Measurement: Sensors 33 (2024) 101114

Contents lists available at ScienceDirect

Measurement: Sensors

journal homepage: www.sciencedirect.com/journal/measurement-sensors

Investigation and implementation of digital software architecture based on internet of things

Jie Liu [a], Boxiang Gong [b,*], Lan Yang [c]

[a] Guizhou Light Industry Technical College, Guiyang, 550000, Guizhou, China
[b] Guizhou Xishan Technology Co., Ltd., Guiyang, 550081, Guizhou, China
[c] Guizhou City Vocational College, Guiyang, 550025, Guizhou, China

ARTICLE INFO

Keywords:
Digital software architecture
Internet of things
Data preprocessing
Data mining
Cluster analysis

ABSTRACT

Application systems are becoming increasingly complex, and software system architecture plays an increasingly important role in the software development process. In view of the problems of the traditional software architecture, such as slow data processing speed, poor stability of the architecture and low accuracy of data classification, this paper studied the digital software architecture based on the Internet of Things (IoT) technology, and conducted data pre-processing on the data collected by sensors, mainly including data reading, data cleaning, data transformation and data reduction. Through experiments, it could be found that the accuracy of digital software architecture using data preprocessing for data classification was more than 93.52%, and the average accuracy of 10 experiments was 94.55%. At the same time, the digital software architecture based on the IoT was tested, and it was found that both the execution time and the original memory were better than other digital software architecture. The overall performance of digital software architecture based on the IoT was better.

1. Introduction

The digitization, networking, informatization, and globalization of social development are one of the characteristics of the new century. With the rapid development of computer network and information technology, as well as the rapid development of the global economy, in order to make the software architecture system more in line with people's needs, the goal of software design has gradually changed between data structure and algorithm research, and the focus of research has shifted to the design of the software architecture system. In the environment of the IoT, the behavior of entities in most software architecture applications is often affected by message exchange, and the behavior of entities is concurrent, independent, and self evolving. For traditional software architecture, it is not determined by the behavior of each part. This would affect the performance of the entire architecture, which is not conducive to meeting people's growing needs, and there are also some shortcomings. Therefore, this paper would study the digital software architecture based on the IoT, so that the behavior of each area in the software architecture can be determined, and all data operations can occur on the architecture, so as to improve the speed of digital software architecture and the accuracy of processing different data, and overcome the shortcomings of traditional software architecture.

Digital software architecture is a series of related abstract patterns used to guide the design of all aspects of large software systems. It is a sketch of a system that describes the abstract components that directly constitute the system and the connections between the components. Digital software architecture can also guarantee the security, performance and maintainability of the software system. Simmhan Yogesh believed that the data-driven IoT software platform was critical to the realization of manageable and sustainable intelligent utilities and the development of new applications. Therefore, he proposed to use software architecture to solve two key operational activities in intelligent utilities: the IoT structure for resource management and data for decision-making [1]. Saavedra Sueldo Carolina proposed a software architecture design method that integrated discrete event process simulators with robot operating systems, allowing for easy exchange of information between factory components [2]. Biondi Alessandro proposed a visionary software architecture, which allowed deep learning and ensures the security and predictability of the architecture through design. At the same time, in order to achieve this goal, the architecture integrated a variety of different technologies [3]. Ashtari Talkhestani Behrang proposed an intelligent digital twin software architecture. The software architecture system could realize plug-in and production, synchronize with real assets, actively obtain data from the real

* Corresponding author.
E-mail addresses: jieliu324627@163.com (J. Liu), xskjgbx@163.com (B. Gong), 2542578983@qq.com (L. Yang).

https://doi.org/10.1016/j.measen.2024.101114
Received 11 July 2023; Received in revised form 23 February 2024; Accepted 11 March 2024
Available online 12 March 2024

environment, and simulate capabilities, thus emphasizing the added value of digital twins in intelligent automation systems [4]. However, these scholars' research on software architecture is not comprehensive enough, and the research on digital software architecture can be better based on the IoT.

With the continuous expansion of the application scope of the IoT, the research on the IoT in software architecture has become increasingly hot. How to better build an excellent software architecture is the focus of many scholars' research, and has achieved some results. Picone Marco proposed a new edge digital twin software architecture model through the IoT system based on digital twins, making it possible for lightweight replication of physical devices, and providing an efficient digital abstraction layer to support autonomous and standard collaboration of things and services [5]. Hajvali Masoumeh provided a software architecture for the medical system based on the IoT. The purpose was to solve some non Functional requirement of the medical system, explain the proposed architecture through the view method, and use graph transformation to transform the model and express the semantics of the operation [6]. Lamonaca Francesco proposed a Structural health monitoring-IoT based on the IoT paradigm, which was used to monitor a building locally. He used multiple sensors to collect information and used this information to identify potential hazardous damage. The proposed software architecture was a method to meet the synchronization requirements [7]. Jacob Pramod Mathew believed that the IoT did not allow the use of common software architecture in different fields, but allowed adjustments according to user requirements. It provided suggestions for solutions and improvements of different software architecture types, and the interaction between identified software architecture elements would provide better help for different software architecture to analyze different types of software architecture of the IoT [8]. In general, the research content of the IoT in software architecture is more, but the research content of digital software architecture based on the IoT is less. In order to improve this content, this paper would integrate the IoT to study digital software architecture.

Due to the wide range and scope of the IoT, experts and researchers in different fields have different starting points for their research on the IoT. The IoT can extend traditional information networks to a wider physical world, which is also the trend of information technology development and would have a profound impact on the development of economy, society, production, and life. For this reason, this paper would use the IoT to study the digital software architecture, and provided a reference for the flexible and efficient application of the IoT. At the same time, this was also to overcome the problems of slow data processing speed and poor stability of the traditional digital software architecture, and help digital software architecture develop better.

