations that are fundamental to operationalizing smart systems that range from smart healthcare to smart transportation and smart ocean technologies. IoUTs as a specific genre of traditional IoTs unify data sensors, wireless networking technologies, and software applications to ingest, process, and analyze oceanographic data. Considering SE for IoTs, in this research we presented our solution on architecting a software-intensive IoUT system that advocates for pattern-driven reuse and algorithmic modularisation of the solution. Specifically, as part of the DeepBlu project, our approach synergized the concepts of software engineering (SE) and IoTs to leverage software architecture, underlying algorithms, and tool support in the development and operation of IoUTs. To evaluate the solution, we used a case study to deploy IoUT sensors for collecting data from two locations including the Arabian Sea, and the Red Sea. Evaluation results indicated sensors' throughput, query response time, and query execution performance to demonstrate the efficacy and efficiency of the solution.

Primary Contributions and implications of the research: We outline the primary contributions of this research as:

- Synergising the methods of SE and application of IoT systems to architect, develop, and validate an IoUT solution for analyzing oceanic data.
- Demonstrating the role of software architecture that enables algorithmic modularisation and case study-based validation of IoT systems.

The research and its finding can inform academic researchers and practitioners to architect and implement IoUTs for smart ocean systems. Specifically, academic researchers can further explore software engineering methods to research and develop emerging solutions based on IoT systems and technologies. Practitioners can explore the algorithmic-based and tool-supported approach to develop IoTs for smart oceans.

Needs for future research: The future research mainly focuses on extending the scope of the proposed solution in terms of (i) incorporating more sensors and diverse case studies, and (ii) mining patterns for the data collected using the proposed IoUT solution.

We aim to incorporate more sensors to gather enriched data that also involves analyzing the concentration of marine life in a specific oceanic zone. Also, there is a need for incorporating more use cases and diverse case studies that can help us gather data from different oceanic locations. The use of additional case studies to evaluate the solution can help us to understand if the proposed architecture and implemented solution in terms of data sensing and data analytics can be scaled up based on the increased size of data. We also plan to extend the solution that can discover patterns in sensed data that could improve predictive analytics. Pattern mining in sensor-ingested oceanic data requires us to extend the current architecture and implement a solution that can discover recurring sequences (analytic layer) as potential patterns to improve human decision support (interface layer).

References

[1] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, M. Guizani, Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges, IEEE Wireless Communications 23 (2016) 10–16.

[2] Statista, Iot and non-iot connections worldwide 2010-2025, 2021. URL: https://tinyurl.com/Statista2021.

[3] GSMA, Iot revenue: State of the market 2020, 2021. URL: https://www.gsma.com/iot/resources/gsmai-iot-revenue-2020/.

[4] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer networks 54 (2010) 2787–2805.

[5] S. America, Internet of things and system control, 2021. URL: https://smartamerica.org/about/.

[6] C. Manville, G. Cochrane, J. Cave, J. Millard, J. K. Pederson, R. K. Thaarup, A. Liebe, M. Wissner, R. Massink, B. Kotterink, Mapping smart cities in the eu (2014).

[7] J.-H. Lee, M. G. Hancock, Toward a framework for smart cities: A comparison of seoul, san francisco and amsterdam, Research Paper, Yonsei University and Stanford University (2012).

[8] C.-C. Kao, Y.-S. Lin, G.-D. Wu, C.-J. Huang, A comprehensive study on the internet of underwater things: applications, challenges, and channel models, Sensors 17 (2017) 1477.

[9] T. Qiu, Z. Zhao, T. Zhang, C. Chen, C. P. Chen, Underwater internet of things in smart ocean: System architecture and open issues, IEEE Transactions on Industrial Informatics 16 (2019) 4297–4307.

[10] Fraunhofer, Smart ocean technologies solutions for responsible ocean use, 2021. URL: https://tinyurl.com/Fraunhofer-Pdf.

[11] X. Larrucea, A. Combelles, J. Favaro, K. Taneja, Software engineering for the internet of things, IEEE Software 34 (2017) 24–28.

