http://www.zju.edu.cn/jzus; E-mail: jzus@zju.edu.cn
ISSN 1009-3095 *Journal of Zhejiang University SCIENCE* V.4, No.5, P.542-548, Sep.-Oct., 2003

Distributed heterogeneous inspecting system and its middleware-based solution*

HUANG Li-can(黄理灿)†, WU Zhao-hui(吴朝晖), PAN Yun-he(潘云鹤)
(*College of Computer Science, Zhejiang University, Hangzhou 310027, China*)
†E-mail: lchuang@cs.zju.edu.cn

Received Dec.6, 2002; revision accepted Mar.8, 2003

Abstract: There are many cases when an organization needs to monitor the data and operations of its supervised departments, especially those departments which are not owned by this organization and are managed by their own information systems. Distributed Heterogeneous Inspecting System (DHIS) is the system an organization uses to monitor its supervised departments by inspecting their information systems. In DHIS, the inspected systems are generally distributed, heterogeneous, and constructed by different companies. DHIS has three key processes-abstracting core data sets and core operation sets, collecting these sets, and inspecting these collected sets. In this paper, we present the concept and mathematical definition of DHIS, a metadata method for solving the interoperability, a security strategy for data transferring, and a middleware-based solution of DHIS. We also describe an example of the inspecting system at WENZHOU custom.

Key words: Distributed computing, Middleware, Heterogeneous systems
Document code: A **CLC number:** TP316

INTRODUCTION

Inspecting service is widely used by an organization to inspect the operations and data of its supervised departments, especially those departments which are not owned by this organization.

There are many systems which have inspecting services, such as Enterprise Resource Planning (ERP), financial system, and so forth, but most of the subsystems in these systems are developed by the same companies, and have the same functions and database structure. However, Distributed Heterogeneous Inspecting System (DHIS) differs greatly from the above-mentioned systems. DHIS is something similar to monitoring system and network management system (Huang and Wu, 2001; 2002; Sidnie, 1995). But in DHIS monitored objects are software, not equipments.

In this paper, we propose the concept and definition of DHIS and present the methods for implementation of DHIS.

DEFINITION OF DHIS

Overview of DHIS

In DHIS the inspecting system collects the data of the inspected systems, (which are generally distributed, heterogeneous and constructed by different companies) and then analyzes and inspects those data and operations using methods such as parameter comparison, sequential analysis, data balancing, data integrity, data mining, and so on. DHIS has three key processes-abstracting the core data sets and core operation sets, collecting these sets, and inspecting these collected sets.

The DHIS architecture is a tree-like hierarchical structure. The leaf nodes are inspected systems, whose data and operations are inspected and audited by the upper-layer inspecting system. Upper nodes above inspecting systems are displaying systems, which get alarm data from

* Project supported by the National Hi-Tech Development Program(863) of China (No. 2001AA111271-2) and virtual cooperation research granted by Chinese Ministry of Education

the lower inspecting systems or lower displaying systems. The inspecting system communicates with all its inspected systems separately. The minimal requirements of the inspecting system in DHIS are as follows:

Collecting data

Inspecting data and operations

Alarming

Query functions and statistical analysis

DHIS is something similar to monitoring system and network management system. However, in DHIS, the monitored objects are software, not equipment, and DHIS is much more complex than the monitoring systems.

The main features of DHIS are:

(1) The inspected systems are distributed and heterogeneous computer systems. They are heterogeneous in hardware, networks, protocols, program languages, system functions, databases, and so on. They are distributed in geological locations, and they are network systems based on WAN.

(2) From the conceptual abstract point of view, the inspected systems have the same core conceptual data sets and same core conceptual operation flows. DHIS uses these core sets to inspect the inspected systems.

(3) The inspected systems are developed by different companies. The previous functions, system structures, database structures, and so on must be kept unchanged, and the code may be permitted to be modified only slightly. Meanwhile, security knowledge of the inspected systems must be kept secret from other systems.

Definition of DHIS

From the ontology point of view, although the names and types of data and operations are different in different inspected systems, we can find the same entities to express these divergent terms. The definition of these terminologies is shown in Table 1.