2. Digital software architecture design

Software architecture can impact data analysis in a variety of ways in real-world applications, and distinct architecture types and design philosophies each have special benefits and use cases. In order to effectively support data analysis and business decisions, it is vital to select the suitable software architectural design mode in practical application based on the unique business requirements and circumstances. An event-driven software architecture design approach called "part-driver architecture" uses events to start the system's execution. Event-driven architecture has the potential to facilitate real-time data flow processing and analysis in the field of data analysis. Every transaction, for instance, can be thought of as an event in a financial trading system, and the system can process and analyze transaction data fast using an event-driven methodology to keep an eye on risks and market dynamics in real time. A popular software architecture design mode called "layered architecture" separates the system into several levels, each of which is in charge of carrying out particular tasks. A hierarchical architecture can assist in the division of data into various layers in the field of data analysis. To better support business analysis and decision making, data can be managed and kept at multiple granularities in the banking system, for instance.

Modern business contexts see a tremendous rise in the volume and complexity of data. The massive volume of data gathered from several channels and sources is a significant obstacle to conventional data processing techniques. The inefficiency of traditional data processing methods in handling such massive volumes of data restricts the precision and effectiveness of data analysis. The diversity of data is a significant problem for current software designs for processing and evaluating data, in addition to the volumetric barrier. Different processing and analysis techniques are needed for different kinds of data, including unstructured, semi-structured, and structured data. Thus, traditional software architectures face challenges in processing and analyzing these diverse kinds of data efficiently. Organizations must implement new software architectures and technologies to overcome these obstacles and raise the caliber and productivity of data processing and analysis. Simultaneously, it must prioritize safeguarding data security and privacy, enhancing the system's maintainability and scalability, and encouraging data integration and exchange among various applications and systems.

2.1. IoT architecture

Large-scale software systems are designed and implemented using digital software architecture, a high-level abstraction that addresses the characteristics, behavior, and structure of the system. One of the fundamental ideas of digital software architecture is modular design, which separates the system into separate, reusable modules to increase the software's maintainability and reusability. The ability to execute software development and testing concurrently thanks to the modular design lowers the complexity and expense of the development process. One of the theoretical pillars of digital software architecture is abstract hierarchy, which separates complicated systems into several abstract levels, each concentrating on a distinct element. Another theoretical foundation of digital software architecture is event-driven design, which uses events to start the system's execution process. The system's ability to manage asynchronous and non-blocking scenarios is enhanced by its event-driven architecture, which also boosts concurrency performance and reaction time.

The architecture of the IoT has many different characteristics. Compared with traditional internet communication and mobile communication, the related devices of the IoT are also developed and updated based on the characteristics of the IoT or existing network devices [9,10]. Its characteristics are as follows:

(1) The terminal modes of the IoT are becoming more and more numerous, with a very large scale. The development of the IoT would lead to more network terminals being connected to the IoT, and different applications would spawn more IoT terminal models [11,12]. How to connect multiple IoT terminals with different power to the IoT is one of the problems that need to be solved urgently in the current software architecture of the IoT.
(2) The amount of data is huge. IoT sensing systems often collect a large amount of IoT data in real-time [13]. For most terminals, the data collected in real-time can lead to changes in a large amount of data, which would impose a heavy burden on the operation of IoT framework software.

2.2. Architecture design

The structure, behavior, and characteristics of an Internet of Things (IoT) system are covered by the high-level abstraction known as the software architecture. It serves as a guide for the design and implementation of IoT systems. A set of fundamental principles must guide the design and implementation of the IoT software architecture in order to guarantee the system's dependability, maintainability, scalability, and security. The following article will outline the fundamentals of IOT

software architecture design: First, modular architecture One of the key tenets of the Internet of Things' software architecture is modular design, which separates the system into separate, reusable modules with distinct interfaces and functions. The system's scalability, maintainability, and reusability can all be enhanced via modular architecture. (2) Security of data One of the key tenets of the Internet of Things software architecture is data security, which calls for the system to have adequate data availability, confidentiality, and integrity. (3) Sustainability One of the key tenets of the Internet of Things' software architecture is maintainability, which calls for an easily upgradable and maintained system. System stability, maintainability, and upgradeability must all be taken into account during the maintainability design process. Maintainability can lower the system's maintenance costs and complexity while increasing the system's availability and dependability. (4) Security One of the key tenets of digital software architecture is security, which calls for adequate security performance, including data availability, confidentiality, and integrity. The security principle guarantees the stability and dependability of the system and gives it the ability to fend off numerous security threats and attacks.

As the design model blueprint of the whole software system, software architecture is the soul of software design and implementation [14,15]. The quality of software architecture system design and performance directly determines the important characteristics of software in all aspects. From the current software development process, the design of software system architecture usually starts from the actual requirements of the software, considering many factors such as concurrency and distribution, and focuses on user needs. The main purpose of the digital software architecture system design researched by the IoT is to determine the flexibility of system customization and the diversity of processes, so that the software architecture can be set more easily in the configuration mode [16–18].

Digital software architecture aims to direct the design of all major software system components, assisting architects in providing a framework for workable system design solutions that satisfy various client needs. In addition, a guarantee for the software system's performance, security, and maintainability can be given by the digital software architecture. The following are the primary issues that the digital software architecture is addressing:

1. System stability: A stable system framework can be provided by digital software architecture, ensuring that the system can function steadily under a variety of conditions.2. Flexibility of the system: Digital software architecture can offer a flexible system design that allows the system to adjust to changing requirements and various application scenarios.3. System performance: The system can maintain high performance under a variety of conditions thanks to digital software architecture, which can efficiently optimize performance. In summary, the digital software architecture is intended to assist developers in creating software systems that are high-quality, dependable, flexible, and maintainable by addressing specific challenges or areas such as system stability, flexibility, performance, etc. The schematic diagram of the digital software architecture based on the IoT is shown in Fig. 1.

