Received February 13, 2017, accepted March 8, 2017, date of publication March 15, 2017, date of current version April 24, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2682258

CPCA: The Cloud Platform of Complex Virtual Instruments System Architecture

CHAO LIU, ZHONGWEN GUO, YUAN FENG, FENG HONG, (Member, IEEE), AND WEI JING
Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China

Corresponding author: Y. Feng (fengyuan@ouc.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61379127, Grant 61379128, and Grant 61572448.

ABSTRACT Virtual instruments is a program that implements functions of an instrument by software which could replace the work of real instruments to save resources. The functions of these sensor-based systems are limited and they commonly cannot manage related information, such as sensors and monitoring objects, due to special requirements. The procedure of development and integration often suffers from low efficiency because of non-standard technologies. To solve the aforementioned problems, an integrated system architecture based on complex virtual instruments (CVI) is proposed, which could not only extend the function of virtual instruments but also ease the development procedure of the system. By analyzing the characters of existing virtual instruments systems, this paper presents the data interaction standard, which has been applied by two IEEE standards, and the architecture, which consists of configuration subsystem, data collection simulation subsystem, Web service registration center, and so on. We also offer a light universal client which could dynamically load the DynamicLinkLibrary of different CVIs, in order to replace the complex integration procedure by scheduling different components. To valid the architecture, three existing systems are reconstructed by the proposed prototype subsystems. The result shows that the proposed architecture is efficient and feasible.

INDEX TERMS Complex virtual instrument, system architecture, development platform, sensor-based system, sensor networks.

I. INTRODUCTION

Sensor-based system has been widely deployed for collecting the data from real world [1]–[3]. These kinds of systems consist of amounts of physical instruments or smart sensors to monitor objective things, such as human bodies, household appliances and environment conditions [4]–[8]. The collected data needs to be transmitted to the cloud server through multi-hop, ethernet, RS232/485 or satellite. In order to intuitively detect the features of the data, all the data needs to be visualized into curves, tables and figures. Virtual instruments is a practical technology to meet the requirements.

Virtual instrument is a program that implements functions of an instrument by software which could replace the work of real instrument to save resources [9]. In the background of ‘‘Industry 4.0’’, the virtual instruments system has been widely deployed and gradually replace the existing sensor-based system. Based on this situation, many real world sensor-based systems are proposed to detect the changing of physical world [10]–[16]. Although these systems are easy to use thanks to their friendly visual interfaces, they often suffer from several problems. First, the virtual instruments only have designed functions, but they commonly do not have the functions of related information management and display, such as sensors, monitoring objects and so on. Second, the instruments of these systems are special designed when the needs of these distributed systems arise. Due to the use of non-standard technologies, the system integration and interoperability is extremely complex. It is difficult to access the data results from different virtual instrument systems in general interfaces with lots of users. Third, the virtual instruments care less about the potential cooperation with other systems. They do not provide service interfaces which can be accessed by other systems. Fourth, the designer has to spend lots of time on writing the codes and debugging them to adapt to different requirements, resulting in long development cycle, huge cost and low efficiency. During the process of developing the application, the combination of software and hardware would also spend more energy.

To solve all the aforementioned problems, we need to propose a new instruments system architecture and a development platform that implements more complex functions, integrates various kinds of virtual instrument systems and provides friendly development Integrated Development Environment(IDE). The architecture can also reduce the development cycle and save the energy on debugging the hardware and software. It is indeed a challenge task. References [16]–[18] solved problems of physical layer design, which alleviate our burden on lower layer. In [19], we proposed an integrated monitoring architecture on Client/Server and Browser/Server which could integrate distributed isomerous senosr-based systems and display the data result into curves. E. Song proposed a service-oriented sensor data interoperability architecture for Institute of Electrical and Electronics Engineers (IEEE) 1451 smart transducers [31]. References [21]–[23] provided some methods to improve the system development efficiency. The mentioned researches all make the contribution to the virtual instruments system development and integration. However, none of these works provide a solution to cover all the challenges mentioned before.

In this paper, we extended the our proposed system architecture CVIS and standard technology in [31] to form a more completed cloud platform which could satisfy a variety of requirements and hardwares. To the best of our knowledge, it is the first custom-built development and implement cloud platform for sensor-based system. To valid our architecture and platform, we reconstruct the existing systems based on proposed architecture, including video surveillance system, household appliance testing system and ocean monitoring system. We also design a drag-and-drop configuration subsystem which further simplifies the development procedure. The process of development and simulation confirms that our architecture is efficient and convenient. Our main contributions are as follows:

(1) We extended the general standard models of the complex virtual instruments system from different kinds of systems, which contributes to system integration and interoperability. The technology has been fully applied by standard IEEE 1851 and P2402.

(2) We provide the concept of complex virtual instruments system which satisfies more complex requirements. The relationship of physical(virtual) instruments and softwares could be dynamically customized by users. Public service interfaces are provided in our architecture.

(3) Development of system is visual and rapid. The development procedure is turned into drag-and-drop components and setting some parameters by given subsystems. The designers do not need the knowledge of programming language.

(4) A light universal client which could dynamically load the DLL of different complex virtual instruments by web service is provided. By the combination of different compiled components and monitoring data, the client could replace the work of different system integration by scheduling different components.

The rest of this paper is organized as follows. Section II outlines the background knowledge of the architecture. The key models of our architecture are illustrated in Section III. Section IV presents the architecture overview. Section V describes the system implementation procedure. Section VI presents the case study. Finally, conclusions and future works are given in Section VII.

II. BACKGROUND

In this section, we will introduce some related works and summarize their characteristics.

A. SENSOR-BASED SYSTEMS

Lots of sensor-based systems have been deployed to monitoring the physical world [10]–[13]. Greenorbs is a forest ecological monitoring system which consist thousands of sensors to collect the temperature, carbon dioxide, humidity [10]. Researchers in [11] use the sensor to detect the vehicle in lane-less traffic. In [12], the authors use smart sensor to detect the air quality by recognize their abnormal behavior.

The mentioned systems are all designed for dedicated purposes. To avoid the waste of resources and reuse sensors, some researchers are focus on configurable system which makes the system universal. The configured system can be easily integrated [19], [20], [22], [23]. For instance, in [23], the author proposes a reconfigurable development system that can adapt to different kinds of sensors to monitor different environments.

However, all these system are dedicated system or designed for special domain. To make the system more flexible and adapts to more common situations, we need a platform which can easily develop sensor-based systems and adapt to different kinds of sensors.