[12] A. Ahmad, M. Fahmideh, A. B. Altamimi, I. Katib, A. Albeshri, A. Alreshidi, A. A. Alanazi, R. Mehmood, Software engineering for iot-driven data analytics applications, IEEE Access (2021) 1–1. doi:10.1109/ACCESS.2021.3065528.

[13] R. Motta, K. Oliveira, G. Travassos, Iot roadmap: Support for internet of things software systems engineering, arXiv preprint arXiv:2103.04969 (2021).

[14] C. Hu, Y. Pu, F. Yang, R. Zhao, A. Alrawais, T. Xiang, Secure and efficient data collection and storage of iot in smart ocean, IEEE Internet of Things Journal 7 (2020) 9980–9994.

[15] A. Alreshidi, A. Ahmad, Architecting software for the internet of thing based systems, Future Internet 11 (2019) 153.

[16] J. Estdale, E. Georgiadou, Applying the iso/iec 25010 quality models to software product, in: European Conference on Software Process Improvement, Springer, 2018, pp. 492–503.

[17] T. Qiu, Z. Zhao, T. Zhang, C. Chen, C. P. Chen, Underwater internet of things in smart ocean: System architecture and open issues, IEEE transactions on industrial informatics 16 (2019) 4297–4307.

[18] C. Hu, Y. Pu, F. Yang, R. Zhao, A. Alrawais, T. Xiang, Secure and efficient data collection and storage of iot in smart ocean, IEEE Internet of Things Journal 7 (2020) 9980–9994.

[19] W. Hasselbring, Software architecture: Past, present, future, in: The Essence of Software Engineering, Springer, Cham, 2018, pp. 169–184.

[20] B. Morin, N. Harrand, F. Fleurey, Model-based software engineering to tame the iot jungle, IEEE Software 34 (2017) 30–36.

[21] F. Zambonelli, Towards a discipline of iot-oriented software engineering., in: WOA, 2016, pp. 1–7.

[22] A. Razzaq, A systematic review on software architectures for iot systems and future direction to the adoption of microservices architecture, SN Computer Science 1 (2020) 1–30.

[23] M. Fahmideh, A. Ahmad, A. Behnaz, J. Grundy, W. Susilo, Software engineering for internet of things: The practitioners' perspective, IEEE Transactions on Software Engineering 48 (2021) 2857–2878.

[24] D. Halpern, Satellites, oceanography and society, Elsevier, 2000.

[25] O. L. Osen, H. Wang, K. B. Hjelmervik, H. Sch, et al., Organizing data from industrial internet of things for maritime operations, in: OCEANS 2017-Aberdeen, IEEE, 2017, pp. 1–5.

[26] T. A. S. Siriweera, I. Paik, B. T. Kumara, K. Koswatta, Intelligent big data analysis architecture based on automatic service composition, in: 2015 IEEE International Congress on Big Data, IEEE, 2015, pp. 276–280.

[27] E. H. Belghith, F. Rioult, M. Bouzidi, Acoustic diversity classifier for automated marine big data analysis, in: 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2018, pp. 130–136.

[28] J. Song, H. Xie, Y. Feng, Correlation analysis method for ocean monitoring big data in a cloud environment, Journal of Coastal Research (2018) 24–28.

[29] M. C. Domingo, An overview of the internet of underwater things, Journal of Network and Computer Applications 35 (2012) 1879–1890.

[30] A. A. Saucan, M. Z. Win, Information-seeking sensor selection for ocean-of-things, IEEE Internet of Things Journal 7 (2020) 10072–10088.

[31] J. Waterston, J. Rhea, S. Peterson, L. Bolick, J. Ayers, J. Ellen, Ocean of things: Affordable maritime sensors with scalable analysis, in: OCEANS 2019-Marseille, IEEE, 2019, pp. 1–6.

[32] N. Wright, H. Chan, Low-cost internet of things ocean observation, in: OCEANS 2016 MTS/IEEE Monterey, IEEE, 2016, pp. 1–5.

