International Journal of Control and Automation
Vol. 10, No. 1 (2017), pp.99-112
http//dx.doi.org/10.14257/ijca.2017.10.1.09

# Research on Self-adaptive Methods of Sensor Data Interface

Chunshan Shen[1], Jun Jiao[1], Huimin Ma[1] and Shuqing Wang[2]

[1] *School of Information & Computer, Anhui Agricultural University, Hefei, Anhui 230031, China*
[2]*Hefei Information Technology University, Hefei, Anhui 230601, China*
*csshen@ustc.edu*

***Abstract***

*The sensor system is facing problems of the diversity of data acquisition and processing, which leading to some negative characteristics, such as low level of software reusing, high cost of system integration and maintenance. Therefore, new software design methods of interface between sensor application system and sensors are needed to reduce negative effects of sensor data interface diversity. The aim of our study was to explore related issues of software dynamic adaption while connecting sensors to sensor application system. Firstly, a new sensor data model was designed to describe some important parameters and attributes of sensors. Based on the sensor data model, a kind of self-adaptive software component of sensor accessing was able to provide static unified interfaces for high-level application software and dynamic adaptive interface for low-level driver software. Results indicated that the sensor data model and the software component of sensor access can solve the issue of sensor data interface diversity to some extent. Reasonably, the self-adaptive methods of sensor data interface based on the model and the component can improve software reusability, interface standardization, and large-scale using of sensor application system.*

***Keywords****: Sensor data model; Sensor data interface; Self-adaptive software*

## 1. Introduction

Software of sensor application system are becoming more complex, and giving high requirements for openness, dynamic and interoperability, with the rapid development of real-time and embedded system applications. However, the actual problem is that sensor application system usually meets specific problems and shows negative characteristics, such as the diversity of data acquisition and processing, low level of software reusing, high cost of system integration and maintenance. With the growth and depth of application, the complexity and rising costs of sensor application system are giving technical and economic difficulties to equipment suppliers and their users.

It is caused by many reasons, but the diversity of sensor interfaces is particularly important that could lead to many other issues including high cost of sensor access, *etc*. Usually, sensor interface means a set of software and hardware protocol specifications. The hardware part provides mechanical and electrical connection agreements, while the software part provides static data formats and dynamic interaction protocols for software integration. The static data formats and dynamic interaction protocols are defined as sensor data interface. The article focuses on solving the software complexity caused by the diversity of sensor data interfaces, but no consideration to mechanical and electrical connection parts, such as the reconfiguration design of hardware interface based on FPGA[1, 2].Some potential significance will be produced if reducing or eliminating the negative effect of the diversity of sensor data interfaces and realizing dynamic self-adaption between sensor application system and sensor data interfaces. For example, the

ISSN: 2005-4297 IJCA

software complexity and cost of system integration and maintenance can be reduced, and software reusability of sensor application system can be improved.

Currently, the main way to solve the diversity problem of sensor data interface was to develop interface standards from the perspective of sensors. Generally, these standards specified data formats and interactive protocols of sensors, such as IEEE P1451[3]、OGC SWE[4]、SpaceWire-PnP[5]、OSIS[6]、PSI5[7]、ASI、NTCIP[8]、CCSI[9]、FSIS[10], and so on. But the above way has set trade barriers and limited the scope of application in different degree, which led to application restriction of sensors without standard interfaces. For example, IEEE P1451.1, a kind of interoperability protocol between sensor nodes or smart sensors, specified a set of standard networked API functions for sensor node communication independent of any network[3, 11]. Just like UCA, OPC, and so on, both communication sides using these API had to maintain complex stack and middleware. And also, other measurement and control systems using the standard must be rebuilt with high cost because it is a new standard. In fact, there were few development platforms supporting the standard, such as Labview, JDDAC (Java Distributed Data Acquisition and Control) [12] and so on. Actual large-scale applications were not so many, but there were some demonstration projects led by IEEE 1451 and OGC SWE members, for instance, the sensor network projects [13, 14] developed by ORNL, Kansas, *etc*. Interestingly, a model used to convert special sensor data interface to IEEE 1451 NCAP protocol was developed in the sensor network project of Kansa, which was viewed as a main difficulty by the developers. A fact is that there were no large-scale emergence of profitable companies specializing in the production of STIM and NCAP models.