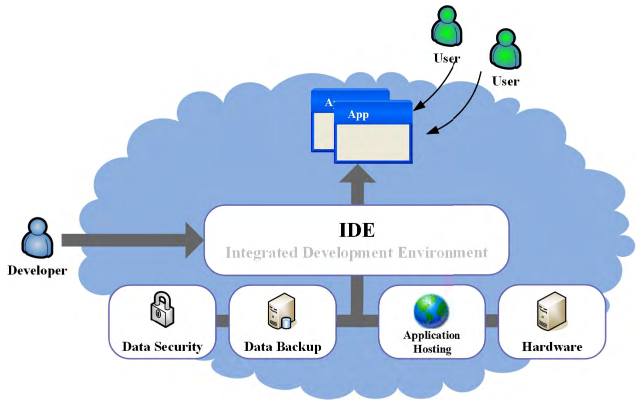


FIGURE 1. Platform as a Service.

B. PLATFORM AS A SERVICE

Platform as a service (PaaS) is a category of cloud computing services [24]. Google Apps, Microsoft Azure and Salesforce CRM are typical cloud computing platform in PaaS [25], [26]. As shown in Fig.1, in these platforms, the developer creates the software using libraries from the provider. The provider provides the hardware platform. Although the coders could develop the software without concerning about hardware foundations, the workload of

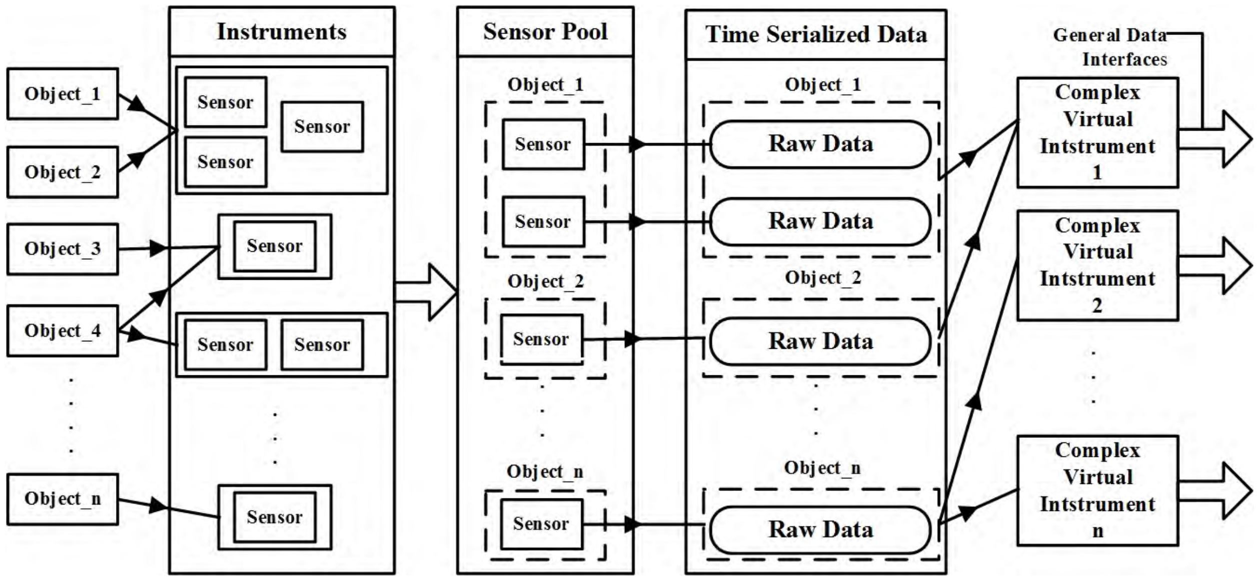


FIGURE 2. Key Models of CVIS.

programming is not relieved [27]. Due to the privacy and security problem, some enterprises are not willing to put there data on third-party platform [28], [29].

III. KEY MODELS

Figure.2 shows the key models which are responsible for the data interaction of the whole system. All of them are independent and integrated with standardized interfaces and mapping model. Based on unified interfaces, the system could be built by setting several parameters through the configuration subsystem which is designed for developers.

A. COMPLEX VIRTUAL INSTRUMENT MODEL

The CVIS is composed of multiple physical(virtual) instruments in order to implement more complex functions. The CVI client is a software which has functions of graphic data display, data management, data analysis, related information management and components reconstruction. The CVI receives the data from either physical sensors or other virtual sources, and has the functions of data display just as traditional virtual instrument. However, this model provides general interfaces for other instruments or remote users to access the data of objects and their corresponding information, such as data result and statistical data. The key components shown in Fig.3 are described as follows:

1) DATA MANAGEMENT MODULE

This module is designed to manage and process the data in this system. In most cases, the data need to be processed before being displayed. This module has the function to achieve this goal. The processed result can be directly used in general virtual instrument engine. The import data should be time serialized data. The management module provides various interfaces for other modules to satisfy more complex functions. It provides similar interfaces as virtual instruments for local user, and also remote interfaces for remote users and other instruments.

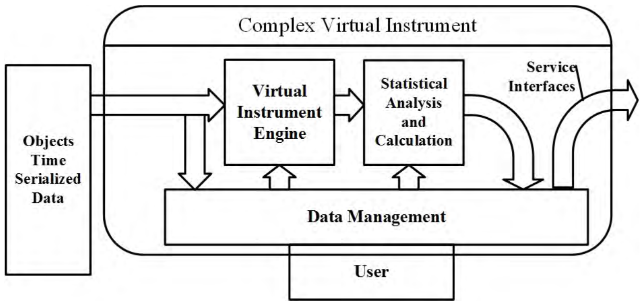

Complex Virtual Instrument
Objects Time Serialized Data
Virtual Instrument Engine
Statistical Analysis and Calculation
Service Interfaces
Data Management
User

FIGURE 3. Complex Virtual Instrument Model.