2.3. Module function introduction

The fundamental building blocks of a software architecture are its modules and components, each of which has distinct roles and duties. The precise definition of the modules and components in the software architecture, together with an explanation of their functions, can be found below: The fundamental building component of a software architecture is a module, which is a function code block that may be executed separately and has certain independence and encapsulation. Every module in the system has a unique interface and purpose for

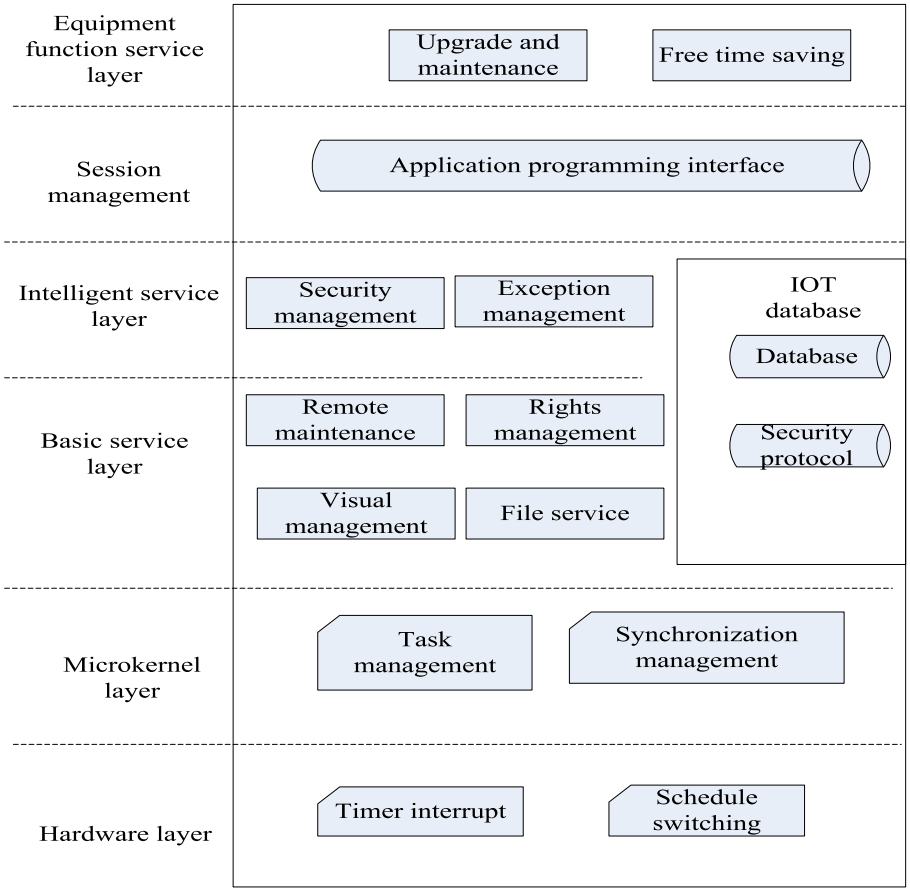

Fig. 1. Schematic diagram of digital software architecture based on IoT.

carrying out a task. One of the key tenets of software architecture design is modular design, which enhances the system's scalability, maintainability, and reusability by breaking it up into several modules. A module's functional explanation Encapsulation: To achieve modularization and code reuse, a module can contain code and work together. Independence: The module has the ability to operate alone and has a separate input/output interface for easy communication with other modules. Reusability: The module's functions can be applied again to cut down on duplication of work and boost productivity. Maintainability: Code can be made more maintainable and less difficult and expensive to maintain by encapsulating and promoting module independence.

The sophisticated unit that makes up the software architecture is called a component. It is a unique module with greater autonomy and professionalism. A component is typically a piece of software that may be installed and operated on its own, offering a range of interfaces for external interaction in support of certain business needs. A sophisticated software architecture can be formed by components working together. Interpretation of components' functions: Professional: Professional and targeted components typically concentrate on particular commercial or functional areas. Independence: To lessen system coupling and increase system scalability and maintainability, components can be installed, operated, and upgraded independently. Interactivity: The system's sophisticated business logic can be accomplished by components working together and interacting through interfaces.

The proper division and design of modules and components in a software architecture can enhance the system's overall performance, maintainability, and scalability. Simultaneously, adhering to the openness and standardization principles can improve the system's ability to integrate and work with other systems.

According to the analysis of the functional modules of the digital software architecture system, the functional module structure diagram of the digital software architecture is divided to achieve the intended functions. The module structure diagram of the digital software system architecture based on the IoT is shown in Fig. 2.

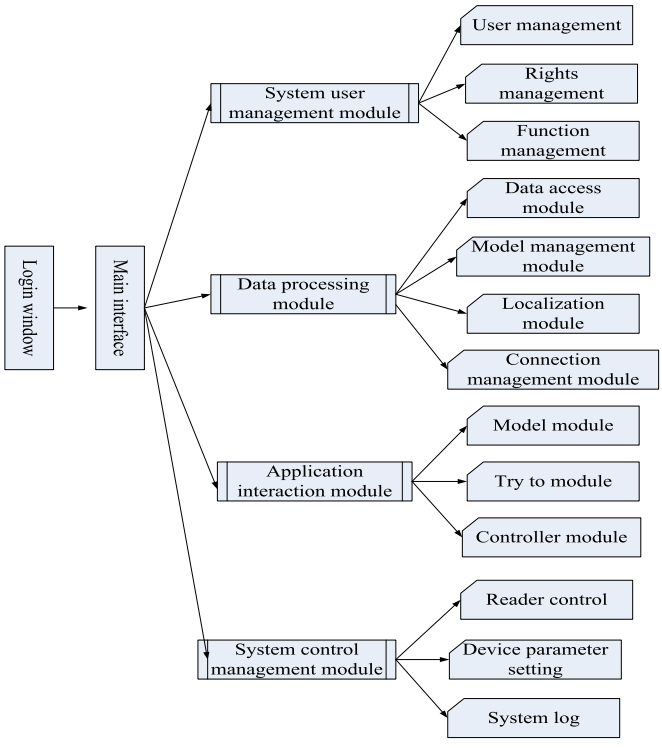

Fig. 2. Module structure diagram of digital software system architecture based on the IoT.

2.3.1. User management module

After the user first accesses the user information management module, the system would grant different permissions to different users based on their registered identity, automatically determining whether the user has access to the subfunction module. If the user has this permission, the system would display the user information query interface. By providing two key information: user account and username, one can directly log in to the system, and the user who logs in can have certain permissions. After finding qualified users, they would be displayed in a list below, allowing users to access the user information viewing interface through a link. The user information would be displayed in plain text.

2.3.2. Data processing module

The data processing module is a specialized module for managing data storage. The bottom layer of this model can use different types of databases to process data information. After processing the information, a model that provides reference materials to the upper layer would bring some benefits. All other module data needs to be stored and processed through the module.