The above sensor interface standards often focused on sensor data description. Therefore, a few information frameworks were designed to describe sensor data and support interoperation between sensors or sensor nodes. Through the information framework, that is a kind of sensor data description model, users could know how information was designed, and how to extract interest information from raw data, which could do a good job for unified sensor data interface. For instance, IEEE 1451 TEDS described key parameters of sensor in detail. OGC SWE defined SensorML and TransducerML based on XML, which is a kind of static information model of sensor data [15-17]. A kind of sensor ontology model was designed by Russomanno to heighten the ability of sensors interoperation and integration in a network environment [18]. A sensor node model was proposed in paper [19] which tried to capture the most important features of sensor nodes, such as function, status and so on. The IrisNet Project developed by Intel and Carnegie Mellon University was to design a wide-range sensor service system. The system used the method of XML data schema to define sensor data by which data and service query could be done in a wider range [20]. Most of these sensor data models defined attributes and contents of sensor data in the way of Markup Schema Model [21]. They addressed mainly static description of characteristic parameters of sensors or software architecture of sensor nodes. But users had to deal with different sensor data interfaces with high cost while design sensor application system.

Why not the other way? The most common way to solve interface issue is that both equal interface sides follow a kind of specification or interface standard. But one side can adapt dynamically interfaces of the other, such as self-adaptive software which can adjust itself according to external running environment [22, 23]. For instance, the Carnegie Mellon Navigation (CARMEN) Toolkit defined a set of normal function interfaces for information acquisition [24]. The article [25] applied Feedback control and artificial intelligence theory to distributed software components of sensor network that could adapt change of external environment through runtime reconfiguration. Now there have been various self-adaptive methods and technologies on wireless sensor network, such as self-organized network, autonomic computing unit, dynamic data fusion, software rejuvenation, multi-agent sensor network and so on[26-30]. However, these methods

 

mainly focused on reduction of software complexity of upper application layer while ignoring lower sensor data interface.

The specificity of sensor application system led to the inevitability of diverse sensor data interfaces. The most promising way to solve the diversity of sensor data interface is to realize software self-adaption of sensor data interface. Therefore, this article addressed a self-adaptive method of sensor data interface. Software self-adaption methods have different characteristics, and can be classed from the size and cost of software adjustment, realization issues, temporal characteristics, and interaction concerns [23]. Taking into account the characteristics of sensor application system, such as limit of software/hardware resources, human involvement, the self-adaption method of sensor data interface was based on little size adjustment, external approach, model-related, and reduction of human involvement. The method focused on sensor data model for sensing external environment, self-adaptive software component called as engine. The engine provided static unified interfaces for high-level application software and dynamic adaptive interfaces for low-level sensor data interface driver software. The conversion between the static interfaces and dynamic interfaces was supported by the sensor data model.

The above idea avoided using of mandatory standards, and reduced the impact on industry and application. In that way, sensor application system could separate upper information-processing logic from diverse sensor data interfaces, which could reduce cost of sensor application system integration more easily, and improve software reuse process of sensor application system.

In the remainder of the paper, section 2 presented self-adaptive software architecture of sensor data interface running in an embedded sensor node. Section 3 described how the sensor data model was designed and section 4 was about self-adaptive software component to access sensor data interface. Section 5 described how the self-adaptive method was applied into a sensor node accessing a sensor data interface.

## 2. Self-adaptive Software Architecture of Sensor Data Interface

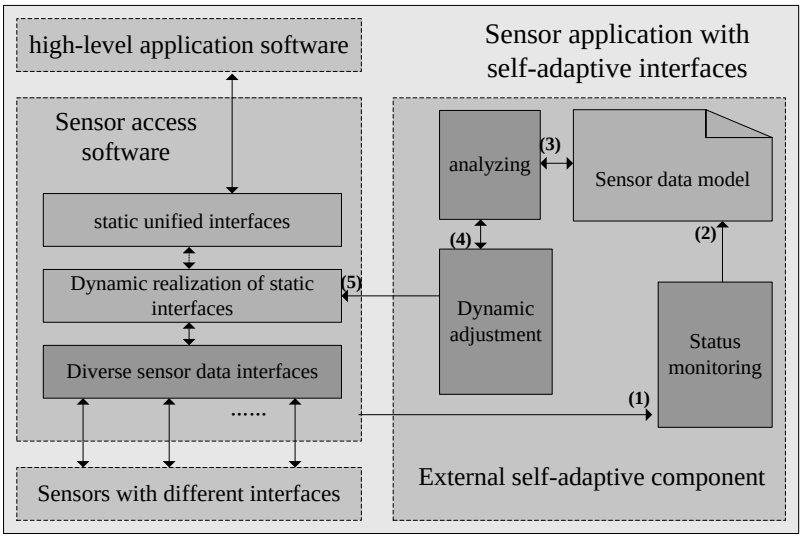


**Figure 1. Self-Adaptive Software Architecture of Sensor Data Interface**

Our approach is illustrated in Figure 1. Step (1): The connecting status could be founded by model of status monitoring while a sensor connected to the sensor application system. Step (2): To select and configure the proper sensor data model file according to the connected sensor. Step (3): To determine how to adjust for accessing the connect

sensor. Step (4): to adjust itself for adapting to external change. Step (5): High-level application software could access the connect sensor through static unified interfaces which have been transformed to special adaptive interfaces. The approach is driven by model, called sensor data model. Therefore, the sensor data model is the important part in our approach and we presented how to establish the sensor data model in detail in section 3.