2) VIRTUAL INSTRUMENT ENGINE

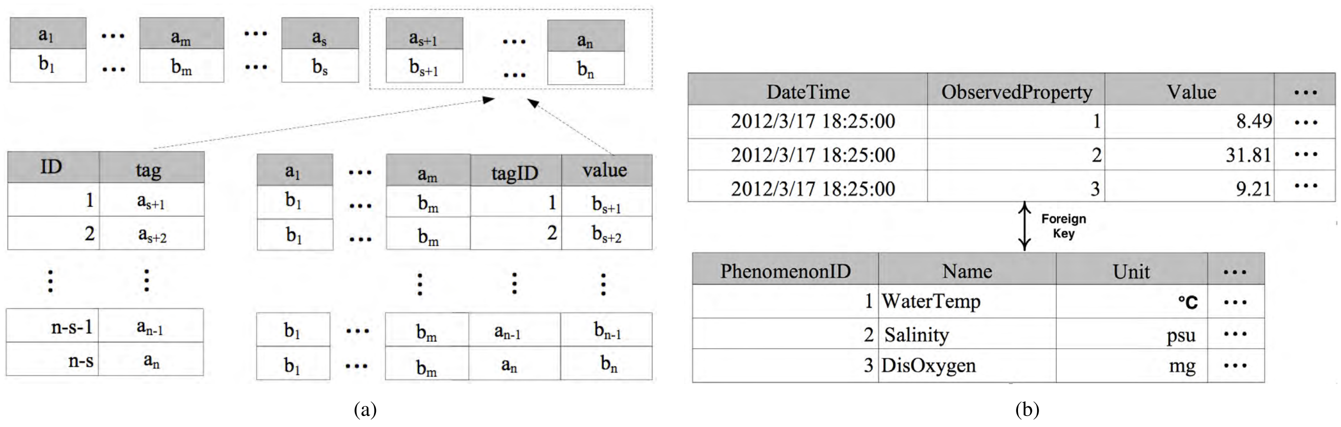
This module receives the data which could directly be used and displays result in unified methods. The engine could dynamically load corresponding components which could adapt to different kinds of data, such as curves, tables and even videos.

3) STATISTICAL MODULE

In addition to data display, sometimes, the users also need the statistical data, such as mean number, median, variance and so on. This module could invoke the DLL to deal with the data.

4) SERVICE INTERFACES

The traditional virtual instruments can replace the data display function of physical instrument. However, it cannot simulate its the data export interfaces. The service interfaces



DateTime	ObservedProperty	Value	…
2012/3/17 18:25:00	1	8.49	…
2012/3/17 18:25:00	2	31.81	…
2012/3/17 18:25:00	3	9.21	…

PhenomenonID	Name	Unit	…
1	WaterTemp	°C	…
2	Salinity	psu	…
3	DisOxygen	mg	…

FIGURE 4. Object Model. (a) Data Structure. (b) Example of Object Description.

of complex virtual instrument could access not only the statistical data but also the information about monitoring objects, sensors and even the DLL of the CVI.

B. OBJECT MODEL

In order to make the system flexible to adapt different kinds of requirements, we need to provide a data structure which could describe all kinds of objects and the attribute can be easily extended during system upgrading. As shown in Fig.4(a), assume object E has attributes $\{a_1, a_2, \ldots, a_n\}$. The primary keys are $\{a_1, \ldots, a_m | m < n\}$. The foreign keys are $\{a_{m+1}, \ldots, a_s | m < s < n\}$. There is one record $\{b_1, b_2, \ldots, b_n\}$. As shown in Fig.4, we put the attribute name and attribute value in different table. When accessing object E's attribute information, we need to get attribute name by tagID and form the whole data by combining the attribute name and attribute value. This structure can be easily extended by add the attribute in tag table.

The example of object model is shown in Fig.4(b), the attribute in the main table only shows the foreign key of each attribute. Before accessing the data, the user should search the attribute name in tag table. No.1 stands for WaterTemp, No.2 stands for Salinity and No.3 stands for DisOxygen. If new attributes added, the developer could add new elements in tag table and refer them in main table.

C. SENSOR POOL MODEL

The sensor pool model should be divided into three parts, sensor group model, sensor interaction command and sensor data model.

Sensor group is usually a sensor node or an instrument combined by several sensors. A group of sensors can execute a task cooperatively. To ensure the flexibility of the model, the sensor group should have four attribute, groupID, groupName, groupDescription and parentID. The groupID is the primary key of sensor group, which can be unique identified. The groupName usually is the name of sensor node or instrument. The groupDescription indicate the function of the sensor group. The parentID indicates parent group. The sensor group can be flexible nestification to realize more complex functions.

Number	①	②	③	④	⑤
Command Contents	yyyy-MM-dd HH:mm:ss.fff	:	DataList	$	Communication Status
Bytes	23	1	N	1	1

FIGURE 5. Sensor Data Model.

Figure 5 shows the extended standard CVIS data format of time serialized sensor data in [31]. The extended version has been formally applied in IEEE standard 1851 and P2402 [30], [34]. The data includes five contents: timestamp, separation, data list, separation and communication status.

The format of timestamp is "yyyy-MM-dd HH:mm:ss.fff", which indicates "year-month-day hour:minute:second: millisecond".

The data list has a standard data format of "sensorID, sensorData@sensorID,sensorData@....". The sensorData is decimalism. If the data is invalid, we use "NULL" to express. The bytes N is no more than 65535. When receiving the data, the client should user "@" and "," to separate the data.

D. MAPPING MODEL

This model describes the mapping relationships between objects, sensors, data sets, complex virtual instruments and so on. The user can easily access related information by mapping model. This model is a great improvement of traditional virtual instrument.

Figure 6 shows the mapping model, we use S, H, P, L, V, CVI stands for object, sensors, physical parameter, logical parameter, observed parameter and complex virtual instrument. The observed parameters contain the physical parameters and logical parameters. Physical parameters are directly acquired from sensors. The logical parameters are calculated by some physical parameters. For example, the salinity of the ocean is calculated by temperature, pressure and conductivity. All the element can be related by incidence matrix

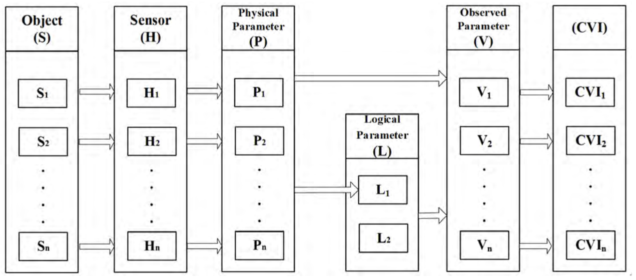


FIGURE 6. Mapping Model.

$M = (M_1, M_2, \ldots, M_k)^T$. $M_k = (m_1, , m_2, m_3, m_4, m_5, m_6)$. m_1 to m_6 are relative number in S, H, P, L, V, CVI. An example of incidence matrix is shown below:

$$M = \begin{pmatrix} 5 & 21 & 1 & 1 & 1 & 1 \\ 5 & 22 & 2 & 2 & 2 & 1 \\ 5 & 23 & 3 & 3 & 3 & 1 \\ 5 & 24 & 4 & 3 & 3 & 1 \end{pmatrix}$$

From the matrix, we know that the object No.5 needs to be monitored by four sensors. The No.3 logical parameter is calculated by No.3 and No.4 physical parameters. All the information will be shown on No.1 complex virtual instrument.