The data storage module is divided into frontend and backend. The front-end is a part that is unrelated to specific information, ensuring that even with special devices, shared information can be provided externally. The front-end includes the following modules:

Data access module: This model uses the upper layer of the data access module. This supports data replication and analysis based on relationships, and supports delayed loading of data information. For special input data, it is necessary to call the backend for specific implementation.

Model management module: This module provides an upper level framework for analyzing data models. Each data model is equivalent to a "class", and developers must determine all data models that should be used based on this standard. Each field in the model must have a type, which can be any content specified in type control.

The front-end abstracts public logic unrelated to specific data storage engines, while the back-end is responsible for implementing various interfaces required by the front-end. Therefore, different data storage engines require different independent backends.

2.3.3. Application interaction module

The application interaction module is responsible for displaying the operation information of the system and providing the function of interaction with users, so that developers can better create a more excellent software architecture. This module is based on the architecture of "model-view-controller". The model module is mainly responsible for communicating with specific organizations and storing data information collected by sensors based on the organization's language. When the data of the module changes, it is necessary to notify the view module of the updated view content, which also prompts the view module to frequently query its data. The view module is mainly responsible for reading data from the model layer and presenting the data in a specific way. At the same time, it is responsible for completing direct interaction with end users and sending user information input to the controller, which informs the model of specific changes in data. In order to avoid damaging the characteristics of the entity, the application interaction module would complete data exchange with the entity through special protocol messages, without the need for a central information server, which is an inevitable requirement for improving interface rendering performance. The network parameter interface is shown in Fig. 3.

The application message would organize all the application data parameters in the original terminal settings interface into this table for reference. The specific expression is as follows:

$$lineDelay_t = \left(\frac{lineLength}{x}\right) * \frac{3}{2} \tag{1}$$

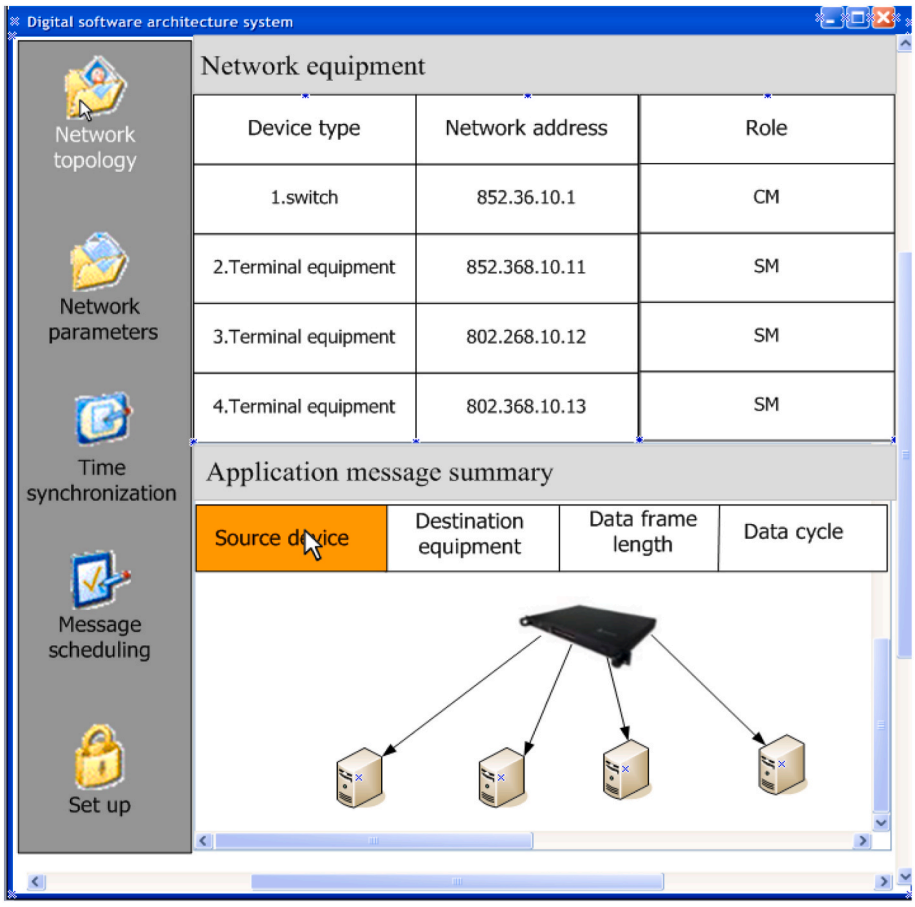

Fig. 3. Network parameter interface.

Among them, x is the reaction speed for interface interaction.

3. Data pre-processing of digital software architecture

The concept of data pre-processing is that before using mature algorithms to estimate the performance model, a lot of processes need to be carried out on the original data, so that it can meet the requirements of relevant models for data pre-processing, and improve the model's utilization of data [19,20]. After preliminary processing, the data would be further corrected and error files would be removed. Duplicate files are merged, and missing files are added. The required files are selected and integrated together, and then the data required for optimizing the model is optimized. The data preprocessing stage is an important and complex process before data analysis. Its importance lies not only in ensuring the accuracy and timeliness of the data, but also in removing irrelevant data from data collection and logs, so that the data has evidence to follow and can well meet the needs of analysis and processing work [21]. The data processing process is shown in Fig. 4.

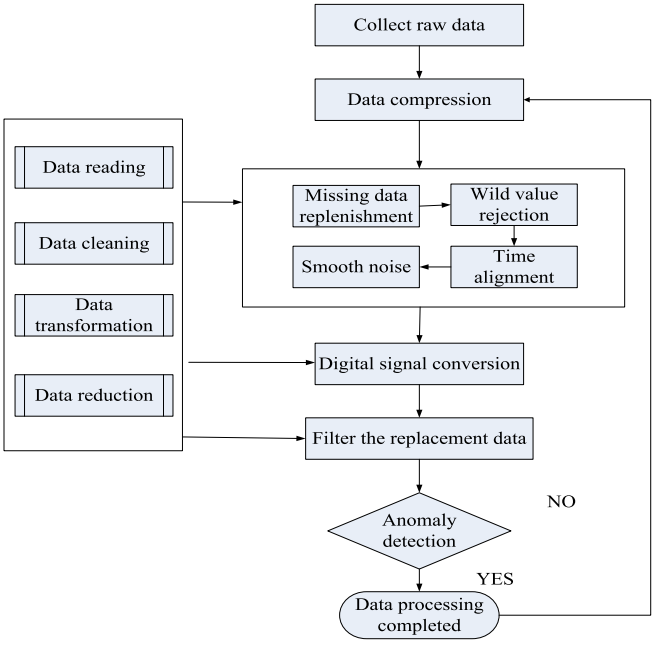

Fig. 4. Data preprocessing flowchart.

