분산 개방형 EMS 설계

이지영. 신철균. 이석진. 최양석. 이정호. 김수일/H.C.Chung. D.S.Hill
현대중공업/온타리오전력공사

Design of Open Distributed EMS

J.Y.Lee.C.G.Shin.S.J.Lee.Y.S.Choi.J.H.Lee.S.I.Kim/H.C.Chung.D.S.Hill
Hyundai Heavy Industries Co., Ltd./Ontario Hydro Inc.

Abstract - The standards and technologies developed by the standard organizations and industry consortiums, which are being driven by the computer industry and are making the foundations of the open distributed systems are presented. Its benefits and impacts on EMS industry are described. This paper contains software architecture of open distributed system software. The proposing software architecture is aiming to develop all software highly versatile and open to third party hardware and software, and able to have the incremental modifications and additions.

1. Introduction

An EMS is essentially a SCADA system added capabilities. It mainly provides A)system security assessment, and precise control and optimization, and B)energy scheduling, accounting, billing and transactions. Open distributed EMS is defined as architecture built on a foundation of industry standard interfaces employing commercially available computer hardware and operating systems, communications hardware and software, database, and user interface and allowing the integration of third party products.

Historically, the unique and proprietary computer systems were the solution. However, the wide spread of IBM clone PCs and UNIX based workstations initiated open system products. Currently, all vendors are participating to some extent in open and distributed architectures.

Various open distributed system technologies are discussed in the section-2. There is neither comparisons nor analyses of the techniques. It is intended to discuss scenarios due to the practical operations of open distributed system technologies. The section-3 is designed to present the proposing Software Architecture for open distributed system. It discusses the high level architecture for all system functions, and presents the design details by software component descriptions with examples. For interested readers, the additional details could be found in the reference [1]. This paper concludes with a)the expected results of the proposing Software Architecture, and b)projected-benefits and open-issues of fully developed open distributed system.

2. Standards and Technologies to Open Distributed System

This section is composed of ten subsections. Each subsection presents quite independent subjects. Therefore, there is no ordering preference except the first and last subsection.

2.1 Definition

Open distributed system technology is based on industry standards that allow the control center to be able to quickly adapt and support utility strategies. Conceptually, open distributed system technology features A)Portability (Migration from platform to platform), B)Scalability (platform Upgrade), C)Expandability (Modular Additions), D)Interoperability (Interfacing with other programs), E)Compatibility (Release Upgrades), F)Consistency(User Interactions), and G) Openness (Third Party off-the-shelf products).

2.2 Hardware Interface

A)Commercial Bus Interface Standards (Multibus, VMEbus, Future Bus, Q-Bus), and B)Parallel Connection Standards (Small Computer System Interface, Intelligent Peripheral Interface, and Storage Module Drive, IEEE488, GPIB)

2.3 Communication Interface

A)Serial Connection Standards (RS-232 Single Ended, RS-422 Differential, and RS-485 Multi-Drop), B)ISO OSI 7 layer model, C)Local Area Network(Ethernet, FDDI, TCP/IP),D)Wide Area Network(WSCC, ELCOM, IDEC, UCA, ICCP),and E)Radio Frequency Connection.

2.4 MMI

Man Machine Interface provides an easy means for input (data) and output (result) processing. It should be an attractive and simple to use interface human and machine. There are many reasons and advantages to use one and another GUIs. The main graphic environments are A)X-Windows, B)Motif, C)GKS (Graphics Kernel System),and D)PHIGS(Programmer's Hierarchical Interactive Graphics System).

2.5 Software Portability Support

A)Standard high level language (FORTRAN, C, C++), B)User level tasks instead of operating system modification, C)Dynamic Shared Libraries, D)Shared Memory regions, E)Configurable parameters for all configurable items, F)Application Services (File management for mirrored files, Date/Time Services, Process Management Services, Configurable Parameter Services,Program Scheduling,Communication Services).

2.6 Distributed Database

A)Single SCADA Image, B)Concurrent Editing, C)Distribution of Real Time Data, D)Fault Detection and Recovery (Roll back, and Roll Forward).