IV. ARCHITECTURE DESIGN

Figure.7 shows the architecture for the CVIS, which consists of data collection subsystem, configuration subsystem, data collection simulation subsystem, web service registration center and a universal client. The data collection subsystem is developed by different companies, which is responsible for collecting the data from physical instruments and transmitting them to the database. Configuration subsystem offers the functions of system customization to adapt to different kinds of monitoring systems. In order to ensure the validity of system configuration, data collection simulation subsystem is designed to pretend the data collection subsystem during system development procedure. The web service registration center provides the functions of web service registration, searching and execution. The main parts of this architecture shown in Fig.7 are described as follows.

A. DATA COLLECTION SUBSYSTEM

This subsystem might be designed by users to fulfill the job of communicating with sensors or intelligent instrument. In order to solve the system interoperability and seamless integration problems, the sensor data collected by the system should be stored in standard format as we proposed. It also needs to compress the data by the method of

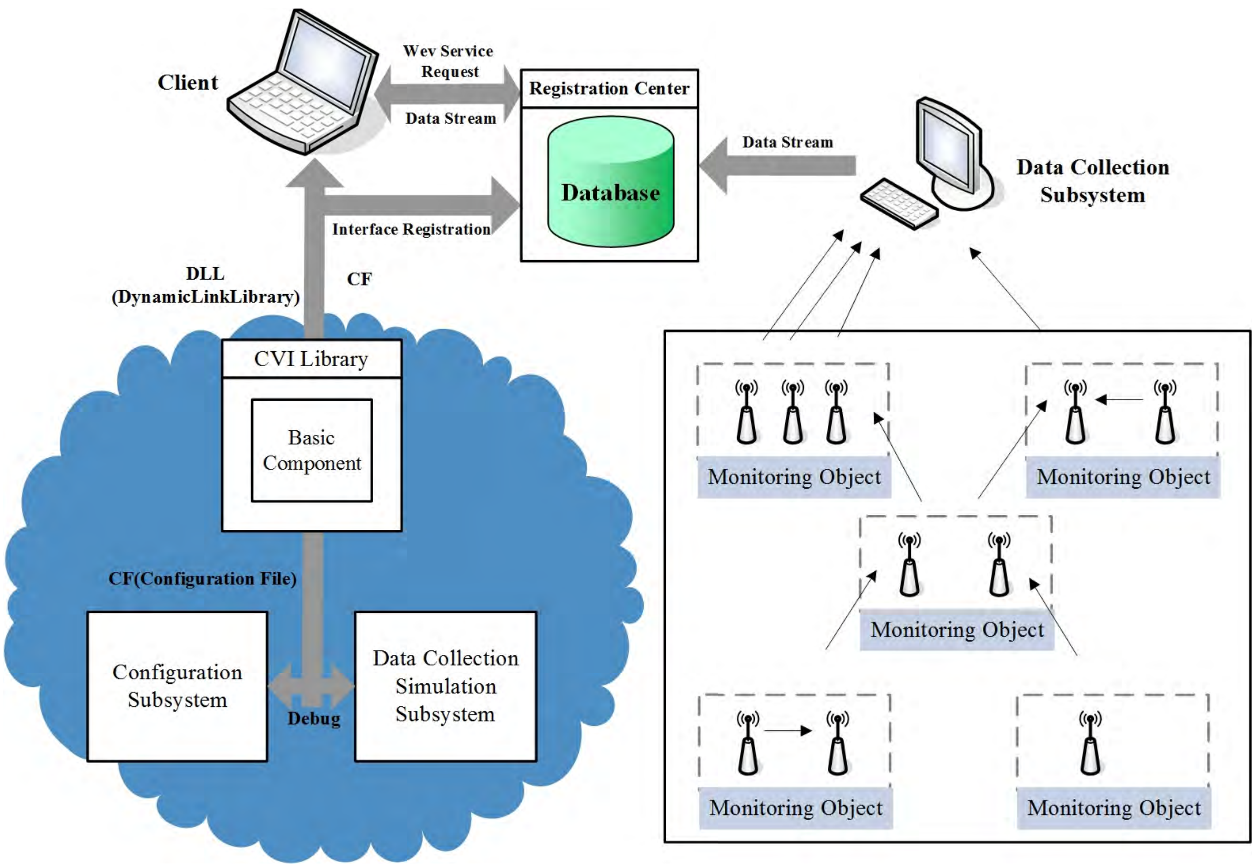


FIGURE 7. Architecture Overview.

relative increment, so as to alleviate the negative impact on system performance by big data.

B. CONFIGURATION SUBSYSTEM

The brand new configuration subsystem is developed to speed up the developing process of CVIS. In [19] and [31], we have turned the software development procedure into setting parameters by their configuration subsystem, which proves to be efficient and feasible. However, the parameter-setting procedure in such method is not very intuitive, and the generated system does not always conform to the programmer's expectations. In order to solve this problem, a platform like Microsoft Foundation Classes (MFC) based on drag-and-drop operations is proposed.

A reusable CVI Library whose components could be loaded by CVI is provided in this architecture. The following components are designed: product information input component; curve display and operation component; vector diagram display and operation component; two-dimensional data display component; video display component; information retrieval component; and data management component. The component based on our standard model could be uploaded to our platform by the third-party.

Developers drag the components from the library to the panel. After dropping it, a configuration window would be popup to setting specific parameters and some correlation information. Then, developer should debug the temporary system with the data collection simulation subsystem. When all the setting procedures are finished, the subsystem would generate a Configuration File (CF) along with the corresponding DLL to users.

C. DATA COLLECTION SIMULATION SUBSYSTEM

Because the data collection devices cannot be provided by our platform, the online developed software will be faced with the problem of mapping with hardware. The data collection simulation subsystem is developed to deal with it. The simulation procedure is executed online, so we do not need to deploy any real devices.

Firstly, the developer should initialize the information data collection devices by using the configuration file written in XML. Secondly, the simulation system receives the data collection orders from the configuration system and analyzes them. Finally, the simulation subsystem return the history data stored in our server to the source.

The subsystem can simulate mainstream data collection devices. According to the communication interfaces between computers and devices, we separate the components into four class, GPIB, USB, RS485 and Ethernet. The interaction between configuration and simulation subsystem is in web service. The used schema based on our key models is shown in Fig.8.

D. WEB SERVICE REGISTRATION CENTER

Web service registration center offers the functions of web service registration, searching and execution. Some dispensable ports are closed within the enterprise's local area network (LAN) to avoid security problem. For the developer without the knowledge of programming language, the employment of web services over Hyper Text Transport Protocol (HTTP) can easily achieve the interoperability and communication problems in CVIS without caring about network protocol.