3.1. Data reading

Sensors collect data from the real world and theoretically integrate it with the real world. Through scientific research and technological development in recent years, stable, mature, and high-performance sensors are being used for monitoring, and sensors are being developed and distributed to more monitoring points to obtain more information from the real world [22,23]. However, due to the large number of data analysis points and complex formats, a significant amount of data may be generated. If most of the abnormal data in these data cannot be accurately identified, incomplete data would not be covered in a timely manner, which would be unfavorable for data analysis. Monitoring data with large errors or poor status often leads to errors in numerical analysis, which also makes subsequent data analysis more difficult. Therefore, it is necessary to first read the data obtained from the sensor, and

then perform preprocessing.

Assuming that the dataset obtained by the sensor is Y and dataset Y contains m fault types, the specific calculation expression is as follows:

$$F(Y) = \sum_{i=1}^{m} \sum_{j=1}^{K_i} q_{ij}\, log_2 \frac{1}{q_{ij}} * \gamma \tag{2}$$

Among them, q_{ij} belongs to [0,1], and $\gamma = m/K$ is the equilibrium factor.

According to the threshold, if the value of the information entropy of the data obtained by the sensor during the time period is greater than the threshold, the formula is as follows:

$$F(P) = ln\left[\sum_{i=1}^{m} \sum_{j=1}^{K_i} q_{ij}\, log_2 \frac{1}{q_{ij}} * \gamma \right] > F_{ave} = \frac{1}{O} \sum_{i=1}^{m} \sum_{o=1}^{N_i} Q(Y_{io}) log_2 \frac{1}{Q(Y_{io})} * \tag{3}$$

At this point, these data are high entropy data. The higher the information entropy of data, the higher the priority.

3.2. Data cleansing

Due to defects in the algorithm or malfunctions during transmission, different accounts may correspond to the same set of descriptions, resulting in duplicate data. When there is a lot of duplicate information, it is easy to make wrong decisions in the complete information. Creating only one piece of information at the same time can serve as a principle to solve duplicate information, which can delete multiple pieces of information based on the verification time information. To ensure data continuity, if data loss occurs, it is necessary to supplement the lost location. The specific measures for cleaning data are as follows:

(1) The most common value is inserted into the missing space. The existing monitoring data has been subjected to regression analysis and processing, and the values that are most likely to be cleaned have been inferred.
(2) The data that needs to be cleaned is directly ignored. There is a large amount of data in the software architecture, and ignoring a small part of the data that needs to be cleaned has little impact on the overall data preprocessing, which can be ignored basically. The drawbacks of this method are also obvious. When the object being checked is an object with a lower sampling frequency, data loss and distortion may occur. The specific expression is as follows:

$$X_i = \sum_{j=1}^{n} \mu_{ij} x_{ij}(t) ij \tag{4}$$

In the formula, μ_{ij} is the posterior probability of echo i from target j, and x_{ij} is the observed value of echo i in target j tracking gate.

3.3. Data transformation

Heterogeneous survey anomaly data in databases of different formats, scales, and locations are integrated. The data comes from many subsystems, and there is also an issue of abnormal data conversion, which leads to significant differences in data types. When the data is different, a table would contain different subsystems to analyze the data. Therefore, data transformation before data processing is a very important step, which would quickly increase the speed of analysis.

Due to the fact that the information collected by various sensors cannot be directly transformed into useable data, some processing methods need to be used to convert the digital signals collected by sensors into numerical values that can reflect structural characteristics. The transformation formula is as follows:

$$X_i = T(Y - M_0) + A \tag{5}$$

In the formula, X_i is the value after scaling transformation, and M_0 is the initial value of the parameter.

The data collected by wireless sensors can be represented by any variable, and the probability distribution of any variable can be used to characterize the uncertainty of all events:

$$\begin{bmatrix} Y \\ Q(Y) \end{bmatrix} = \begin{bmatrix} Y = Y_1 & Y = Y_2 \cdots Y = Y_i \cdots Y = Y_m \\ Q(Y_1) & Q(Y_2) \cdots Q(Y_i) \cdots Q(Y_m) \end{bmatrix} \tag{6}$$

Among them, $Q(Y) = \{Y_i | 0 \le Q(Y_i) \le 1 \cup \sum_{i=1}^{n} Q(Y_i) = 1\}$.

3.4. Data reduction

Although some data exists, it has no effect on data processing and can also affect the efficiency of the entire data processing. Therefore, reducing data is very necessary. Based on the principle of maintaining data quality as much as possible, the data volume is refined to the maximum extent possible. There are two main methods: selection behavior and data sampling. Selection behavior is used as the basis for data reduction, which includes pruning, finding formulas, and other contents. Pruning is the removal of attributes that have little or no impact on data processing; Branch is the main feature of comparing two sets of data, completing content with similar features; Searching for a formula is to find the relationship between two or more data, which is used to determine anomalies and compare states. The foundation of finding good relationships is relationships, and it is necessary to uncover big relationships. By selecting attributes, the features of the data can be effectively seen, reducing the size of the data. Data sampling is usually carried out using mathematical methods, which replace lengthy data with simple and efficient indicators or data representations, making the entire dataset more efficient and reliable.

The task of time alignment is to align the observation data from different sensors with different sampling intervals to a unified fusion time interval. The specific expression is as follows:

$$\widehat{x_i}(m) = b_1 \sum_{i=1}^{m} x_i + b_2 \sum_{i=1}^{m} i * x_i \tag{7}$$

Among them, $b_1 = -2/m$.

