Transactions of the Korean Nuclear Society Spring Meeting
Chuncheon, Korea, May 25-26 2006

Software Development of RMS for HANARO Reactor by Using an Architectural Approach

Yong Suk Suh,[a] Seok Boong Hong,[a] Ki Sung Son,[b] Ki Hyun Lee,[b] Hyeon Soo Kim [c]
a I&C and HF Div., KAERI, 150 Dukjin-dong, Yuseong-gu, Daejon, Korea, 305-353, yssuh@kaeri.re.kr
b Control Tech. Research Inst., SEC Co., Ltd.,974-1 Goyeon-ri Woongchon-myon, Ulju-gun, Ulsan, Korea, 689-871
c Dept. of Computer Science and Eng., Chungnam Nat'l Univ., 220 Gung-dong, Yuseong-gu, Daejon, Korea, 305-764

1. Introduction

In KAERI, the project for updating the Radiation Monitoring System (RMS) for HANARO was launched in Nov. 2005 with a budget of 80 million won. The RMS was originally developed with Santa Cruz Operation (SCO) operating system and Dataview graphic system by Victoreen in USA. At the time of the Y2k problem, it was upgraded with Windows NT and Delphi by Visiontech in Korea in order to overcome the problem [1]. In 2005, when KAERI planed to construct a Fuel Test Loop (FTL) facility and install 6 Local Monitoring Systems (LMS) for the FTL inside HANARO, it was requested to update the HANARO RMS since several operating problems and maintenance difficulties had been reported. It has been difficult to maintain 10 thousand lines of a source code due to poor documentation and many dead codes in it.

The RMS consists of 30 LMSs, 5 Remote Monitoring Terminals (RMTs). The LMSs are electrical class 1E and the RMTs are Non-1E. The scope of the project is to develop the RMTs since the LMS was supplied by Victoreen. Actually, Korea has no company manufacturing the LMS so the development project of RMS in Korea is to develop the RMTs by importing the LMSs from overseas.

There are two constraints imposed on the software development for the RMTs: short period (6 months) and low budget (60 million won). The user requirements for the RMTs are summarized as follows:

> *"The upgraded system should function better than the current system. No loss of data is allowed, easily maintained and extended, and capable of portability with radiation monitoring systems which will be constructed in the future."*

In order to meet the requirements under the constraints, it is necessary to find an efficient way for the development. We considered adopting two methods for it. The first method is a fast prototyping with a minimal programming. The fast prototyping was inevitable because the functional requirements were not clearly fixed when the project was launched, but simply stated that better functions were required. For this reason, the prototyping is the best solution to see what the user want. The second one is the Attribute Driven Design Method (ADDM) [2]. The ADDM leads us to perform the following procedures:

1) Elicit a quality goal from the requirements.
2) Construct an architecture to meet the goal.
3) Validate the architecture if the goal is achieved.
4) Refine the architecture if necessary and identify the reusability of the architecture.

The ADDM requires the developers to concentrate on the quality goal of the target before starting a design. It is reasonable persuasion to avoid the failure of project in case concentrating on programming. It then requires building a system by seeing if it can meet the goal. We considered all these activities as an architectural approach to develop a system in this paper. The approach we performed in the software engineering process such as an analysis, design, implement, and test, is described in the next section.

2. The Architectural Approach

The term "Architecture" is defined as "the art or science of building" in the merriam-webster dictionary. With the definition, we applied the ADDM to building the RMS as described in the following subsections.

2.1 Quality goal and tactics

After analyzing the software development requirements, quality goals are elicited from them as follows:

1) Functionality: better function than the current RMS.
2) Reliability: no loss of data is allowed.
3) Maintainability: complete within 24 hours.
4) Extendibility: easy addition of LMSs, RMTs, and user interfaces.
5) Portability: portable with future RMS in KAERI.

In order to meet the quality goals, we decided to have following tactics based on engineering experience:

1) Functionality: use of a prototyping as a means to understand the functional requirements.
2) Reliability: construction of dual hot-standby systems.
3) Maintainability: loosely coupled modules and a localization.
4) Extendibility: object-oriented design and programming.
5) Portability: use of open interface standards.

2.2 System Architecture

The architecture of the RMS was originally designed as a distributed system with RS485/232C communications. We changed the communication to Ethernet as shown in Fig. 1. We can achieve the extendibility and portability from the system architecture.