2.7 Open EMS Application Interfaces

Initiative began with foundation for open cooperative for utility software (FOCUS). The main issues are reaching consensus, achieving recognition as a standard and keeping standard up to date. Three Layer Approach (Model definition and export specification, Specification of API's for data access, and Definition of object model and interface layer) could be the general solution. EPRI working group on control center application program interfaces(CCAPI) is making effort for the complete solution.

2.8 Customer Focus

The products are developed mainly to meet the customer needs, requirements and improvements. The following strategies should lead to the best results: Internal Customer Survey, External Customer Survey, Customer Involvement in Direction of EMS, and Feedback from Rapid Prototyping.

2.9 Impact On EMS Industry

A)Environment of rapid, new product introductions, B)Continuous Standard Evolution, C)Continuing Education, D)Increasing number of incremental upgrades, E)Procurement Including Third Party Suppliers, F)Utility Enterprise-wide Integration.

2.10 Open Distributed System Evolution Scenario

2.10.1 Foundation Phase

A)Distributed Processors On LAN, B)POSIX Operating System Interface, C)X/MOTIF Man Machine Interface, and D)TCP/IP-OSI LAN.

2.10.2 Advanced Phase

A)System vendors make Database Definition Public, B)Catalog of Available Third Party Applications, C)Standards for EMS Database Specification and Interface, and D)Periodic release with updates by organizations and industries.

EMS designers are experiencing that the industries are moving from foundation phase to advanced phase. We may have the different form of the advanced phase due to the fast developing technologies and industries interests. There are too many technologies and unknown equations, therefore, EMS design is tremendously difficult task and the business is quite risky investment. The reference [2] presents the examples of the vendors and end users open distributed EMS.

3. Software Architecture for Open Distributed System

This section presents EMS design (Software Architecture) of the ongoing EMS development project by Hyundai Heavy Industries. The proposing Software Architecture is designed for all system and application functions. The Software Architecture will define a model on which the design of new software can be based.

3.1 Scope of Software Architecture

The Software Architecture of the systems must support the following different uses:

A) Operational use, these users are a small controlled set of users(e.g. control room staffs). The systems and the user interfaces need to be robust, and have high performance and availability. The user interface to these systems by:

a) user interfaces supplied as part of the systems. Perhaps, the specialized interfaces, as they need the high performance. b) Data interfaces, there are specific functions where the user either wants to supply the data or receive the data directly to the users own computing resources. and c) commercially supplied users interfaces (e.g. Web browsers), there is no need to load special software into the users PCs. This interface may become more wide spread in the future.

B) Information use, a large number of users require access to the information in the systems. These users often do not want specialized software loaded on their PCs but they want to use generally available commercial software. These users want a) information displays e.g. using Web browser, b) direct access to the databases for decision support, for one of kind queries using standard spreadsheets, etc, c) direct access for applications developed by the end user, and d) tools to extract specific information or produce standard reports.

3.2 Objectives of Software Architecture

3.2.1 Managing a Changing Software Environment

Defining the Software Architecture is difficult when the software technique are constantly changing. The Software Architecture must not be restrictive, preventing the use of new productive techniques. Generally it is the user interface that is undergoing rapid changes. The relational databases that store the data and the C++ and FORTRAN software programs that provide the business rules and algorithms are much more stable, they are evolving rather than undergoing the radical changes in direction experienced in the user interface area.

3.2.2 Managing Changing User Requirements

It is unreasonable to expect that a) all the users have the same requirements and b)the user to define a set of requirements that will remain unchanged for any length of time. The Software Architecture must promote functionally modular software, as is found in component type Software Architectures. Such architectures provide for: a) rapid prototype development, by the effective reuse of as much existing software as possible (both low level common routines and higher level functional routines), and b) ability to quickly make changes to existing systems because of their well defined modularity, and the extensive software reuse capability.

3.2.3Work well Across Local and Wide Area Network

The user interfaces will be provided on the users own dedicated personal workstation(s), whereas the databases and calculation routines will be provided on central servers. In addition, for almost all systems, the users will be connected to the central servers by a variety of means: a) locally, over the high speed LAN, b) remotely over medium speed networks, and c)remotely over slow speed connections, e.g. dial-up or Internet. The Software Architecture must provide a model and guidelines that allow the design process to minimize network traffic in order to provide acceptable performance for all these types of connection.