4. Experiment part of digital software architecture based on IoT

This experiment aims to investigate digital software architecture design and optimization techniques in an Internet of Things (IoT) context. We will investigate ways to increase the effectiveness of data collection, processing, and transmission while maintaining system dependability and safety by examining the actual sensor data. The real set of data used in this experiment is derived from Company A's data. The data is gathered on the network by means of the crawler software. The majority of the data consists of specific personnel's activity details, such as time stamp, location, and kind of action. To get rid of duplicate data and outlier values, the data set is cleaned and preprocessed.

Throughout the experiment, make the following assumptions: (1) IOT devices and sensor nodes are capable of stable, accurate operation. (2) During data transmission, it can be accurately categorized and identified. (3) It is capable of efficiently gathering, analyzing, and sending sensor data.

The following steps make up the experiment:A. Data collection: Utilize the smart home company's sensor nodes to gather information on a predetermined interval.B. Preprocessing the data: remove impurities, clean up, and extract features from the gathered information.C. System performance evaluation: Measure metrics like packet loss rate, latency, and data transmission time to assess the system's dependability and performance.

The following are the particular cases, constraints, and metrics of the Internet of Things-based digital software architecture assessment: The architecture can be utilized to create a more effective, convenient, and

secure living and working environment in the areas of smart homes, intelligent transportation, medical, and other situations, as well as intelligent management and control of various goods and equipment. The essential component of the Internet of Things architecture is the cloud platform layer of software, which enables the management, storage, and analysis of diverse data while offering the required service support for higher-level applications. The following are the primary index parameters used in the Internet of Things-based digital software architecture evaluation: The Internet of Things system's level of intelligence and performance are measured by the intelligent performance index. Safety performance indicators: These are metrics for the Internet of Things system's security and preventive measures. Reliable performance indicators: These speak to the Internet of Things system's dependability and productivity.

A lot of data can be collected through sensors in the IoT. Classifying and subdividing these data would help eliminate unimportant data and help digital software architecture run faster. The data preprocessing method is used to classify 100,000 pieces of data collected by the sensor, and it is found that it can effectively improve the accuracy of data classification of the digital software architecture based on the IoT. In order to avoid accidents in the experiment, an iterative experiment was carried out, and the obtained data was classified. This experiment was repeated 10 times, and there may be some minor changes or improvements between each experiment, so that the experimental data can be more scientific and reasonable and accurate conclusions can be drawn. The results obtained from 10 experiments were also compared to a digital software architecture based on data mining and cluster analysis. The specific comparison results are shown in Fig. 5.

The x-axis in Fig. 5 represents the number of experiments, with a total of 10 conducted. The y-axis represents the accuracy of data classification. Through the study of Fig. 5, the accuracy of digital software architecture for data classification was effectively improved by using data preprocessing methods. Compared with other two digital software architecture, the accuracy of classification had obvious advantages, which was conducive to improving the utilization of data by the architecture. Among them, the accuracy of digital software architecture using data preprocessing for data classification was more than 93.52%, and the average accuracy of 10 experiments was 94.55%. The classification accuracy of digital software architecture based on data mining and cluster analysis was below 91.8% and 91.2% respectively. The average values of the 10 experiments were 4.33% and 4.66% lower than the digital software architecture using data preprocessing, respectively. The digital software architecture using data preprocessing had the lowest classification accuracy for data in the ninth experiment, which was only 93.53%, but it was 2.8% and 5.04% higher than the digital software architecture based on data mining and cluster analysis, respectively. The digital software architecture using data mining had the highest classification accuracy for data in the sixth experiment, which was 91.77%, but it was 2.48% lower than the digital software architecture using data preprocessing; the digital software architecture using cluster analysis had the highest classification accuracy for data in the second experiment, which was 91.12%, but it was still 3.16% lower than the digital software architecture using data preprocessing.

Whether an architecture is excellent or not, in addition to the reasonable allocation of data and the reasons for strengthening data, it also needs to take into account the time of data processing of the architecture. If the architecture takes a very long time for data processing, it is also not possible. However, the digital software architecture using data preprocessing requires less time to process different amounts of data, and the processing speed is faster. In order to better reflect its advantages in data processing speed, the time obtained by processing 1 million pieces of data, 2 million pieces of data, 3 million pieces of data and other data was compared with the data processing time of digital software architecture based on data mining and cluster analysis. The specific comparison results are shown in Fig. 6.

In Fig. 6, the x-axis represents the specific amount of data extracted, and the unit of these data is 10,000, while the y-axis represents the time required for processing the extracted data, and the unit is minute. As shown in Fig. 6, the data preprocessing method effectively improves the speed of digital software architecture for data processing and reduces the time required for data processing. Compared with other two digital software architecture, it has absolute advantages in data processing. Among them, when the amount of data to be processed is 1 million, the time required for the digital software architecture using the data preprocessing method is 1.2 min, which is less than 0.7 min and 0.5 min respectively than the digital software architecture using data mining and cluster analysis; When the amount of data to be processed is 10 million, the time required for the digital software architecture using the data preprocessing method is 6.8 min, which is less than 5.1 min and 7.8 min respectively than the digital software architecture using data mining and cluster analysis. When the amount of data to be processed is less than 6 million, the digital software architecture using data mining needs far more time for data processing than the digital software architecture of data preprocessing and cluster analysis, especially the digital software architecture using data preprocessing methods; when the amount of data to be processed is more than 7 million, the digital software architecture using data mining requires more time for data processing than the software architecture based on data preprocessing methods, but less than the software architecture using clustering analysis. To sum up, the digital software architecture uses data preprocessing methods to process data, which requires the least time and has the best performance.

The digital software architecture is tested, mainly on the execution time of the digital software architecture, the success rate of data operation, the occupancy rate of the Central Processing Unit (CPU), and the memory occupancy rate. The test is carried out by using different data sets, such as 100,000 pieces of data, 200,000 pieces of data, etc., to verify the superiority of the digital software architecture based on the

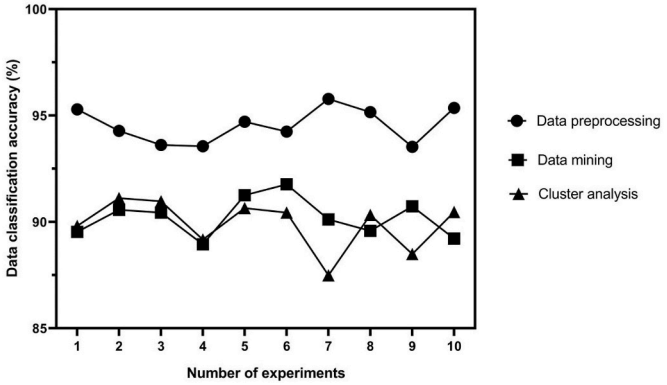

Fig. 5. Comparison of data classification accuracy of three different digital software architecture.