After system developing, our configuration subsystem will list all the web services which might be used. Developers check the interfaces which he would like to apply before submitting and set their authorities. When downloading the CF and DLL, the configuration subsystem will also register the web service interfaces to the center. All the data

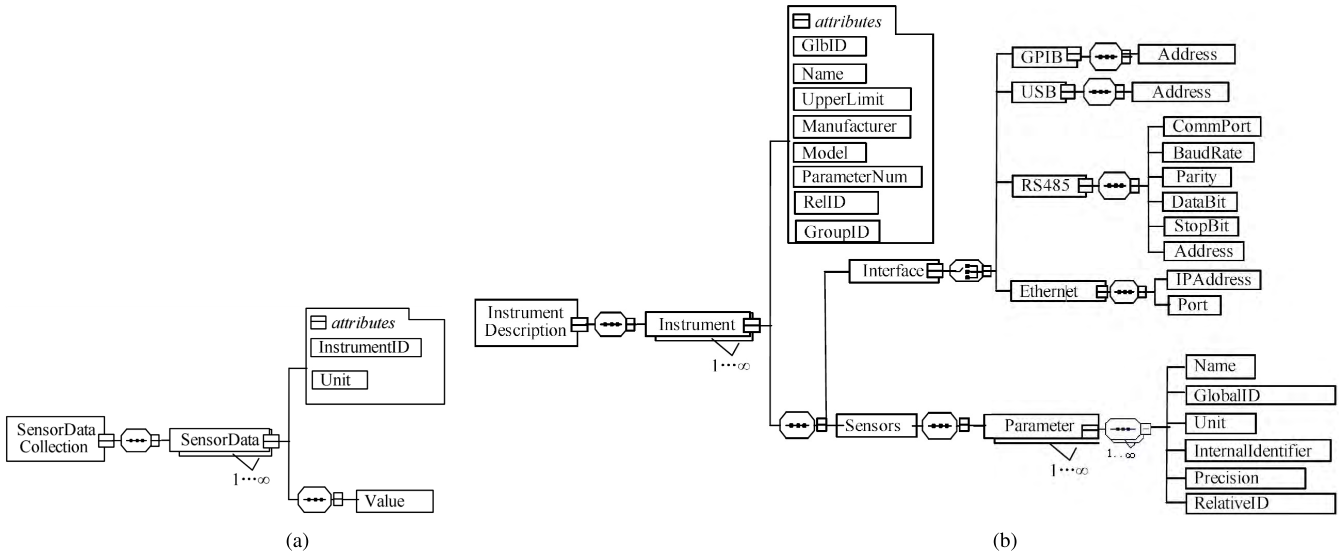


FIGURE 8. XML Schema of Data Interaction. (a) Sensor Data Schema. (b) Instrument Description Schema.

interaction operations between client and database would be completed by web service in IEEE Standard P1851.

The center can not only provide data interfaces but also public interfaces for DLL transmission, because other systems might access and view the data of some CVIs. For example, a product testing management system generates a test report, and the tester needs to view the curves of the test data. The center could find the location of DLL which need to be used, and transmit the DLL to the management system. The system loads the DLL and combines the components with test data to realize the function of curve display. The registered interfaces could be searched by a search engine provided by registration center.

E. UNIVERSAL CLIENT

In [31], we combined several complex virtual instruments to generate a CVIS client. In order to seamless integrate all kinds of monitoring systems, the bloatware client needs to package many kinds of CVI components, which brings negative effect on system performance. So we design a brand new light universal client which can dynamically schedule different CVI through loading DLL during monitoring.

After launching the client, the client provides a search engine which could search all the accessible interfaces. The user can select any CVI that he wants to view. The Graphic User Interface (GUI) is shown in Fig.9. The user has the authority to manage the interfaces which he would like to share with other systems.

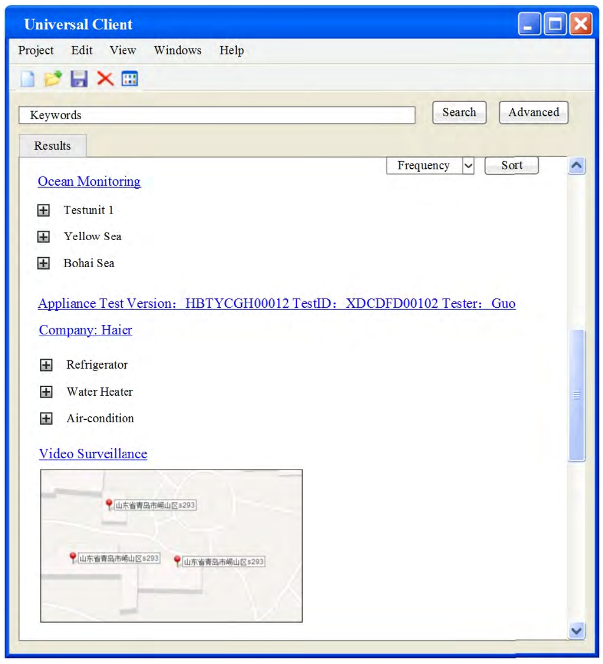


FIGURE 9. GUI of Universal Client.

V. IMPLEMENTATION PROCEDURE

The developing procedure of a common CVIS is shown in Fig.10. First, the developer creates the system interfaces through components drag-and-drop operations based on requirements, and then sets some configuration information including the window information, the subwindow information, the coordinate parameters, mapping information and so on. Second, the hardware information will be set in simulation subsystem to pretend real devices to realize system debugging. Third, after debugging, the configuration subsystem packs the XML configuration file and the DLL which will be used in CVIS. The package would be downloaded by users, meanwhile, all the web service interfaces will be registered to registration center. The shared DLL should also stored in central server.

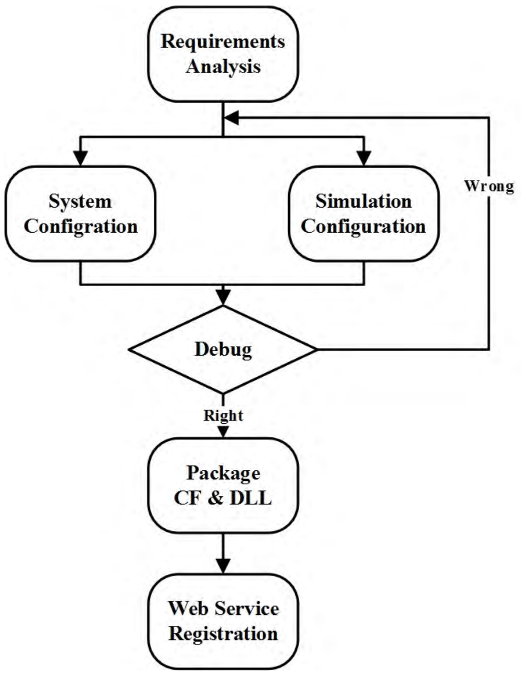


FIGURE 10. Developing Procedure.