3.2.4 Develop and Support Software effectively

Defining a Software Architecture that promotes effective development and support of software must deal with three issues:

A) Avoiding Duplication

Many user interface programs need to display the same information, many programs need access to the same data or need to perform the same calculation on the data. The Software Architecture needs to ensure that it is not necessary to duplicate code or data. Such duplication increases the time and cost for both software development and software support. The Software Architecture needs to provide common display modules that can be used by all user interface programs,data that is stored once in the system,but to which all programs have easy access, and calculation modules be available for all programs to use.

B) Promoting Specialized Skills

To develop modern software requires detailed knowledge of numerous areas, user interface programming, database design and programming, software techniques and interfaces(e.g. ODBC, exception handling). It is no longer reasonable to expect all staff to be fully capable in all areas, they must have a general appreciation of all areas, but a detailed understanding of specific areas. The Software Architecture must promote modular design and interfaces to ensure that work produced by specialists in one area can be integrated with other areas, and one specialist can use the work produced by a specialist from a different area, without having to understand the internal details.

C) Ensuring Consistency and Integrity of Information

The software design needs consistency, both from the way it is designed and the products it uses, in order for the software to be maintained, administered, and operated effectively. The integrity of the system is based on providing consistent information that is two different displays or reports, should not provide different values for the same information. Generally, inconsistencies arise because the same value is calculated by two different routines under different conditions, or updates are only partially implemented, that is, not all affected code is changed. Again this comes down to the Software Architecture promoting modularity, where all programs use common services, so that when a change is applied there is no concern that some instances of the code have been missed.

3.3 Software Component Descriptions

Software components form the basis of the Software Architecture. Each component is providing a well defined function or service, which is self sustaining including initialization, error handling, etc. When components provide a service to other components they will do so via well defined interfaces, that are based on object technology (this does not imply that the internal design of the component need be based on an object design), this will allow for the use of standard network and object services. The Software Architecture has following basic software components:

3.3.1 Software Components that Generally Execute on the Server Hardware

A) System Programs: The system programs are the main programs, they control the sequence of events, almost always the cyclic processes. When started up they will initialize all the application servers for their area; e.g. security assessment. Their logic will control when services are performed, and they are responsible for all synchronizing and sequence control, e.g. ensuring state estimator completes before downstream security functions are performed. They do not perform much work themselves, they call various services in the application programs to get the work done.

B) Applications Servers: these provide the common algorithms or the business rules needed by the system and user interface programs and the INTERNET servers. An application server handles a single function, it contains both the algorithm/calculation services and the data update and retrieval services for the particular function. The application server runs on the server hardware and not on the client PC.

C) Operational Database Servers: their main responsibilities are a)storing the data, b)providing the access services that provide a logical interface to the data, c)ensuring the integrity of the data, and d)ensuring the security. These support the functioning of the operational systems, and their integrity is critical, therefore, access to these databases is only via controlled interfaces (application servers). The databases are based on commercial database management systems, wherever possible, but also include custom designed database servers when required (e.g. high performance).

D) Historical Database Severs: the operational database servers only maintain data for as long as it is active and needed for operational purposes. When the data becomes inactive it is moved to the historical database server, that provides the data warehousing capability. The historical database server contains all historical data, such as, real-time data, alarm data, system log information, bid data, schedule data, settlement data, etc. They are read only but the user has full access to the data and the database structure. They are commercial database management systems (e.g. SQL) that are easily accessible by the user.

E)Information Servers: these provide the interface to information to all users. The information servers will handle all general user requests for information. These will be Web servers. They will have some pre-formatted pages that are frequently requested, these pages will be updated daily or weekly depending on the information, but the majority of the users requests will be handled by dynamically getting the appropriate data from the operational servers via the services of the application servers, and from the historical database servers.