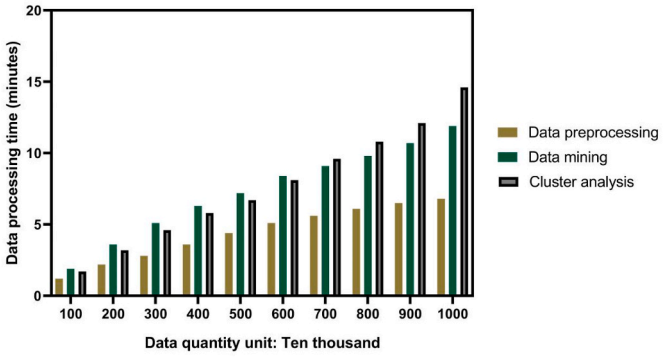

Fig. 6. Comparison of data processing time required by three different digital software architecture.

IoT. The architecture test results are compared with the digital software architecture based on Service-Oriented Architecture (SOA), Data Mining (DM), and Cluster Analysis (CA). The specific comparison results are shown in Table 1.

As shown in Table 1, the digital software architecture based on different methods is tested. Through many tests, it can be found that the digital software architecture based on the IoT has advantages in all aspects, and is superior to other digital software architecture in performance. With the increase of the amount of data, although the execution time of the digital software architecture based on the IoT is increasing, its growth rate is smaller than the other three digital software architecture. At the same time, although the success rate of data operation is also decreasing with the increase of data volume, the speed of reduction is lower than that of the other three digital software architecture. For the CPU share, from 100,000 to 300,000 pieces of data, the increase of digital software architecture based on the IoT is 4.1%. The digital software architecture based on SOA, DM and CA is 5.5%, 5.49% and 13.5% respectively. The CPU share of the digital software architecture of the IoT is 1.4%, 1.39% and 9.4% lower than those of the three digital software architecture. For memory occupancy, from 100,000 to 300,000 pieces of data, the increase of digital software architecture based on the IoT is 2.52%. The digital software architecture based on SOA, DM and CA is 9.87%, 9.46% and 8.11% respectively. The CPU share of the digital software architecture of the IoT is 7.35%, 6.94% and 5.59% lower than those of the three digital software architecture. To sum up, the performance of the digital software architecture based on the IoT is better in all aspects. No matter the executed data, or the data's share of memory and CPU, it is better than other digital software architecture.

Depending on the needs and application scenarios, the digital software architecture based on the Internet of Things and the software architecture based on data mining and clustering analysis each offer advantages.

The benefits of digital software architecture based on the Internet of Things are mostly seen in the following areas: Scalability: By simplifying the overall architecture and reducing the need for physical device maintenance and application, the Internet of Things design allows devices to be deployed in more places. Greater extensibility to enable a wider range of application situations. Greater effectiveness. The Internet of Things can calculate data with strong timeliness, and communication efficiency is always improving, resulting in decreasing information transmission delays. Productivity can rise in tandem with the rise in decision-making rate. Cut costs: By lowering the investment in physical equipment, the Internet of Things can help businesses invest more in other areas of improvement and capital applications. This includes lowering the cost of purchasing and maintaining equipment later on. Simplified administration: By simplifying the Internet of Things architecture, it is possible to drastically cut down on the amount of management content that is involved in day-to-day operations and instead concentrate on monitoring actual objects. It can also use the Internet of Things to provide remote maintenance and troubleshooting in the event of a failure, which will lessen the workload. More expert data analysis: With the aid of the Internet of Things platform, certain small and medium-sized businesses that are comparatively low on technology can take advantage of the industry's relatively sophisticated technical resources to perform a more expert analysis of the data from their current operations and identify areas in which they fall short. Boost operability: The Internet of Things offers robust operability, a wide range of protocol compatibility, and excellent compatibility. Compatibility can be somewhat improved by its linked devices.

The benefits of data mining and cluster analysis-based software architecture are mostly seen in big data processing, which can efficiently mine and analyze massive data sets, extract important knowledge and information, and support decision-making and planning for businesses. In addition, cluster analysis can assist businesses in grouping and classifying data to improve data utilization and understanding. In conclusion, there are benefits to both the Internet of Things-based digital software architecture and the data mining and clustering analysis-based software architecture. Businesses should select the right architecture based on their unique requirements and use cases.

Table 1
Test results of different types of digital software architecture.

Evaluating indicator	IoT	SOA	DM	CA
Extract 100,000 pieces of data				
Execution time (s)	22	29	31	35
Success rate of data operation (%)	98.81	91.26	90.37	91.55
CPU usage rate (%)	41.2	54.9	58.3	56.7
Memory share (%)	40.36	50.31	48.37	60.28
Extract 200,000 pieces of data				
Execution time (s)	28	42	57	66
Success rate of data operation (%)	98.66	90.88	89.71	90.76
CPU usage rate (%)	43.6	61.7	66.8	65.4
Memory share (%)	41.57	54.39	53.26	64.12
Extract 300,000 pieces of data				
Execution time (s)	36	59	73	87
Success rate of data operation (%)	98.47	90.41	89.49	90.12
CPU usage rate (%)	45.3	60.4	63.79	70.2
Memory share (%)	42.88	60.18	57.83	68.39

5. Conclusions

In this era of rapid development of information technology, people's dependence on computers has also become stronger and stronger. More and more work is gradually separated from human labor and is completed by relying on software architecture. Therefore, people have higher and higher requirements for software architecture. The existing software architecture can not meet the growing needs of people, and its processing speed for complex data is relatively slow. In the face of large-scale data, the traditional software architecture is easy to develop faults when processing data. This paper studied the digital software architecture based on the IoT, and compared it with the traditional software architecture. It was found that the digital software architecture based on the IoT was more in line with people's needs. It also had faster data processing speed and overall performance. This greatly improved the efficiency of software research and development, reduced the possible errors in the process of demand analysis and software architecture design, and reduced the human resources and costs of the later maintenance of software architecture.