## 3. Design of Sensor Data Model

### 3.1. Definition of Sensor Data Model

At the beginning, it is necessary to give a detail definition for sensor data model. It was defined as following.

**Sensor Data Model**: A set of rules of information description used to describe characteristic parameters, data format, and signal processing of sensor or sensor node.

Sensor characteristic parameters are the basic description on physical features of sensors, which is the basis for use of the sensor. Description contents include a global unique identification number, channel number, channel type, channel sampling period, the maximum transfer rate, measurement range, data encoding format, actual basic protocol for data transmission, calibration polynomial function and so on. The describing way should refer to existing international standards and be compatible with the existing protocols.

Data format refers to the way sensors represent and store the data obtained from the environment. It is an effective way to help application system to acquire, transfer and archive sensor data, without caring about the original data format. Here, the sensor data could be data frame that described the status of environment, and be also control commands that described the expectation of environment. Particularly, a normal description of observation and measurement data was important for terminal users to acquire sensor information effectively.

Sensor signal processing was the process of perception of environmental information. Signal processing should be defined through a standard format, unified interface language and information model, which could be helpful to rapid development of environment sensing system. Usually, these signal processing was described in sensor data model in the form of components and services.

### 3.2. Conceptual Model of Sensor Data

The steps of sensor data model designing included Conceptual model and description model. The conceptual model of sensor data was based on object-oriented method, which described characteristic parameter, data format, interactive protocols, signal processing, and running status. Concept modeling included terms obtaining, concept classification, explicitly of properties and constraints, consistency checking, and correction and evaluation process. The concept model of sensor data was completed through a kind of software tool with the name of Protégé, shown as Figure 2.

### 3.3. Description Model of Sensor Data

Concept model needed to be read and written by users, mainly including other application software. Therefore, the model should be expressed in a certain way which application software could understand easily. After some research, the XML was selected to describe the sensor data model. And also, a little software tool which can read and write the sensor data description file was developed. The root of description model was showed in Figure 3. Additionally, the sensor data model was abbreviated as SDM-XML.

 