Table 1 Terminology Definition

Term	Symbol	Definition
Conceptual data element	d	Data element entity which may have different names and types in different inspected systems
Conceptual field	f	Table field entity which may have different names and types in different inspected systems
Conceptual table	T	$T=(f_1,f_2,f_3\cdots)$, here, f_1,f_2,f_3 are conceptual fields
Conceptual event	e	Event entity which may have different names and types in different inspected systems.
Conceptual event series	Se	$Se=(e_1,e_2,e_3\cdots)$, here, e_1,e_2,e_3 are conceptual events
Core conceptual data set	CSd	$CSd=\{d \mid d\in S_i,\ i\in[1\cdots n]\}$
Core conceptual table	CT	$CT=\{T \mid T\in S_i,\ i\in[1\cdots n]\},\ CT\subset CSd$
Core conceptual flow	CF	$CF=\{\}Se \mid Se\in S_i,\ i\in[1\cdots n]\}$

* Note: S_i is the ith-inspected system.

From the above definitions, we give out the definition of DHIS.

DHIS is the system which collects CSd (including CT) and CF of the distributed and heterogeneous inspected systems into the inspecting system, and inspects these data in order to discover the false data and un-permitted operations.

Taking $SCSd$ as the set of CSd, SCF as the set of CF, we have:

$$DHIS=(SCSd,\ SCF,\ Up,\ Insp)$$

Here, $Insp$ is the set of inspecting functions several of which are listed in Table 2. Function Up collects and updates $SCSd$ and SCF of the inspected systems.

$$Up: SCSd_i \times SCF_i \times t \rightarrow SCSd_{ic} \times SCF_{ic},\ i\in[1\cdots n]$$

Here, $SCSd_i$ and SCF_i are $SCSd$ and SCF of the ith-inspected system respectively; t is time that defines data sequential relations. $SCSd_{ic}$ and SCF_{ic} are the presentations of $SCSd_i$ and SCF_i in inspecting system, which imply the time identification.

Inspecting functions can be classified into the following two classes from the time point of

Table 2 List of inspecting functions

Function name	Definition
Parameter comparison	*Inspp*: $SCSd_{ic} \times SCSd_s \times threshold \rightarrow AuditResult$
Data integrity	*Inspc*: $SCSd_{ic} \rightarrow AuditResult$
Data balance	*Inspb*: $SCSd_{ic} \rightarrow AuditResult$
Operation flow	*Inspf*: $SCF_{ic} \times SCF_s \rightarrow AuditResult$
Sequential analysis	*Insps*: $SCSd_{ic} \rightarrow AuditResult$

* Note: $SCSd_{ic}$, SCF_{ic} are defined in the paragraph above, in which i ($i = 1 \cdots n$) means that the data come from the i-th inspected system. $SCSd_s$ is standard parameter of $SCSd$, and SCF_s is the standard operation flow. *Threshold* is the set of thresholds. *AuditResult* is the result of inspecting, that is, *AuditResult* = {determined falsity, possible, falsity, possible reality, reality}

view: (1) post-inspecting which inspects the data and operation flows after the event happened in order to alarm the un-regulated data and flows. (2) pre-inspecting which inspects the real-time data and operation flows and controls the operations of the inspected systems. Inspecting functions can also be classified into the following two classes from the application field viewpoint: (1) general inspecting functions, which can be applied in various inspecting systems. (2) domain-related inspecting functions, which are tightly related to the specific fields.

MIDDLEWARE-BASED SOLUTION OF DHIS

Representation conceptual data with metadata

We use metadata to express core conceptual data sets and core conceptual operation flows. The entities with the same conceptual meanings are generally represented by different local names and types in different inspected systems and inspecting system, but the relationship of these entities can be defined by a unique entity ID. The metadata of entities in DHIS have the items of ID, local name, local type, attribute value, and so on. The metadata of several entities made by the ontology tool Protégé 2000 (Stanford Medical Informatics, 2000) are shown in Fig.1.