The following factors primarily illustrate the limitations of the development and application of Internet of Things-based digital software architecture: 1. Inadequate institutional standards: Because the Internet of Things' architecture spans many different technical domains, there aren't enough comprehensive technical standards for it because there is ineffective coordination and communication between the organizations that create them. This has created certain challenges for the global adoption and advancement of Internet of Things technologies. 2. Expensive: The deployment of an Internet of Things-based digital software architecture calls for a large amount of infrastructure support and equipment, which might have expensive procurement and operating costs.3. Concerns about security and privacy: The Internet of Things technology must gather and examine a lot of data, some of which may contain private information about an individual. It is crucial to figure out how to secure these data's security and privacy.

Funding

This manuscript did not receive funding in form.

Declaration of competing interest

The authors declare that there is no conflict of interest with any financial organizations regarding the material reported in this manuscript.

Data availability

Data will be made available on request.

References

[1] Yogesh Simmhan, P. Ravindra, S. Chaturvedi, M. Hegde, R. Ballamajalu, Towards a data-driven IoT software architecture for smart city utilities, Software Pract. Ex. 48 (7) (2018) 1390–1416.

[2] Carolina Saavedra Sueldo, I. Perez Colo, M. De Paula, S.A. Villar, G.G. Acosta, ROS-based architecture for fast digital twin development of smart manufacturing robotized systems, Ann. Oper. Res. 322 (1) (2023) 75–99.

[3] Alessandro Biondi, F. Nesti, G. Cicero, D. Casini, G. Buttazzo, A safe, secure, and predictable software architecture for deep learning in safety-critical systems, IEEE Embedded Systems Letters 12 (3) (2019) 78–82.

[4] Ashtari Talkhestani, Behrang, T. Jung, B. Lindemann, N. Sahlab, N. Jazdi, W. Schloegl, M. Weyrich, An architecture of an intelligent digital twin in a cyber-physical production system, Automatisierungstechnik 67 (9) (2019) 762–782, *at-*.

[5] Marco Picone, Marco Mamei, Franco Zambonelli, A flexible and modular architecture for edge digital twin: implementation and evaluation, ACM Trans. Internet Technol. 4 (1) (2023) 1–32.

[6] Masoumeh Hajvali, S. Adabi, A. Rezaee, M. Hosseinzadeh, Software architecture for IoT-based health-care systems with cloud/fog service model, Cluster Comput. 25 (1) (2022) 91–118.

[7] Francesco Lamonaca, C. Scuro, D. Grimaldi, R.S. Olivito, P.F. Sciammarella, D. L. Carnì, A layered IoT-based architecture for a distributed structural health monitoring system, Acta Imeko 8 (2) (2019) 45–52.

[8] Pramod Mathew Jacob, Prasanna Mani, Software architecture pattern selection model for Internet of Things based systems, IET Softw. 12 (5) (2018) 390–396.

[9] G. Tsoukaneri, M. Condoluci, T. Mahmoodi, M. Dohler, M.K. Marina, Group communications in narrowband-IoT: architecture, procedures, and evaluation, IEEE Internet Things J. 5 (3) (2018) 1539–1549.

[10] J. Mocnej, W.K. Seah, A. Pekar, I. Zolotova, Decentralised IoT architecture for efficient resources utilisation, IFAC-PapersOnLine 51 (6) (2018) 168–173.

[11] L. Liu, J. Xu, Y. Huan, Z. Zou, S.C. Yeh, L.R. Zheng, A smart dental health-IoT platform based on intelligent hardware, deep learning, and mobile terminal, IEEE journal of biomedical and health informatics 24 (3) (2019) 898–906.

[12] I. Cvitić, D. Peraković, M. Periša, M.D. Stojanović, Novel classification of IoT devices based on traffic flow features, J. Organ. End User Comput. 33 (6) (2021) 1–20.

[13] Hanyur Abdullah, Functional polymer materials in environmental biosensors in the context of the internet of things, Academic Journal of Environmental Biology 3 (3) (2022) 60–68.

[14] Lijun Lun, Xin Chi, Hui Xu, Coverage criteria for component path-oriented in software architecture, Eng. Lett. 27 (1) (2019) 40–52.

[15] Amarjeet Prajapati, A particle swarm optimization approach for large-scale many-objective software architecture recovery, Journal of King Saud University-Computer and Information Sciences 34 (10) (2022) 8501–8513.

[16] A.S. Antonov, R.V. Maier, Development and implementation of the Algo500 scalable digital platform architecture, Lobachevskii J. Math. 43 (4) (2022) 837–847.

[17] Marcelo Campo, Analia Amandi, Julio Cesar Biset, A software architecture perspective about Moodle flexibility for supporting empirical research of teaching theories, Educ. Inf. Technol. 26 (1) (2021) 817–842.

[18] Varuun Vermane, Multilayer distributed system software architecture based on aspect service and web service, Distributed Processing System 2 (4) (2021) 52–60.

[19] Dimitry Tegunov, Patrick Cramer, Real-time cryo-electron microscopy data preprocessing with Warp, Nat. Methods 16 (11) (2019) 1146–1152.

[20] J. Cai, Y. Yang, H. Yang, X. Zhao, J. Hao, ARIS: a noise insensitive data pre-processing scheme for data reduction using influence space, ACM Trans. Knowl. Discov. Data 16 (6) (2022) 1–39.

[21] Poornima Unnikrishnan, V. Jothiprakash, Data-driven multi-time-step ahead daily rainfall forecasting using singular spectrum analysis-based data pre-processing, J. Hydroinf. 20 (3) (2018) 645–667.

[22] Y. Li, M. Yang, J. Hua, Z. Xu, J. Wang and X. Fang, "A channel attention-based method for micro-motor armature surface defect detection," IEEE Sensor. J., doi: 10.1109/JSEN.2022.3159293..

[23] Zeng, X., Wang, Z., & Hu, Y. “Enabling Efficient Deep Convolutional Neural Network-Based Sensor Fusion for Autonomous Driving.” *arXiv preprint arXiv* (2022):2202.11231..