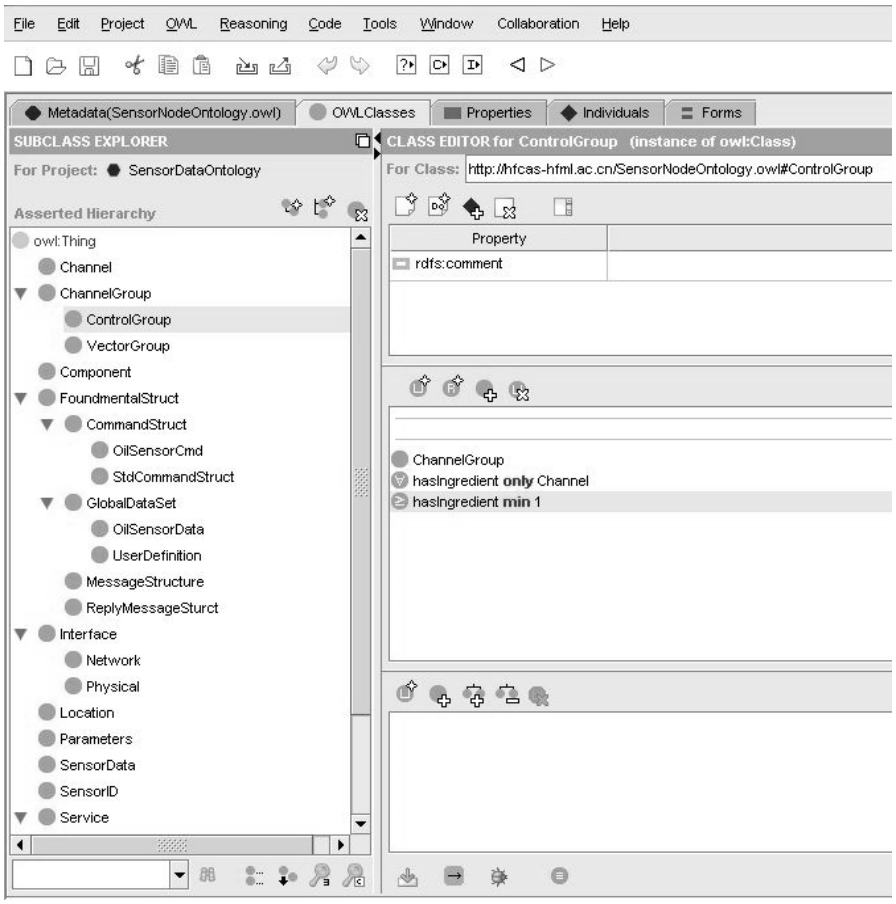


**Figure 2. Ontology of Sensor Node Designed In Protégé**

The main meaning of elements in Figure 3 was following. The element SensorID was used to identify the sensor node, and it consisted of number, manufacturer, production time and introduction. The element Status was used to record and show recent situation and events of sensor node. There were two kinds of data structure to save the status data. One was situation data used to show operation status, and the other was event data used to record events happened in sensor node. The element FoundmentalStruct defined the basic data format for message transferring, data storage, command sending among sensor nodes. The basic data formats included message structure, response structure, command structure and data frame. The element Parameters were some important data describing the basic features of sensor node, similar to IEEE 1451 TEDS. The parameter was packaged according to the order of length, data value, and checksum. And the last element Interface defined external functions of sensor node, which could be called by network users.

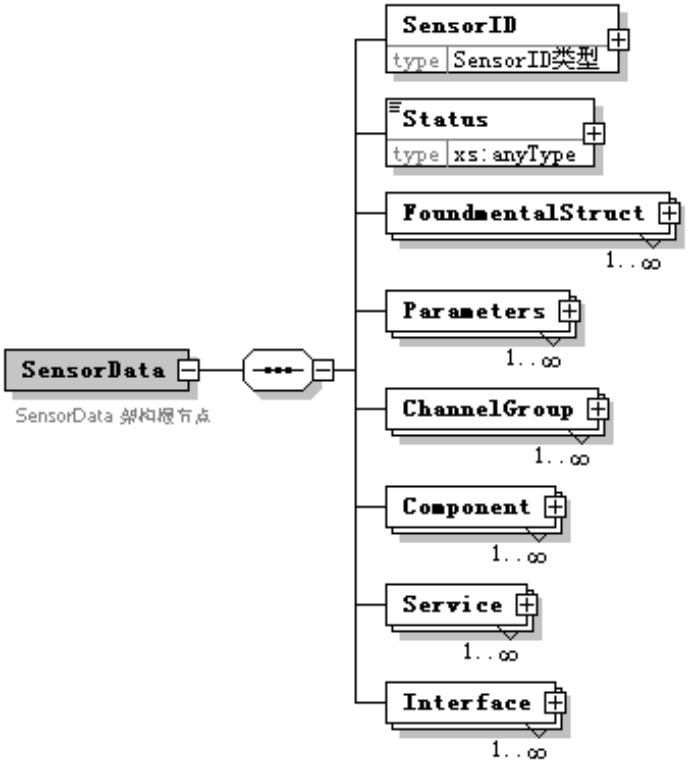


**Figure 3. Root of Description Model of Sensor Data (SDM-XML)**

## 4. Software Component to Access Sensor Data Interfaces

The self-adaptive software component is a basic part to realize dynamic interface to access sensor data. As discussed above, the self-adaptive component was driven by sensor data model, and was composed of three parts, interface status sensing, dynamic interface analyzing and reconfiguration. Run-time detection and tracking technologies were used to capture access and running statues of sensors of sensor node. The result of status sensing was abstracted into numerical which was associated to the property values of sensor data model. Based on the sensor data model, dynamic interfaces could be designed for diverse sensor data access.

An embedded software component "SensorCOM" has realized above function partially. It was based on an embedded development platform of ARM CPU and WinCE 5.0. SensorCOM provided two kinds of interfaces which were high-level application oriented unified static interface and low-level sensor oriented dynamic interface. In fact, there was only one kind of interface, *i.e*. the static unified interface. The dynamic interface to access different sensor was just a realization of static interface. The function of SensorCOM mainly focused on sensor access detection, configuration, addition, querying, and real-time access of sensor data.

There were seven objects in SensorCOM, showed in Figure 4. Each object provided a set of interfaces called by external application. For example, the object SensorManager provided the interface ISensor which was composed of some functions. These functions were listed partially in Table 1.