DHISMetaData	Name	Type	Cardinality	Other Facets
Conceptual-data-element	data-element-ID	String	single	
	data element local name	String	single	
	type	String	single	
Conceptual_field	field-ID	String	single	
	field local name	String	single	
	type	String	single	
Conceptual_event	event-ID	String	single	
	event local name	String	single	
	type	String	single	
Conceptual-event_series	event-series-ID	String	single	
	event series local name	String	single	
	number of events	Integer	single	default=(n)
	event-ID(n)	String	multiple	default={(event-ID)*n}
Core-conceptual-data-set	core-data-set-ID	String	single	
	core data set local name	String	single	
	number of data elements	Integer	single	default=(n)
	data-element-ID(n)	String	multiple	default={(data-element-ID)*n}
Core-conceptual-table	core-table-ID	String	single	
	core table local name	String	single	
	number of fields	Integer	single	default=(n)
	field-ID(n)	String	multiple	default={(field-ID)*n}
Core-conceptual-flow	core-flow-ID	String	single	
	core flow local name	String	single	
	number of event-series	Integer	single	default=(n)
	event-series-ID(n)	String	multiple	default={(event-series-ID)*n}

Fig.1 DHIS Metadata

Security

The user of DHIS such as custom will penalize the supervised departments if DHIS finds forged data or unlawful operations. So, it is very important to get factual data. Besides collecting data factually, keeping the data unchanged in the process of transmission is also prerequisite, especially as most of data contain much secret business information of inspected departments.

The security services of data transfer should include message authentication, integrity, and confidentiality. To suit these requirements, the protocol we used, which is originated from the protocols of Huang *et al*. (2001; 2002), is something similar to SNMPv3 (Case *et al*., 1999). The message of the protocol is authenticated by using MD5 (Message digest algorithm), and encrypted with DES (Data Encryption Standard algorithm).

Data-collection with middleware technology

Data-collection is difficult because the inspected systems are distributed, heterogeneous, and generally developed by different companies. Although there are many methods to collect data, middleware technologies such as CORBA, COM/DCOM, Java RMI, and so on (Chiang, 2001; Object Management Group, 1998; Platt, 1999; Richard, 1998; Sneed, 1996) are good choices. As to DHIS, middleware technology has many advantages. Firstly, it has good security property because it needs no security knowledge such as database password, and so on. Secondly, middleware technology has the agility to satisfy the systems' diverse requirements. Thirdly, middleware can collect the data from the inspected systems into inspecting system, can inspect in real-time these data, and does not influence the inspected systems' running. Fourthly, middleware technology requires only slight modification of program code.

Our implemented middleware of DHIS (called as DHISMW, see Fig. 2) is based on message queuing technology (BEA Systems, Inc., 1997). In DHISMW middleware, the collecting data strategy that data are sent automatically by the owner of the data, rather than passively by requesting and responding mode, helps to implement the systems more easily and reduce the amount of traffic. The owner of data knows when the data are to be changed, but the receiving end cannot. If we use request and respond mode, we must request repeatedly, and can not get data in real-time. The protocol we used in DHISWM is the subset of the protocol we proposed in the reference (Huang *et al*., 2001; 2002) with some modification, which has real-time and scalable properties.