[33] M. Popescu, P. J. Dugan, M. Pourhomayoun, D. Risch, H. W. Lewis III, C. W. Clark, Bioacoustical periodic pulse train signal detection and classification using spectrogram intensity binarization and energy projection, arXiv preprint arXiv:1305.3250 (2013).

[34] J. Schmidhuber, Deep learning in neural networks: An overview, Neural networks 61 (2015) 85–117.

[35] L. Deng, D. Yu, Deep learning: Methods and applications, Found. Trends Signal Process. 7 (2014) 197–387. URL: https://doi.org/10.1561/2000000039. doi:10.1561/2000000039.

[36] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (2015) 436–444.

[37] L. Bellatreche, P. Furtado, M. K. Mohania, Guest editorial: a special issue in physical design for big data warehousing and mining, Distributed and Parallel Databases 34 (2016) 289–292.

[38] Y. Demchenko, C. De Laat, P. Membrey, Defining architecture components of the big data ecosystem, in: 2014 International Conference on Collaboration Technologies and Systems (CTS), IEEE, 2014, pp. 104–112.

[39] Y. Du, Z. Wang, D. Huang, J. Yu, Study of migration model based on the massive marine data hybrid cloud storage, in: 2012 First International Conference on Agro-Geoinformatics (Agro-Geoinformatics), IEEE, 2012, pp. 1–4.

[40] K. Yang, X. Jia, K. Ren, R. Xie, L. Huang, Enabling efficient access control with dynamic policy updating for big data in the cloud, in: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, IEEE, 2014, pp. 2013–2021.

[41] D. Huang, D. Zhao, L. Wei, Z. Wang, Y. Du, Modeling and analysis in marine big data: advances and challenges, Mathematical Problems in Engineering 2015 (2015).

[42] E. Y. Chang, H. Bai, K. Zhu, Parallel algorithms for mining large-scale rich-media data, in: Proceedings of the 17th ACM international conference on Multimedia, 2009, pp. 917–918.

[43] C. K.-S. Leung, Y. Hayduk, Mining frequent patterns from uncertain data with mapreduce for big data analytics, in: International Conference on Database Systems for Advanced Applications, Springer, 2013, pp. 440–455.

[44] C. K.-S. Leung, R. K. MacKinnon, F. Jiang, Reducing the search space for big data mining for interesting patterns from uncertain data, in: 2014 IEEE International Congress on Big Data, IEEE, 2014, pp. 315–322.

[45] X. Wu, S. Zhang, Synthesizing high-frequency rules from different data sources, IEEE Transactions on Knowledge and Data Engineering 15 (2003) 353–367.

[46] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 71–80.

[47] C. Tziortzioti, D. Amaxilatis, I. Mavrommati, I. Chatzigiannakis, Iot sensors in sea water environment: Ahoy! experiences from a short summer trial, Electronic Notes in Theoretical Computer Science 343 (2019) 117–130.

[48] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering–a systematic literature review, Information and software technology 51 (2009) 7–15.

[49] J. Yang, J. Wen, B. Jiang, H. Song, F. Kong, Z. Zhen, Data resolution improvement for ocean of things based on improved fcm, in: 2020 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2020, pp. 709–713.

[50] X. Deng, Y. Jiang, L. T. Yang, L. Yi, J. Chen, Y. Liu, X. Li, Learning-automata-based confident information coverage barriers for smart ocean internet of things, IEEE Internet of Things Journal 7 (2020) 9919–9929.

[51] C. Hu, Y. Pu, F. Yang, R. Zhao, A. Alrawais, T. Xiang, Secure and efficient data collection and storage of iot in smart ocean, IEEE Internet of Things Journal 7 (2020) 9980–9994.

[52] C. Wang, J. Lu, X. Ding, C. Jiang, J. Yang, J. Shen, Design, modeling, control, and experiments for a fish-robot-based iot platform to enable smart ocean, IEEE Internet of Things Journal 8 (2021) 9317–9329.

[53] X. Li, H. Liu, W. Wang, Y. Zheng, H. Lv, Z. Lv, Big data analysis of the internet of things in the digital twins of smart city based on deep learning, Future Generation Computer Systems 128 (2022) 167–177.