 

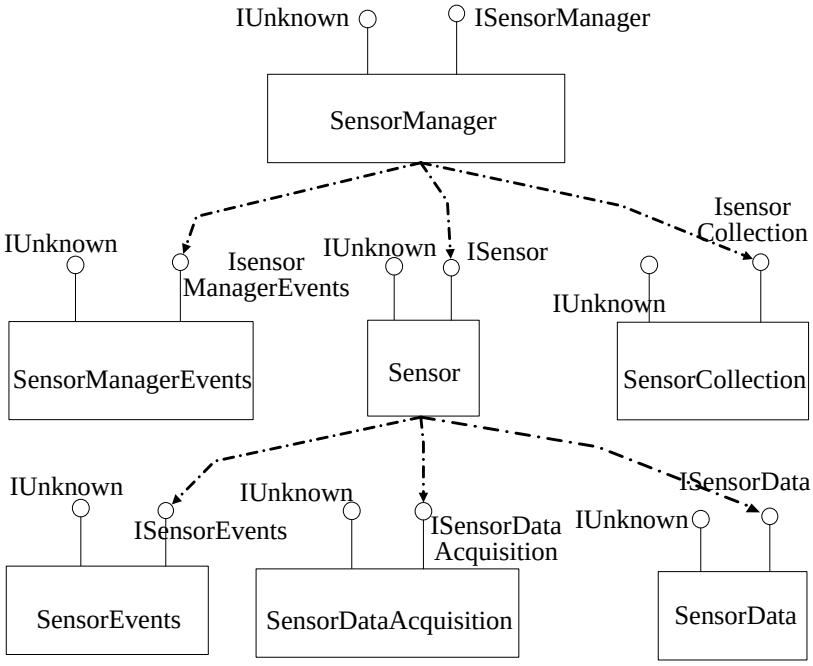


**Figure 4. The Functions of Sensorcom**

**Table 1. Partial Functions of Isensor**

| Function name | description |
|---|---|
| GetID | To get global unique identifier. |
| GetCommand | To get a particular command data of sensor |
| GetData | To get latest sensing data |
| GetType | To get the type of sensor |
| GetProperty | To get a specified property of sensor |
| GetProperties | To get a group of properties of sensor |
| GetState | To get the status of sensor |
| GetEventInterest | To get subscribed events |
| SetProperty | To configure a specified property |
| SetProperties | To configure a group of properties |
| SetEventInterest | To subscribe events of sensor |

## 5. Application of Self-Adaptive Methods Of Sensor Data Interface

### 5.1. Problem Description

Traditionally, it is needed to design data acquisition and processing model for each sensor of multi-sensor system integration. When a sensor is changed, or has new requirement change, developers often spend a lot of effort to transplant or modify code to meet new needs. In this way, it may be led to much redundant code generation, high development cost, and lower reliability in the absence of a unified standard framework supporting. Usually, sensor networks, composed of the above sensor node, were applied into specific domain, with a certain degree of interoperability, is expensive and poor flexibility, difficult to maintain and extend.

Interestingly, the above discussed self-adaptive method of sensor data interface base on SDM-XML and software component of interface access could solve the problem in a certain degree. Speaking further, the software development specification of sensor

application system integration could appear with the support of standard sensor data model, just like the above SDM-XML. And also, the specification could improve software reusability, system interoperability, and reduce system development costs. SDM-XML is useful to provide developers with a single data source and a single application program interface. Of course, SDM-XML should be parsed by some kind of common software, such as the self-adaptive interface component described in Section 4. Through the approach with SDM-XML and self-adaptive interface component, it is not necessary for software developers to develop specific program for each sensor. In the following, the data acquisition program of a multi-function pressure sensor "PressSensor" was illustrated to describe how the self-adaptive method was used to improve its reusability.

### 5.2. Application Example

It was stated in the example that the modified data acquisition program of "PressSensor" could easily get and send commands based on its SDM-XML file and interface component.

PressSensor was composed mainly of the main processing unit (MSC1210), RS232 communication interface, and four sensor channels which were pressure sensor one, pressure sensor two, a acceleration sensor, and a temperature sensor. It could be used widely for oil-filed device to calculate the oil amount of each stroke according to the pressure and acceleration data, while the temperature sensor could be helpful to temperature compensation for the bad oil-field environment.

When power on, PressSensor would send sensor data frame continuously after receiving the six-byte command data frame with the content of "0xFF, 0x7D, 0,0,0,0" illustrated in Table 2. And the returning data frame of PressSensor was showed in Table 3.