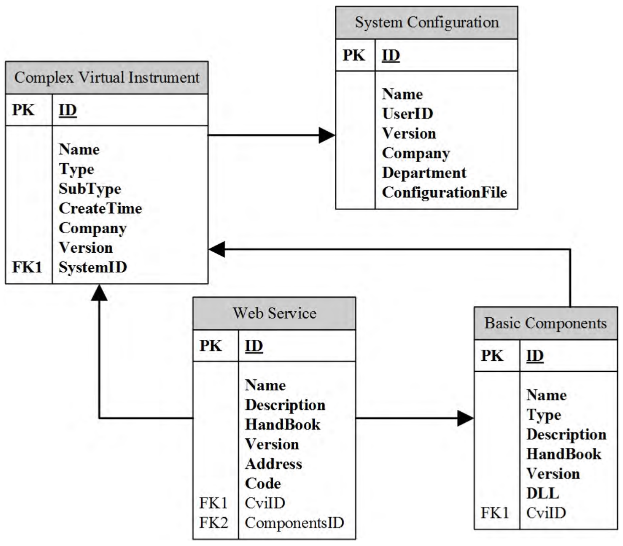
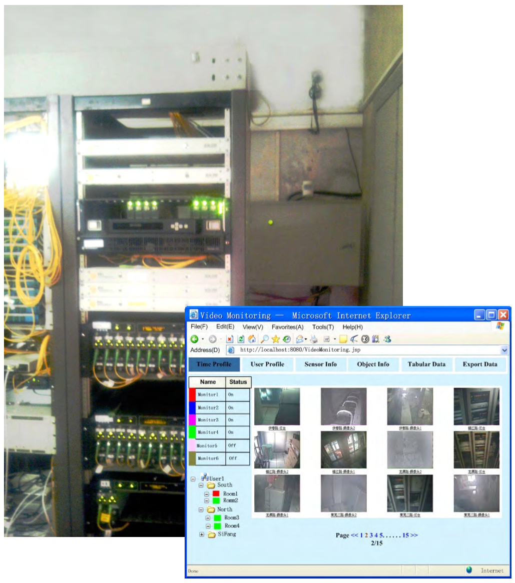
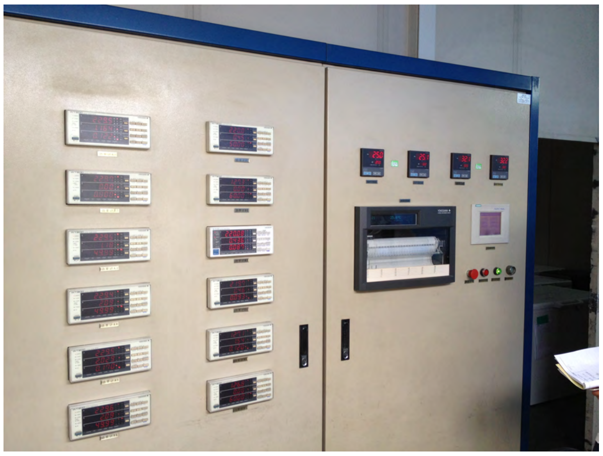


FIGURE 11. Part of Database Schema of CVI.

The client work flow is as follows. First, the user uses the search engine embedded in the client to search the CVI he



(a)



(b)

FIGURE 12. System Environments. (a) Video Surveillance System. (b) Appliance Test System.

would like to view. Second, the client access the data from database through web service, and the related DLL will be loaded in the client through mapping model. Third, the client finishes the component initialization and displays the data results into curves, video and so on. The database schema used in components loading is described in Fig.11. Different CVI can be dynamically scheduled by the universal client, which will replace the work of system integration.

Especially, the shared CVI web service interfaces can also be accessed by other system. The scheduling procedure is similar with the client. If another system would like to view the data of one CVIS, the DLL mapped with the data could be downloaded from the central server to combine with the data. We have already embedded this function into our measurement-based product testing management system [32]. It could dynamically view the graphic test data in test reports.

(a)

(b)

FIGURE 13. System Environments. (a) Ocean Monitoring System. (b) SUGON High Performance Cluster.