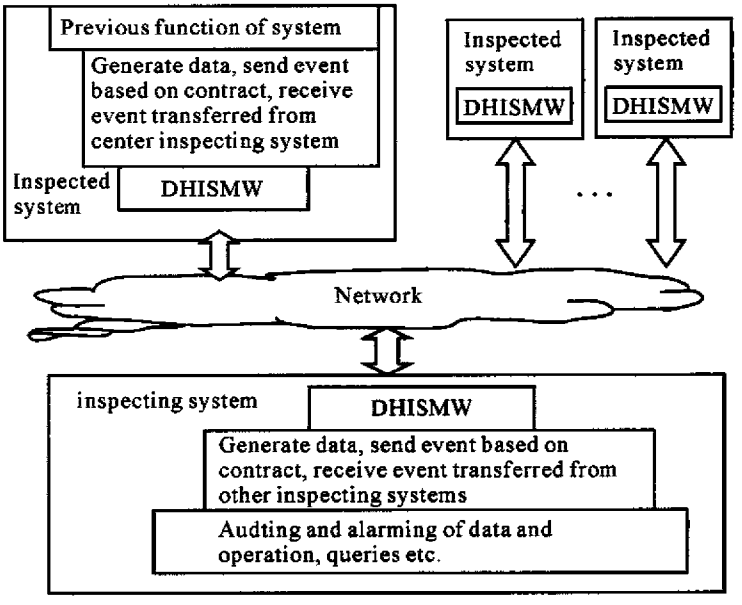

Fig.2 Middleware-based DHIS

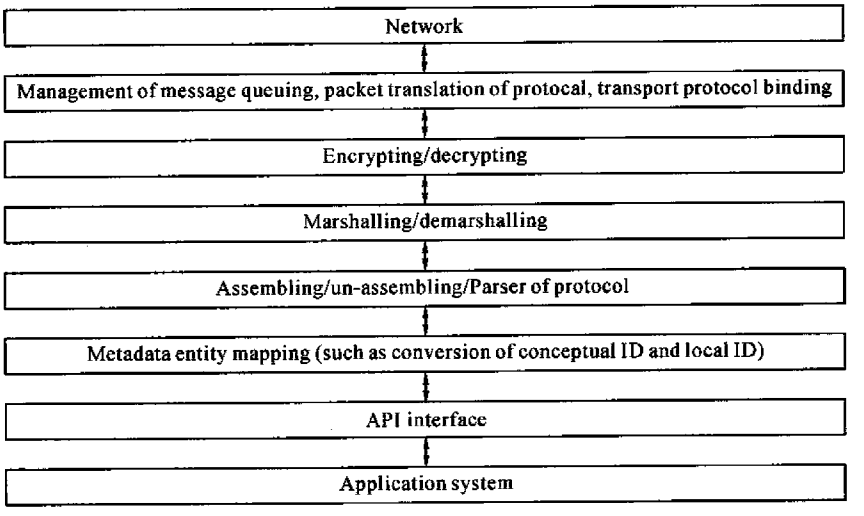

Fig.3 Middleware of DHIS (DHISMW)

DHISMW has following modules (see Fig.3):

Metadata-file-reading: This module reads the content of metadata file, and stores the metadata of DHIS in the memory. The inspected systems and inspecting system use the metadata for conversion between local names and IDs.

Communication module: This module deals with message queuing, packet transferring, transport protocol binding, the destination of transferred packet, line connected status, security services, and so forth.

Encryption/decryption: This module deals with encryption and decryption of the related segment of the message.

Marshalling/demarshalling: This module deals with marshalling and demarshalling of the transferring data.

Parser: This module parses the received packet, reorganizes Packet Data Unit (PDU) into core data sets and so on based on the specification and metadata of DHIS before submitting these data to application system through Application Programming Interface (API). Contrariwise, before sending, these data must be organized into PDU based on the specification and metadata of DHIS. The format of PDU is defined and encoded with ASN.1 (ITU-T, 1998a; 1998b) in the following.

```
PDU of DHIS ::= SEQUENCE {
  core-data-set SEQUENCE {
    core-data-set-ID VisibleString,
    data-element SEQUENCE OF SEQUENCE {
      data-element-ID VisibleString,
      value VisibleString}},
  core-table-set SEQUENCE {
    core-table-ID VisibleString,
    field SEQUENCE OF SEQUENCE {
      field-ID VisibleString,
      value VisibleString }},
  core-flow SEQUENCE {
    core-flow-ID VisibleString,
    event-series SEQUENCE OF SEQUENCE {
      event-series-ID VisibleString,
      event SEQUENCE OF SEQUENCE {
      event-ID VisibleString,
      value VisibleString}}}}
```

AN EXAMPLE

DHIS at the WENZHOU custom is an important project based on the general bureau of customs to reform management such as preventing from cheating on tariff. This system includes 10 inspected port systems and one center inspecting system. The network between center custom and ports is WAN. The center inspecting system runs in LAN network containing one communication server and 12 workstations. The number of ports and number of workstations may be enlarged or reduced according to the varied situations.