**Table 2. Command Data Frame Format of PressSensor**

| **1-Octet** | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Package header, and fixed value was 0xFF | | | | | | | |
| Command type, and value 0x7D indicated sending data frame without temperature information | | | | | | | |
| undefined | | | | | | | |
| undefined | | | | | | | |
| undefined | | | | | | | |
| undefined | | | | | | | |

**Table 3. One Data Frame of PressSensor**

| **1-Octet** | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Package header, and fixed value was 0xFA | | | | | | | |
| Package header, and fixed value was 0xAF | | | | | | | |
| Type identifier of data frame, and value 0xFF indicated sending data frame without temperature information | | | | | | | |
| High 8-bit pressure value (sum of two channels) | | | | | | | |
| Low 8-bit pressure value (sum of two channels) | | | | | | | |
| undefined | | | | | | | |
| undefined | | | | | | | |
| High 8-bit acceleration value | | | | | | | |
| Low 8-bit acceleration value | | | | | | | |
| undefined | | | | | | | |

The following was the code fragment that sent the command for acquiring sensor data frame without temperature information.

 

```
//to send command,   and acquire sensor data frame without temperature information.
……
unsigned char pc2msc_cmd[PC2MSC_CMD_NUM];
pc2msc_cmd[1]=125;
DWORD lpNumOfBytesWritten;
if(!m_comDev.Write(pc2msc_cmd,PC2MSC_CMD_NUM,&lpNumOfBytesWritten))
……
```

Unfortunately, there were many locations like the above code fragment in the data acquisition program of PressSensor. It is required for developers to be familiar with the command format of PressSensor while coding, and have to modify all locations while a command format changed. Obviously, the code was poor reusability and difficult to maintain and extend.

To solve the above problem, the SDM-XML and SensorCOM were introduced. An interface function GetCommand was defined to generate command data frame based on SDM-XML file of PressSensor. The function GetCommand was packaged into the interface ISensor of SensorCOM. In this way, needed command data could be generated dynamically while calling GetCommand with necessary parameters. Now, the above code fragment could be improved into the following form.

```
//to generate command data frame based on SDM-XML and SensorCOM, and acquire sensor data frame without temperature information.
……
pSensor.GetCommand("Oilsensor001.xml",
      "OilSensorCmd", "StartSampling", & command);
if(!m_comDev.Write(command.aContent, command.length,&lpNumOfBytesWritten))
……
```

Fortunately, developers would only modify the sensor data model file of PressSensor called as “Oilsensor001.xml” partly showed in appendix 1 while the command format changed. And the complexcity involving data acquisition program of PressSensor would be greatly reduced. Evidently, the method based on SDM-XML and SensorCOM could simplify the developer's work and improve the program reusability and system interoperability. The Oilsensor001.xml was the SDM-XML file, a kind of sensor data model file, describing multi-function pressure sensor PressSensor. Experimental results indicated that the method could improve the reusability of sensor application system.

## 6. Summary

The method of self-adaptive sensor data interface presented in this paper was to solve the problem of sensor data interface diversity. It separated information processing logic interested by users from the various sensor data interfaces and could avoid the impact and restriction by using mandatory standards of industries and applications. Through the application of the method, the cost and Complexity of sensor application system integration could be reduced, and software reuse process could be accelerated, which would promote the foundation of large scale application of sensor application system and standardization. We would extend the method in agricultural information system in our future work, and optimize continuously in order to create an industrial standard.

## Appendix 1 Oilsensor001.xml（part）：

```
<?xml version="1.0" encoding="UTF-8"?>
<SensorData xmlns="http://hfcas-hfml.ac.cn/SensorDataXMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
        <SensorID>
                <UUID>china-hcilab-200810103000-001</UUID>
                <Manufacture>China</Manufacture>
                <Time>2008-12-10T10:30:00</Time>
<Description>PressSensor </Description>
        </SensorID>
        <Status/>
        <FoundmentalStruct>
                <CommandStruct>
                        <OilSensorCmd>
                                <Octet0>0xFF</Octet0>
                                <Octet1>
                                        <SetZero>0x7F</SetZero>
                                </Octet1>
                                <Octet2/>
                                <Octet3/>
                                <Octet4/>
                                <Octet5/>
                        </OilSensorCmd>
                </CommandStruct>
        </FoundmentalStruct>
        <FoundmentalStruct>
                <CommandStruct>
                        <OilSensorCmd>
                                <Octet0>0xFF</Octet0>
                                <Octet1>
                                <StartSampling>0x7D</StartSampling>
                                </Octet1>
                                <Octet2/>
                                <Octet3/>
                                <Octet4/>
                                <Octet5/>
                        </OilSensorCmd>
                </CommandStruct>
        </FoundmentalStruct>
    <!-- …Omission… -->
        <FoundmentalStruct>
        <!-- …Omission… -->
        <FoundmentalStruct>
        <!-- …Omission… -->
        <Parameters>
                <Length>10</Length>
                <Data>
                        <Calibration>
                                <Identifier>
                                        <Name>2</Name>
                                        <Length>4</Length>
                                        <Value>
                                                <Class>
                                                <Calibration>2</Calibration>
                                                </Class>
                                                <Reserved1>0</Reserved1>
                                                <Reserved2>0</Reserved2>
                                                <Reserved3>0</Reserved3>
                                        </Value>
                                </Identifier>
                                <CalibrationPara CalibrationID="1">
                                        <Method>
                                                <Linear>1</Linear>
                                        </Method>
                                        <CoefficientList>
                                                <coefficient>
                                                <VaryC1>0.004218</VaryC1>
                                   <ConstC0>0.030056</ConstC0>
                                                </coefficient>
                                        </CoefficientList>
                                </CalibrationPara>
                        </Calibration>
```