3.3.2 Software Components that Execute on the Client Hardware:

A) User Interface Programs: these are user task oriented programs, the displays integrate the information of many functions needed to perform a user task, e.g. integration of outage information, and security analysis. These programs create display data from many different sources in order to present information in an integrated way that is geared to the user tasks. The user interface programs are the controlling programs in the user's PC, they are the highest level window container. They call display services(objects) within the display servers to format and output the requested information.

B) display servers: these provide the capability to display basic information e.g. display outage information for specific outage. A display server will contain all the display services(objects) for a particular function. The services are used by the user interface programs and in the web page displays. The services are responsible for making the request to the appropriate application server to get the data, and then formatting the data and outputting the display. The display servers run on the user PC.

C) standard software: Web browsers, spreadsheets and the like on the user's PC.

3.3.3 Examples

The following examples show how these software components interact :

Example 1- Security Assessment

The system is operating and the security assessment chain needs to complete the cycle, or has to be terminated so that an update sequence can be performed.

Programming Description

A) The security assessment system program is responsible for controlling the sequence of execution of the security assessment chain.

B) The security assessment system program calls the status change method of the real-time processing server, passing the time of the last status change processed. The status change method returns whether there has been a status change since. The security assessment system program makes the call to the status change method as if it was a local method within his process space, the system resolves the location of the server and initiates the request across the network.

C) The security assessment system program then decides if an update or cyclic chain initiation is required. If so, then it initiates a call to the read real-time data from the real-time processing server to transfer the real-time data to the security application database.

D)The security assessment system program then call the compute methods of all the security applications necessary to complete the full assessment. It ensures that the processing happens in the correct order, waiting for functions to finish before initiating others, e.g. State Estimator must complete before Security Analysis can be initiated, and others it initiates in parallel, if there is no dependency.

Example 2- Line Limit Display

The user currently has a regional diagram displayed, (showing lines and there flows) and wants to display the thermal limits details of a particular line.

Programming Description

A) The BES (Bulk Electric System) Display program (a user interface program) controls the window that is displaying the regional picture.

B) The display object makes a request for the thermal limit data to service(object) in the application server. To the display server the data retrieval service appears as if it is local, the system determines the physical location of the application server, and routes the request across the network. The request for a specific application service is concise request requiring a single return message, which contains all the relevant information.

C) The application service within the application server makes all the necessary calls to the databases (this may involve several calls to different databases). It handles the database connections and the numerous messages, such as involved in ODBC requests. It then assembles the data and returns the single message to the display object.

D) Upon receiving the requested data, the display object performs the processing to display the information. It creates a window, formats the data, and outputs the display. The display object constructs the display the same as any Visual Basic or Visual C++ program does, it uses controls placed on a form.

E) The BES display program then waits for further user initiated events.

4. CONCLUSIONS

Proposing Software Architecture defines the various software components and techniques, and how to divide the functions between the components. Using the Software Architecture will ensure that new software takes advantage of the system architecture, including the network without degrading performance, and that new software can be developed quickly and with improved quality by being able to use software that already exists. Furthermore, the Software Architecture provides a separation of the user interface from the databases and business rules and algorithms, to maintain the investment in as much software as possible, while being able to change user interface elements to meet current demands.

Fully operational open distributed system would provide the following benefits: A)incremental expansion, B)modular upgrade, C)multiple source of supply, D)range of available third party software and hardware, E)equipment price/performance, F)availability of trained personnel, G)large user's group. However, there are a number of open-issues (e.g. a)evolving and recognition of standards, and b)interfacing to real-time proprietary database, and to EMS MMI system) to be resolved prior to have the complete open distributed system. Certainly, it is the challenge to EMS industry. There are undergoing continuous developments, which are leading to race without a finish line, to achieve the complete development of the open distributed system.

5. Acknowledgment

The development work is financed in part by a contract from Korea Electric Power Company. The authors of HHI would like to thank CYBERSOFT (Florida, U.S.A.) consultants for extremely productive discussions.

[References]

[1] J.Y. Lee at el., EMS Development Project Report, KEPCO, DAE-JEON, Oct.1996.

[2]Y. Kokai at el, Recent Development In Open Systems For EMS/SCADA, 12th Power Systems Computation Conference, Dresden, Aug.1996.