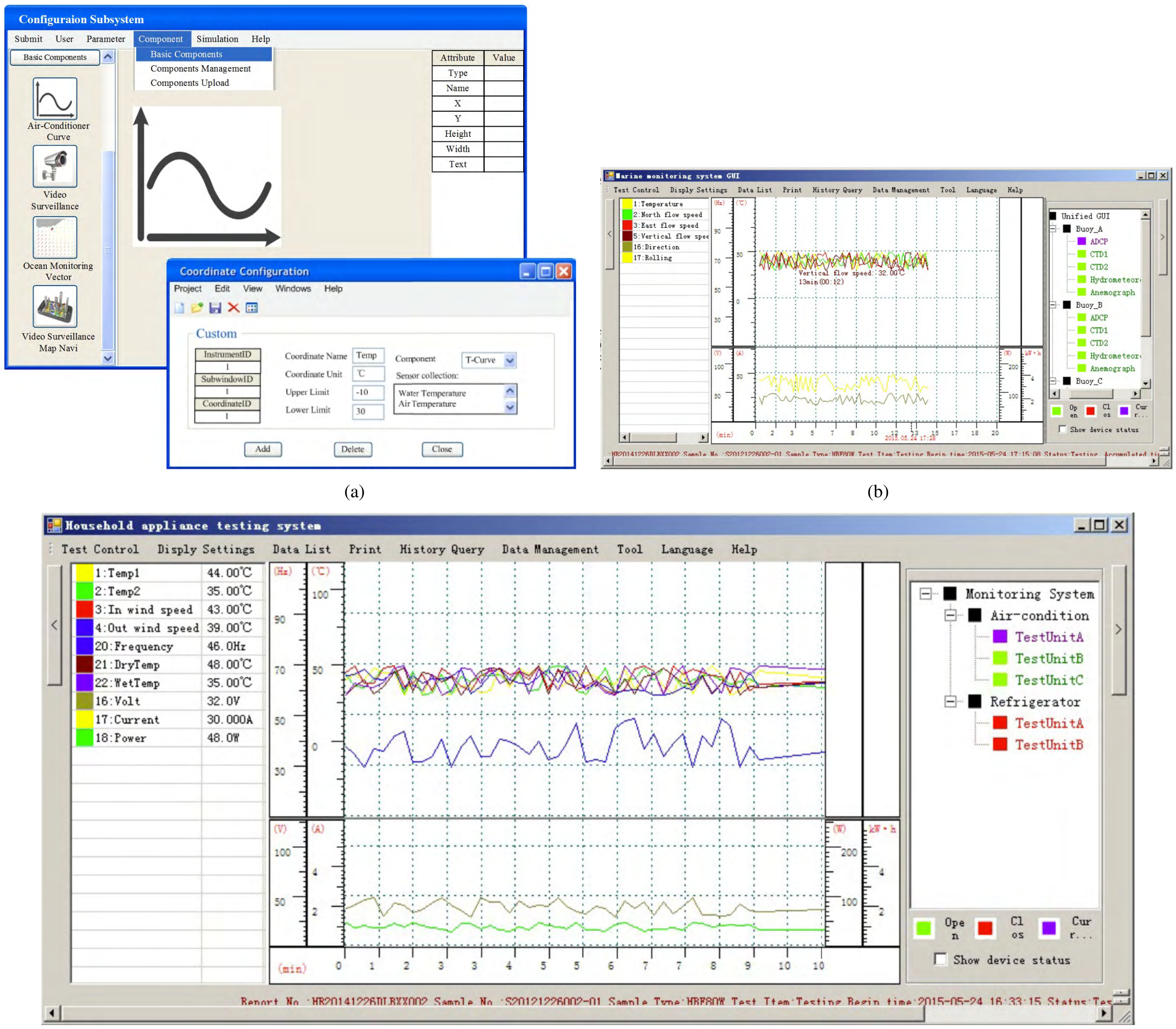
VI. CASE STUDY

Based on our architecture, we reconstruct three existing sensor-based systems which are all developed by our team to valid our architecture in one week. In this section, we will introduce detail information about our CVIS.

A. HARDWARE PLATFORM

The first system is video surveillance system. It employs two remote cameras to monitor the situation of the whole room. Meanwhile, three smoke detector sensors are equipped in this room, which can detect the real-time smoke density. All the devices are connected with computer via ethernet. Figure.12 (a) shows the surveillance environment and devices.

The second one is household appliance testing system, which contains 20 temperature sensors of Type T series, 32 power meters and one hybrid recorder. All the devices are used to monitor the statement of the household appliance during testing. The Type T sensors and power meters are



🟨	1:Temp1	44.00℃
🟩	2:Temp2	35.00℃
🟥	3:In wind speed	43.00℃
🟦	4:Out wind speed	39.00℃
🟪	20:Frequency	46.0Hz
🟫	21:DryTemp	48.00℃
🟪	22:WetTemp	35.00℃
🟫	16:Volt	32.0V
🟨	17:Current	30.000A
🟩	18:Power	48.0W

Report No.:HR20141226DLBXX002 Sample No.:S20121226002-01 Sample Type:HBF80W Test Item:Testing Begin time:2015-05-24 16:33:15 Status:Tes

FIGURE 14. GUI of CVIS Software. (a) Configuration Subsystem. (b) Loaded Universal Client(Ocean Monitering). (c) Loaded Universal Client (Appliance Testing).

connected with DA100 data collection device with RS-485. The DA100 and the central server are communicated in ethernet. The hybrid recorder is connected with the server via the GPIB ports. Figure.12 (b) shows the monitoring devices and some instruments.

The third one is based on our ocean monitoring system [33]. We extends our Ocean Sense project by adding ten underwater sensors nodes to monitor the underwater environment. Eighteen overwater nodes can communicate with the underwater sensors in ultrasonic. Two sensor nodes on the seashore are used to transmit the packets from the ocean nodes to the base station by muti-hop. The base station is connected with central server in ethernet. Figure.13 (a) shows the two off-shore relaying nodes and some overwater nodes.

The registration center with Oracle 10g and our cloud computing platform of CVIS are deployed on SUGON TC4600 high performance cluster which is shown in Fig.13 (b) A Thinkpad T60 laptop is used to run our universal client and management system.

B. CVIS SOFTWARE

Based on the hardware platform, we have reconstructed three systems through our configuration subsystem shown in Fig.14 (a). After developing, all the web service interfaces are registered in center. The configuration file are saved in XML on client computer.

After selecting the searched result, we open these three systems. Figure.14 (b) shows the graphic user interfaces of household appliance testing. Several kinds of data are shown in it, dry bulb, temperature, power and current. Through this interface, users could not only access the data result, but also related information about information retrieval, sensors and monitoring objects. Through the tabs on the top of the panel, we can switch the opened CVI. Figure.14 (c) is shows the ocean monitoring system which can access the flow, temperature and some other feature of the ocean. Through the universal client, we can easily access different kinds of CVIS by loading different CVI components.

VII. CONCLUSION AND FUTURE WORK

Based on the experience of developing sensor-based system, we have acquired the common characters of sensor-based systems. Then, we propose an platform based on CVIS architecture which is flexible and universal. The CVIS system can be built by drag-and-drop operations. The universal client can dynamically schedule the complex virtual instruments by loading there DLL, which replacing the system integration procedure. The reconstruction of the three reconstructed systems proves that our platform and architecture is effective and feasible. The tentative observation of this architecture promises that it could adapt to other situations.

In this paper, we provide a well-defined platform for sensor-based systems in which the users can develop and access heterogeneous in a unified cloud platform. Nevertheless, many issues still remain to be explored. Our ongoing works are: (1) Expanded this platform to other domain; (2) Enrich the component library; (3) Conclude the component development standard which could let third-party developer to uploaded their work.

REFERENCES

[1] Y. Kim, R. G. Evans, and W. M. Iversen, "Remote sensing and control of an irrigation system using a distributed wireless sensor network," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 7, pp. 1379–1387, Jul. 2008.

[2] A. Carullo, S. Corbellini, M. Parvis, and A. Vallan, "A wireless sensor network for cold-chain monitoring," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 5, pp. 1405–1411, May 2009.

[3] A. M. Abdelgawad and M. A. Bayoumi, "Remote measuring for sand in pipelines using wireless sensor network," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 4, pp. 1443–1452, Apr. 2011.

[4] K. Lee, D. Murray, D. Hughes, and W. Joosen, "Extending sensor networks into the cloud using amazon Web services," in *Proc. IEEE Int. Conf. Netw. Embedded Syst. Enterprise Appl. (NESEA)*, Nov. 2010, pp. 1–7.

[5] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 34–40, Jun. 2004.

[6] C. Otto, A. Milenkovic, C. Sanders, and E. Jovanov, "System architecture of a wireless body area sensor network for ubiquitous health monitoring," *J. Mobile Multimedia*, vol. 4, no. 1, pp. 307–326, 2006.

[7] R. Sen *et al.*, "Kyun queue: A sensor network system to monitor road traffic queues," in *Proc. 10th ACM Conf. Embedded Netw. Sensor Syst. ACM*, 2012, pp. 127–140.

[8] M. J. Chae, H. S. Yoo, J. Y. Kim, and M. Y. Cho, "Development of a wireless sensor network system for suspension bridge health monitoring," *Autom. Construct.*, vol. 21, pp. 237–252, Jan. 2012.