 

```
                    </Data>
                    <CheckSum>0</CheckSum>
            </Parameters>
            <!-- …Omission… -->
            <ChannelGroup CalibrationID="1">
                    <ControlGroup>
<Channel isEmbededChannel="false" isEventChannel="false" domain="Electrical">
                                        <ChannelNumber>0</ChannelNumber>
                                        <Type>
                                                <Input>1</Input>
                                        </Type>
                                        <Mode>
<ModeClass>SamplingMode</ModeClass>
<ModeDescription>Continue</ModeDescription>
                                                <ValueAttribute>
                                                        <Name/>
                                                        <Value/>
                                                </ValueAttribute>
                                        </Mode>
                                        <PhysicalUnit>
                                                <kilograms/>
                                        </PhysicalUnit>
                                        <Range>
                                                <Lower>0</Lower>
                                                <Upper>2000</Upper>
                                        </Range>
                                        <TimePara>
                                        <SamplingPeriod>50</SamplingPeriod>
                                                <DataUpdate>50</DataUpdate>
                                        </TimePara>
                                </Channel>
                    </ControlGroup>
            </ChannelGroup>
            <ChannelGroup CalibrationID="1">
            <!-- …Omission… -->
            <ChannelGroup CalibrationID="1">
            <!-- …Omission… -->
            <Component>
            <!-- …Omission… -->
            <Service>
            <!-- …Omission… -->
            <Interface>
            <!-- …Omission… -->
</SensorData>
```

## Acknowledgements

We would like to thank the researchers of School of Information & Computer of Anhui Agricultural University for their valuable support and advice during designing the self-adaptive method of sensor data interface. We also wish to thank some members of Hefei High Magnetic Field Laboratory of China for their help in this paper.” “

## References

[1] Q. Chi, “A Reconfigurable Smart Sensor Interface for Industrial WSN in IoT Environment. Industrial Informatics”, IEEE Transactions on, vol. 10, no. 2, **(2014)**, pp. 1417 - 1425.
[2] G.J. García, “A Survey on FPGA-Based Sensor Systems: Towards Intelligent and Reconfigurable Low-Power Sensors for Computer Vision, Control and Signal Processing”, Sensors, vol. 14, no. 4, **(2014)**, pp. 6247-6278.
[3] (TC-9), T.C.o.S.T., “IEEE P1451.0 Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats”, IEEE Instrumentation and Measurement Society, **(2007)**.
[4] M. Botts, “OGC® Sensor Web Enablement: Overview And High Level Architecture”, Open Geospatial Consortium Inc., **(2007)**.
[5] P. Mendham, A.F. Florit and S. Parkes, “SpaceWire-PnP Protocol Definition”, Space Technology Centre School of Computing University of Dundee Dundee, DD1 4HN Scotland, UK, **(2009)**.
[6] “Optical Sensor Interface Standard”, http://www.ntb.ch/pub/bscw.cgi/d18647/OSIS WG2 Standard Documentation.pdf.

 