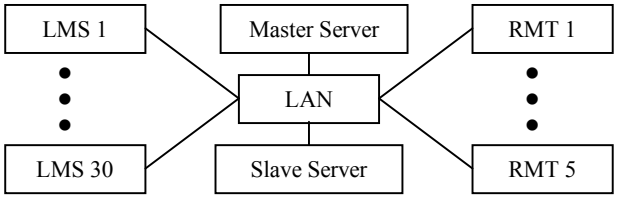

Fig. 1 System Architecture

2.3 Server Architecture

The server architecture was designed with dual hot-standby server systems as shown in Fig. 2 so that we can achieve the reliability from the system architecture. The server and the slave are synchronized at start-up time. The master server synchronizes not only the real-time data but also the historically logged data with the slave. They communicate with each other with a heartbeat. When the master fails, the slave becomes the master. When the failed master comes back to join the LAN, it becomes the slave. At this time the slave should catch-up with the logged data of the master. This is done by the master.

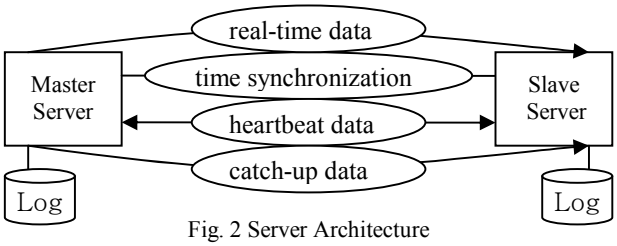

Fig. 2 Server Architecture

2.4 Software Architecture

The software is developed with Adroit which is a tool for building a SCADA (Supervisory Control and Data Acquisition) [3]. The reason for selecting Adroit is due to a low budget for the development and a minimization of the programming effort so we can easily prototype the user interfaces. The Adroit supports the object-oriented design and client-server architecture as shown in Fig. 3. We can easily define agents as classes and create tags as instances of the agents without a programming. And then we dynamically design graphic objects for user interfaces and only assign the tags to the graphic objects. We can achieve the maintainability and extendibility with the Adroit.

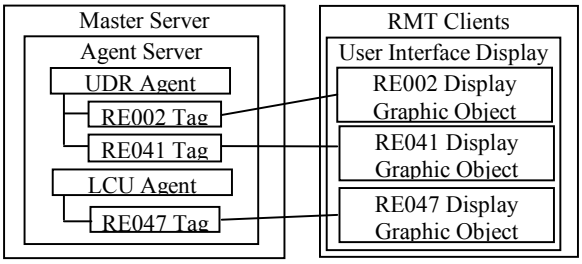

Fig. 3 Software Architecture

The Adroit supports OPC (Object linking embedding for Process Control) which is an internationally industrial standard providing an open interface for exchanging data dynamically between different manufacturers. In order to communicate with Victoreen LMSs, we had to make a driver program as a Windows DLL (Dynamic Link Library) program and add it to the Adroit. We can achieve the portability by using the OPC interface and the DLL programming.

2.5 Validation of the Architecture

The architecture is being validated through testing. Functionality is being iteratively validated by demonstrating the prototype to the user and by performing FAT (Factory Acceptance Testing). Reliability is being validated by making the server fail and ensuring that no loss of data occurs. Maintainability is being validated by making any of the changes or faults and ensuring that they are fixed within 24 hours. Extendibility is being validated by adding LMSs, RMTs, and user interfaces. Portability is being validated by communicating with other systems.

2.6 Reusability of Architecture

If the reusability of the system, the architecture, or the software is achieved, they provide very valuable assets for future projects. In the case of the RMS development, we identified that the developed RMS itself can be reused for a future project by simply changing tags and user interfaces.

3. Conclusion

With the constraints such as a short period of development, low budget, and unfixed functional requirements, it was necessary to find an efficient way of developing the RMS software. For this, we adopted an architectural approach and used the Adroit tool. While performing the approach, the quality goal we elicited led us to properly build the RMS architecture containing a system, server, and software. We built the LAN-based distributed system, dual hot-standby servers, and client-server software architecture. We adopted an object-oriented design concept for achieving a maintainability and extendibility so that objects in clients and servers can run independently with minimal interfaces. Using the Adroit tool, we could prototype the system iteratively with a minimal programming. The architectural approach provided us with an efficient way to build and validate the architecture per quality goals. We will reuse the architecture for future RMS development.

REFERENCES

[1] KAERI/RR-2059/99, "Upgrade of RMS for Y2k Problems in RX and related Building of HANARO", KAERI, 2000.
[2] Len Bass, et al., *Software Architecture in Practice*, 2nd Ed. Addison Wesley, 2003.
[3] www.adroit.co.za, www.bnftech.com