[9] L. Cristaldi, A. Ferrero, and V. Piuri, "Programmable instrument, virtual instrument, and distributed measurement system: what is really useful, innovative and technically sound?" *IEEE Instrum. Meas. Mag.*, vol. 2, no. 3, pp. 10–13, Dec. 2000.

[10] Y. Liu *et al.*, "Does wireless sensor network scale? A measurement study on greenorbs," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 873–881.

[11] S. S. M. Ali, B. George, L. Vanajakshi, V. Jayashankar, and V. J. Kumar, "A multiple loop vehicle detection system for heterogeneous and lane-less traffic," in *Proc. IEEE I2MTC*, Hangzhou, China, May 2011, pp. 1–5.

[12] O. A. Postolache, J. M. D. Pereira, and P. M. B. S. Girao, "Smart sensors network for air quality monitoring applications," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 9, pp. 3253–3262, Sep. 2009.

[13] J. J. G. de la Rosa, A. Agüera-Pérez, J. C. Palomares-Salas, J. M. Sierra-Fernández, and A. Moreno-Muñoz, "A novel virtual instrument for power quality surveillance based in higher-order statistics and case-based reasoning," *Measurement*, vol. 45, no. 7, pp. 1824–1835, Aug. 2012.

[14] J. Wang, X. Gao, and Q. Gao, "The design of wireless testing system for concrete rigidity based on virtual instrument," in *Proc. CECNet*, 2012, pp. 2968–2971.

[15] I. Orovic, "A virtual instrument for time-frequency analysis of signals with highly nonstationary instantaneous frequency," *IEEE Trans. Instrum. Meas.* vol. 60, no. 3, pp. 791–803, Mar. 2011.

[16] J. Lin, W. Xiao, F. L. Lewis, and L. Xie, "Energy-efficient distributed adaptive multisensor scheduling for target tracking in wireless sensor networks," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 6, pp. 1886–1896, Jun. 2009.

[17] R. X. Gao and Z. Fan, "Architectural design of a sensory node controller for optimized energy utilization in sensor networks," *IEEE Trans. Instrum. Meas.*, vol. 55, no. 2, pp. 415–428, Apr. 2006.

[18] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embedded Comput. Syst.* , vol. 6, no. 4, pp. 1886–1896, Sep. 2007.

[19] Z. Guo, P. Chen, H. Zhang, M. Jiang, and C. Li, "IMA: An integrated monitoring architecture with sensor networks," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 5, pp. 1287–1295, May 2012.

[20] E. Song and K. Lee, "Service-oriented sensor data interoperability for IEEE 1451 smart transducers," in *Proc. I2MTC*, Singapore, May 2009, pp. 1043–1048.

[21] M. Bertocco, S. Cappellazzo, A. Carullo, and M. Parvis, "Virtual environment for fast development of distributed measurement applications," *IEEE Trans. Instrum. Meas.*, vol. 52, no. 3, pp. 681–685, Jun. 2003.

[22] W. Winiecki and M. Karkowski, "A new Java-based software environment for distributed measuring systems design," *IEEE Trans. Instrum. Meas.*, vol. 51, no. 6, pp. 1340–1346, Dec. 2002.

[23] S. Jaskó and G. Simon, "CSP-based sensor network architecture for reconfigurable measurement systems," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 6, pp. 2104–2117, Jun. 2011.

[24] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generat. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[25] A. Weiss, "Computing in the Clouds," *Networker*, vol. 11, no. 4, pp. 16–25, Dec. 2007.

[26] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *Proc. 10th Int. Symp. Pervasive Syst. Algorithms, Netw. (ISPAN)*, 2009, pp. 4–16.

[27] R. Mietzner, T. Unger, and F. Leymann, "Cafe: A generic configurable customizable composite cloud application framework," in *On the Move to Meaningful Internet Systems*. Berlin, Germany: Springer, 2009, pp. 357–364.

[28] D. Chen and H. Zhao, "Data security and privacy protection issues in cloud computing," in *Proc. Int. Conf. Comput. Sci. Electron. Eng. (ICCSEE)*, vol. 1. Mar. 2012, pp. 647–651.

[29] C. Wang, Q. Wang, K. Ren, and W. Lou "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. IEEE, INFOCOM*, Mar. 2010, pp. 1–9.

[30] Z. Guo and H. Zhang, "Standard for design criteria of integrated sensor-based test applications for household appliances," IEEE Standards 1851-2012, Aug. 2012.

[31] Z. Guo, C. Liu, X. Wang, H. Ma, and Y. Jiang, "CVIS: Complex virtual instruments system architecture," in *Proc. I2MTC*, Minneapolis, MN, USA, May 2013, pp. 534–539.

[32] Y. Yu, Z. Guo, Z. Sun, and Y. Lu, "An integrated application framework for measurement-based product testing management," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, May 2012, pp. 2758–2763.

[33] J. Mingxing, Z. Guo, F. Hong, Y. Ma, and H. Luo, "OceanSense: A practical wireless sensor network on the surface of the sea," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2009, pp. 1–5.

[34] "Standard design criteria of complex virtual instruments for ocean observation," IEEE Standards ISO/AWI 21851, 2017.

CHAO LIU was born in China in 1988. He received the Ph.D. degree from the Ocean University of China, Qingdao, China, in 2016.

He is currently an Assistant Professor with the Department of Computer Science and Technology, Ocean University of China. His main research interests are sensor networks, distributed measurement systems, cloud computing, and so on.

ZHONGWEN GUO was born in China in 1965. He received the B.S. degree in computer science and technology from Tongji University, Shanghai, China, in 1987, and the M.S. and Ph.D. degrees from the Ocean University of China, Qingdao, China.

He is currently a Professor and a Doctoral Advisor with the Department of Computer Science and Technology, Ocean University of China. His main research interests are sensor networks, distributed measurement systems, ocean monitoring, and so on.

YUAN FENG was born in China in 1978. He received the Ph.D. degree from the Ocean University of China, Qingdao, China, in 2008.

He is currently an Associate Professor with the Department of Information Science and Engineering, Ocean University of China. His main research interests are sensor networks, cloud computing, and so on.

FENG HONG (M'11) was born in China in 1977. He received the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 2006.

He is currently a Professor with the Department of Information Science and Engineering, Ocean University of China, Qingdao, China. His main research interests are sensor networks, mobile computing, and so on.

WEI JING was born in China in 1989. He received the B.S. degree in computer science and technology from the Ocean University of China, Qingdao, China, in 2013. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Ocean University of China.

His main research interests are sensor networks, distributed measurement systems, cloud computing, and so on.