[7] Organization, P., "Peripheral Sensor Interface for Automotive Applications", http://www.psi5.org/en/pool/pdf/psi5_specification_v13_080729.pdf, July 31, **(2008)**.
[8] NTCIP, J.C.o.t., 9001 new version v04, The NTCIP Guide. **(2009)**.
[9] Team, J.-C.S.A., Common CBRN Sensor Interface http://www.jpeocbd.osd.mil, February 18, **(2008)**.
[10] G. Morrison, "FSIS ™ Ferrybox Sensor Interface Standard. International SeaKeepers Society", http://www.coaps.fsu.edu/RVSMDC/marine_workshop3/presentations/Morrisonposter.pdf.
[11] NIST, "IEEE Standards 1451 Family", IEEE Standards Department, http://www.ieee.com.
[12] "Java Distributed Data Acquisition and Control (JDDAC)", https://jddac.dev.java.net/.
[13] B.L. Gorman, "Advancing Sensor Web Interoperability. Sensors, http://www.sensorsmag.com/sensors/Homeland+Security/Advancing-Sensor-Web-Interoperability/ArticleStandard/Article/detail/185897, vol. 22, no. 4, **(2005)**, pp. 14–18.
[14] P. Mani, "Unified SensorNet Architecture with Multiple Owners An Implementation Report", The University of Kansas Technical Report, Oak Ridge National Laboratory, **(2010)**.
[15] M. Botts and A. Robin, "Sensor Model Language (SensorML)", Open Geospatial Consortium Inc., **(2007)**.
[16] S. Cox, "Observations and Measurements", Open Geospatial Consortium Inc., **(2007)**.
[17] S. Havens, "Transducer Markup Language (TML)", Open Geospatial Consortium Inc., **(2007)**.
[18] D.J. Russomanno, C. Kothari and O. Thomas, "Sensor Ontologies: From Shallow to Deep Models", Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory, 20-22 March, **(2005)**, pp. 107-112.
[19] S. Avancha, C. Patel and A. Joshi, "Ontology-driven Adaptive Sensor Networks", Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, **(2004)**, pp. p. 194- 202.
[20] P.B. Gibbons, "Irisnet: An architecture for a world-wide sensor web", IEEE Pervasive Computing, vol. 2, no. 4, **(2003)**, pp. 22-23.
[21] T. Strang and C. Linnhoff-Popien, "A Context Modeling Survey", In Proceedings of Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp. 2004, **(2004)**.
[22] P. Oreizy, "An Architecture-Based Approach to Self-Adaptive Software. IEEE Intel. Syst., vol. 14, no. 3, **(1999)**, 54–62..
[23] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges", ACM Transactions on Autonomous and Adaptive Systems, vol. 4, no. 2, **(2009)**.
[24] M. Montemerlo, N. Roy and S. Thrun, "Perspectives on Standardization in Mobile Robot Programming: The Carnegie Mellon Navigation (CARMEN) Toolkit", IEEE International Conference on Intelligent Robots and Systems, vol. 3, **(2003)**, pp. 2436-2441.
[25] D.M. Zhang, "EAAR: An approach to environment adaptive application reconfiguration in sensor network", in Mobile Ad-Hoc and Sensor Networks, Proceedings, X. Jia, J. Wu, and Y.X. He, Editors. **(2005)**, pp. 259-268.
[26] S. Hallsteinsen and R.T. Sanders, "Software architecture for self-adapting sub-sea sensor networks: Work in progress", Athens, Glyfada, Greece: IEEE Computer Society, **(2009)**.
[27] T.R.W. Scogland and W.-C. Feng, "Runtime Adaptation for Autonomic Heterogeneous Computing. Cluster, Cloud and Grid Computing (CCGrid)", 2014 14th IEEE/ACM International Symposium on, Chicago, IL, **(2014)**, pp. 562 - 565.
[28] D.Reissner, "Energy-efficient adaptive communication by preference-based routing and forecasting. Multi-Conference on Systems", Signals & Devices (SSD), 2014 11th International, Barcelona, **(2014)**, pp. 1 - 6.
[29] J. Guo, "Research of Software Multi-level Rejuvenation Method Based on Prediction", ICISEM '13 Proceedings of the 2013 International Conference on Information System and Engineering Management, IEEE Computer Society Washington, DC, USA, **(2013)**, p. 194-198.
[30] J.P. Mano, J.P. George and M.P. Gleizes, "Adaptive Multi-agent System for Multi-sensor Maritime Surveillance", in Advances in Practical Applications of Agents and Multiagent Systems, Y. Demazeau, *et al*., Editors, **(2010)**, pp. 285-290.

## Authors

**Chunshan Shen**, he was born in Anhui, china, in 1980. He received the M.S. degree and Ph.D. degree from Graduate University of Chinese Academy of Sciences in 2005 and 2010, respectively. Now he is senior engineer at School of Information & Computer, Anhui Agricultural University. His research interests are focused on Management and Control Integration, IT project management, *etc*.

 

**Jun Jiao**, he was born in Nanjing CIty, China,in 1964.He received the M.S. degree and Ph.D. degree from Anhui Agricultural University and HeFei University of Technology, respectively in 2003 and 2010.Now he is associate professor at School of Information and Computer, Anhui Agricultural University. His research interests are focused on Intelligent Control and Internet of things,*etc*.

**Huimin Ma**, she was born in Shanxi, china, in 1984. She received the Ph.D. degree from Graduate University of Chinese Academy of Sciences in 2012. Now she is a lecturer at School of Information & Computer, Anhui Agricultural University. She research interests are focused on intelligent information processing, intelligent algorithm, *etc*.

**Shuqin Wang**, she was born in Anhui, china, in 1980. She is a teacher of Hefei Information Technology University, China. Her research interests are in the software engineering and project management.