The inspected port system includes container management system, oil-can management system, and bulk cargo management system. The inspecting system exchanges real-time data with port systems, and collects all ports' changed

core data. The sources of the inspected data such as weight, volume of oilcan, container position, and so on are ports' monitoring systems and management systems. Inspecting system inspects the core data and core operations from ports, and alarms if there are any errors. We have abstracted bulk cargo conceptual table, container conceptual table, bill of lading conceptual table, oilcan conceptual table, and port storehouse conceptual table. For example, container conceptual table includes container No conceptual field, lead sealing No conceptual field, container position conceptual field, container type conceptual field, weight conceptual field, total weight conceptual field, owner conceptual field, and so forth. Moreover, we have also abstracted core conceptual operation flows, such as oilcan pass-operation flow, container pass-operation flow and bulk cargo pass-operation flow, and so on.

Because all of the inspected systems are running on the Windows operation system, we use COM technology to implement the middleware. HgzxCom.dll is inserted into the inspecting system as its component; hgkaCom.dll is inserted into each inspected port system as its component. When core conceptual data of port system have changed, they are sent to the instance of hgkaCom.dll middleware using AddValue method (see Table 3), then passed over to the instance of hgzxcom.dll middleware. After that, the instance of hgzxCom.dll middleware uses UpDataStr event to notify inspecting system that there are data to be updated. Whereas, inspecting system uses PassCommand method to send pass commands to port systems.

Table 3 Middleware DHISMW API interface

API name	DLL	Meanings
PortID	1	Set Port ID
IPAddress	1	Set Center IP Address
StartUp	1,2	Start middleware
AddValue	1	Send data to center
PassCommand	2	Send command to port
LineStatus	1,2	Line status(event)
PassCommandStr	1	Notify port application of having received center command (event)
UpDataStr	2	Notify center application of having received port data (event)
TransAllData	1	Notify port application to send all data based on contract (event)
SQLstr	2	Get Data from port databases using SQL string
SQLstrReply	1	Send results after executing SQLstr command
AnswerStr	1	Notify port application that its sent data have been received by center(event)
AnswerPassCmdStr	2	Notify center application that its sent command have been received by port (event)

* Note: 1. Custom port middleware: hgkaCom.dll; 2. Custom center middleware: hgzxCom.dll

Because heterogeneous inspected systems of DHIS are usually developed by different companies, the development of DHIS is a challenge. The experience we got and the lessons we learned in the development of this example project may be useful to other similar system developers. The main experience and lessons are as follows:

The requirement analysis stage is important in the life cycle of software development. This stage is very important especially to DHIS. Because heterogeneous systems are usually developed by different companies, the changes of requirements will disperse to all these heterogeneous systems and lead to waste of time and effort of all the involved companies. So, the user of DHIS must propose full and detailed requirements, and stipulate various standard specifications such as inspecting items and standard parameters for auditing and alarming, and so on.

The full and detailed specification documents and middleware usage documents must be sent to and clearly explained to all the involved companies. The vagueness of the specifications and middleware usage will cause a lot of errors which are very difficult to debug.

Many errors are caused by metadata definition and editing errors, and API interface usage errors. The errors are difficult to debug because they may be caused by the application system er-

rorsˈ middleware errorsˈ communication errorsˈ and the other-end middleware and application errors. For easily debuggingˈ we introduce event interfaces （ AnswerStr/AnswerPassCmdStr ） to notify whether the data/command string is sent successfully or not. In the string returned by AnswerStr/AnswerPassCmdStrˈ the prefix "ErrorBeforeEntermiddleware" means that the errors were produced before the application system sent the data/command string to DHISMW middleware: the prefix " ErrorAfterEntermiddleware " means that the errors were produced after the application system sent the data/command string to DHISMW middlewareˈ but failed to send it to the proper destination: no prefix means that the data/command string was sent successfully.

