Research on the Component-based Design Method of the Ground Measurement and Control System

Wei-dong MA* and Zhi-ying WANG

Institute of Electronic Engineering, China Academy of Engineering Physics, China

*Corresponding author

Keywords: Domain Model, Component-based software engineering, Measurement system.

Abstract. Modern software systems are growing ever more complex, and the component-based software engineering (CBSE) is viewed as an opportunity to deal with the increasing complexity of measurement and control system. This paper analyzes the task profile of measurement and control system (GMCS) and discusses the component-based design methods. The paper submits a novel model called CV-MDN for software of GMCS that includes five parts: user controller, display views, domain model, data services and network service. The five parts provide cooperation method with software buses. Finally, the paper introduces the architecture of software design and it analyzes the software system functions. The CV-MDN model can be formed a general method for design of GMCS.

Introduction

The ground measurement and control system (GMCS) is a kind of equipment test simulation system, which is usually composed of host computer control unit and slave embedded computer measurement and control unit. The main function of GMCS is to support ground power, to provide the excitation signals and control signals. It collects the various actions, system state parameters and the test information of the electronics devices so that processes the corresponding data.

The classic main control unit is shown in figure 1. The measurement and control system is composed of three parts: the main measurement and control unit, the embedded measurement unit, and the auxiliary device. The communication between them is carried out through the Ethernet (UDP protocol).

![Diagram of the main control unit](image)

This paper introduces a new design of distributed simulation system based on software components. Component-based software engineering helps deal with complexity by following a divide-and-conquer approach, modularising large software systems into smaller, reusable units-called (software) components. CBSE is supposed to be particularly effective in the context where components are developed by independent third party developers and bought by application builders to be composed into complete applications. The combination of software bus architecture and reuse components of GMCS is a better method to development and integration the system. It can reduce the cost, enhance the static and dynamic adaptability, optimization model and verification.
Component-based Software Design

With the high reliability requirement of military equipment, the hardware and software reuse technology has become the focus. An effective way to improve the traditional software design is to depend software reuse technology to design the software architecture. Component-Based development method is thought to be an effective technique to tackle software crisis. The component-based strategy aims at managing complexity, shortening time-to-market, and reducing maintenance requirements by building systems with existing components, and it is focused on embedded control software for vehicle systems, e.g., passenger cars [1].

Through the status analysis of software development modes of GMCS for the domestic large-scale rocket engines, the Rational Unified Process "4+1" view method is proposed [2]. The modeling of software was carried out in combination with object-oriented design idea. It laid a good foundation for solving the problems generally existing in the software development of ground testing and monitoring system for large-scale rocket engines.

Component-based software technology is the further development of object-oriented technology, and the software developers can directly use the existing components, through the assembly to complete the development of the system. A component is usually defined an executable unit that encapsulates data and functionality. Most of the distributed components are implemented by software bus. "Software backplane + Component" general software architecture has been used to achieve the stability of the core layer, general, component layer configurable, scalable [4]. In the airborne software architecture design phase, people assign the safety related development assurance level to specific component to ensure the quality of the product [5]. A new component model is proposed based on the Web Service technology and the strong assembly and disassembly of components [6]. By separating the component-based applications with the framework of the internal components, the proposed component model is reusable and consistent to multi-platforms.

Component-based software design can improve software reliability. For example, in the stage of airborne software architecture design, the security level is allocated to specific components to ensure product quality [5]. The evaluation of system reliability based on component plays a very important role in software engineering. Component-based system has the following characteristics: each member only completed part of the system, the function of the whole system must be completed through collaboration between components, so the reliability of each component will ultimately affect the reliability of the whole system [7].

The Design Model of MVC

The programming of GUI is often organized around the event-driven so that the user interface, data processing, program functions and display code are completely tangled together. However, the data processing and interface are binding to the application greatly reduces the flexibility of the program, and some changes in software requirements will involve a lot of code re-design that increasing the workload of program development and maintenance.

Model-view-controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts in order to separate internal representations of information from the ways that information is presented and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development. Because MVC decouples the various components of an application, developers are able to work in parallel on different components without impacting and/or blocking one another. By creating components that are independent of one another, developers are able to reuse components quickly and easily in other applications.

The MVC means of communication between different parts of the object definition, program structure clear and flexible, as shown in figure 2.
The CV-MDN Model of GMCS

This paper proposes a CV-MDN model that combined the component-based design with MVC, which is shown in figure 3.

The CV-MDN is the abbreviation of Console Controller, View Windows, Domain Model Services, Data Services and Network Services. CV represents the user controller and display interface. MDN represents domain model, data service and network service component. They are designed according to the needs of the component, through the mutual influence and close cooperation between each other to complete the system goals.

Software Deployment Architecture

According to the CV-MDN design pattern, we put forward the ground control simulation software deployment framework, as shown in figure 4. The architecture is based on distributed architecture design, which can separate the device control and test results.
The whole measurement and control software system adopts distributed architecture, based on cross platform and component design. The overall operation entities include: network and data processing center, database, command line console and other data display components. Network and data processing center with MDN components, complete acceptance, transmission, organization, storage and control data, the background service process operation.

The main control in CV uses the command line script window mode, which is based on the command interpreter mode, avoiding the use of the mandatory user interface. The DSL script can be prepared to deal with a certain type of simulation tasks. The other display components in CV run in full window mode, which supports the distributed data processing environment, can adjust the size and position of the window, and realize the adaptive display zoom. Through the network and the background data processing center communication, access to relevant display, print data. All the windows can be opened and closed at any time, automatically connect data processing or disconnect the data processing source.

The network data package parser adopts configurable component design, which can add different configuration components for different simulation tasks. All display windows using a universal, uniform design. Parameter settings using the way to read the configuration file, the results are shown in tabular form, with self-test, debugging, testing and other functions.

**Software Design Framework**

The implement of the software is composed of four subsystems, which are Web application server, communication server, command line console, and simulation slave computer. The Web application server and the communication server are deployed on the same machine. Command line console is an independent GUI application, as the client of the communication server can run on a variety of different computers, and communication between the server through the TCP socket communication. The Embedded machine is simulated as the external equipment of the system, and it communicates with the communication server through UDP socket.
Summary
The component-based design method of GMCS is discussed in this paper. On the basis of MVC design pattern and component-based design, this paper presents a CV-MDN model. The model consists of five components: user control, display interface, domain model, data service and network service. The CV-MDN model is successfully implemented in the implementation of GMCS.

Acknowledgement
This research was financially supported by the Defense Industrial Technology Development Program.

References