CONCLUSION

From a conceptual abstractˈ we propose the concept and definition of DHISˈ describe the details of the function of DHISˈ and propose the method of collecting data based on middleware technology. The difficulties of collecting data in DHIS are as follows: different inspected systems are usually developed by different companiesˈ and the systems' properties such as the database structure are "black boxes"; the software architectureˈ databaseˈ previous functions and so on will be kept unchangedˈ and the code may be modified only slightly; based on security considerationˈ the knowledge of security such as operation system passwordsˈ database passwords must be kept secret to other systems. We solved the above hard problems based on message queuing middlewareˈ via conversion between conceptual ID and local nameˈ and by using contract-basedˈ data auto-transferring technologies. Inspect functions can be classified as pre-inspecting and after-inspecting ones according to timeˈ and as general inspecting and domain-related inspecting ones according to application fields. Generally speakingˈ the inspecting functions of DHIS can be transformed into program code easily.

Our work is also beneficial to similar systemsˈ and suitable for distributed heterogeneous data sharing center.

Our next work is to enhance the middleware to suit all network operation systems.

ACKNOWLEDGMENTS

Special thanks are due to graduate students of the Computer Science Department of Zhejiang University Tang Shengˈ Chen Fengˈ Gao Yangˈ Xu Qi who took part in the development of the Inspecting system of WENZHOU custom. Thanks to the leaders of WENZHOU Custom for their kind support and cooperation.

References

BEA Systemsˈ Inc.ˈ 1997. Introduction to message queuingˈ http://edocs.bea.com/tuxedo/msgq/ intro/.

Caseˈ J.ˈ Mundyˈ R.ˈ Partainˈ D. and Stewartˈ B.ˈ 1999. Introduction to Version 3 of the Internet-Standard Network Management Framework. Internet Engineering Task Force RFC 2570.

Chiangˈ C.C.ˈ 2001. Wrapping legacy systems for use in heterogeneous computing environments. *Information and Software Technology*ˈ **43**: 497 – 507.

Huangˈ L.C. and Wuˈ Z.H.ˈ 2001. An SC-SS Interconnected Protocol between Heterogeneous and Distributed Monitoring and Controlling Systems. *In*: The Proceedings of the Sixth International Conference for Young Computer Scientist ICYCS2001ˈ International Academic Publishersˈ p. 188 – 192.

Huangˈ L.C. and Wuˈ Z.H.ˈ 2002. The Architecture of Centralized Monitoring and Controlling System and its High-Performance and Interoperability Protocol. *In*: The Proceedings of the International Conference of Telecommunication ICT 2002ˈ VOL（3）ˈ p. 117 – 123.

ITU-Tˈ 1998a. Information Technology - Abstract Syntax Notation One（ASN.1）: Specification of Basic Notation. International Standard ITU-T Rec. X.680ˈ 1997 | ISO/IEC 8824 – 1ˈ 1998.

ITU-Tˈ 1998b. Information Technology - ASN.1 Encoding Rules: Specification of Packed Encoding Rules（PER）. International Standard ITU-T Rec. X.691ˈ 1997 | ISO/IEC 8825-2ˈ 1998.

Object Management Groupˈ 1998. CORBA 2.2 Specification. http://www.omg.org/cgi-bin/ doc? formal/98-07-01.

Plattˈ S. D.ˈ 1999. Understanding COM + . Microsoft Press.

Richardˈ M. H.ˈ 1998. Enterprise JavaBeans. O'Reilly & Associatesˈ Inc.

Sidnieˈ M. F.ˈ 1995. SNMP: A Guide to Network Management.ˈ New York: McGraw-Hillˈ Sneed M. H.ˈ 1996. Encapsulating Legacy Systems for Use in Client/Server Systems. *In*: Proceedings of WCRE'96ˈ IEEE Pressˈ New York.

Sneedˈ M.H.ˈ 1996. Encapsulating Legacy Systems for Use in Client/Server Systems. *In*: Proceedings of WCRE'96ˈ IEEE Pressˈ New York.

Stanford Medical Informaticsˈ 2000. Protégé -2000. http://protege.stanford.edu/.