Management Software for Stratospheric Airship

Afshin shaabany [1], Fatemeh Jamshidi [2]

[1, 2] Science and Research Branch, Islamic Azad University, Fars, Iran
afshinshy@yahoo.com

Abstract: In this paper a management software for avionics system of stratosphere airship is introduced that is sufficiently accurate and reliable. This paper introduces the object-oriented design of the management software based on the Unified Modeling Language (UML). First, the UML notation used in this paper and modeling steps is introduced. Then, the avionics system of stratosphere airship is depicted. Moreover, requirement analysis is proposed. Finally, we present the framework of management software and the detailed design of the class model.
[Afshin shaabany, Fatemeh jamshidi. **Management Software for Stratospheric Airship**. Journal of American Science 2011;7(9):144-148]. (ISSN: 1545-1003). http://www.americanscience.org.

Keywords: Management Software; Software Design; Stratosphere Airship

1. Introduction

Presently, there is a strong interest in developing unmanned stratosphere airship for its unique advantages in military application and scientific exploration. The avionics system is very important to the unmanned airship. So the management software for the avionics system must be designed sufficiently accurate and reliable. The management functions are realized by embedded software based on the VxWorks real-time operating system. The Object-Oriented Design Methods has been used in the design process of many kinds of software, including embedded software (Selic, 1994). The software designed by the OO method is excellent in maintainability, expansibility, flexibility and reusability of codes. Therefore, we use the OO method to model and design the management software for stratosphere airship.

In the design process of software, it is essential to do the system modeling, which is helpful to system visualization, especially for complicated system like the management software for stratosphere airship. UML is a common visualization modeling language and used to describe and visualize the OO system. So the management software is developed based on the UML (Booch, 1999 and Guoshun, 2010 and Bikander, 2000 and Kefu, 2006).

2. UML Notation and Modeling Steps

UML supplies a standard way to write an OO system's blueprints. Many kinds of diagrams are provided by the UML to observe the different aspects of the system individually. The diagrams used in this paper are introduced as follows (Hassan, 2004 and Bruce, 1998 and Ganssle, 1999 and Selic, 2000 and Atkinson, 1997 and Jong, 2002).

A. Use Case Diagram

The use case diagram is a way of associating the 'actors' and 'use-cases' in a system via the use of relationships. Usually it is used to specify the functional requirements of the system and lay the foundation of the development process of the software system.

B. Class Diagram

Class diagram is the most common diagram in the OO modeling. It shows the static structure of the system and describes a set of classes, interfaces and their relationships to other classes, such as inheritance or generalization, and associations.

C. Collaboration Diagram

Collaboration diagram defines a specific way to use objects in a system by showing the possible interactions between them.

The OO Design Method has three steps in modeling. The first step is to analyze the requirements of system, then set up the static model for system. The last step is to describe the behavior of the system. All the models in the first and second step are static, including the use case diagrams and class diagrams.

3. Introduction of Avionics System

There are many subsystems in the stratosphere airship, such as avionics system. This subsystem is equipped in the gondola fixed below the envelope of stratosphere airship. Considering the complicated interaction between devices, data bus technology is adopted here. CAN bus is applied to fulfill data exchanges between devices equipped outside of the gondola, while 1553B bus is utilized to transmit navigation, telecontrol and telemetry data in the gondola. The onboard computers including flight control computer and vehicle management computer are the core nodes of all the buses. These computers can communicate with each device of the buses.

4. Requirement Analysis

The functional requirements of the system are specified in terms of use cases (Figure 1). For the management software, use cases are divided into three use case packets: (1) data management, (2) system detection, (3) error management.

A. Data Management

The stratosphere airship must deal with the system data in real time, so the functions of data receiving and sending are the base of the software. The use case packet includes telecontrol and telemetry data management, status data management, and the operation of data sending and receiving.

B. System Detection

The stratosphere airship needs to operate automatically for a very long time, so it is necessary for the management software to detect system status. This use case packet includes two aspects: self-detection and device detection.

C. Error Management

When system status is abnormal, the management software must diagnose and expel the breakdown. This use case packet includes computer error management and peripheral equipment error management.

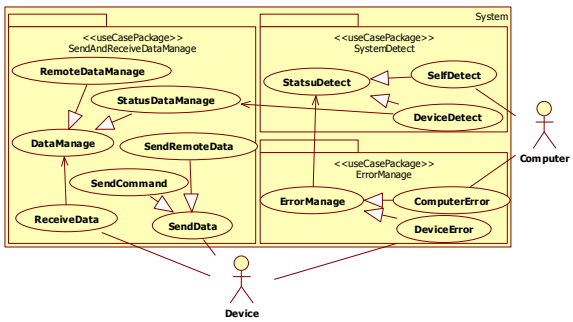

Figure 1. Use case diagram of management software.

5. Framework Design

Based on the system use case diagram, the framework design is the foundation of development process. Good design of the framework can improve the maintainability and expansibility of the software, reduce the unnecessary work. In the framework design of management software, many kinds of class models (Figure 2) are designed to meet the requirements.

Three kinds of classes are designed: (1) abstract data class, (2) device interface class, (3) task class. The task class includes data exchange class, data manipulation class, timer class, period task class, system detection class and error management class. The design of these classes is illustrated as follows:

A. Abstract Data Class

The global variables of the software are packaged in the abstract data class, including Status Data class, SJZYC Frame class, SJZYK Frame class and SJZCX Frame class. These classes are used to store the status data, telecontrol and telemetry data. All task and device interface classes need to read and update the global variables to implement homologous functions.

B. Device Interface Class

According to different devices, the device interface classes are designed, including Device_ CAN_ Interface class, Device_ 1553B_ Interface class, Device_ ALT1_ Interface class and Device_ ALT2_ Interface class. These classes are used to initialize the devices, send and receive relevant data.

C. Task Class

Data exchange class: The data exchange class is designed to send and receive the status data, tele control and telemetry data, including Task Read Data_ High CAN1 class, Task Read Data_ High CAN2 class and Task Send Data class. After conversion, data exchange class sends and receives the data by calling the device interface class, for example the Device_ CAN_ Interface class.

Data manipulation class: The data manipulation class is used to realize the function of data processing. The Task Data Manage CAN class and Task Data Manage Msg class deal with the data received from the CAN bus and 1553B bus separately.

Timer class: Task Period_ Time class realizes the function of timing periodically sends messages to the data exchange class and give semaphores to active the period tasks.

Period task class: The Task Period class belongs to the period task class. This class takes the semaphores from the Task Period_ Time class and carries out the period tasks.

System detection class: Task BIT class belongs to the system detection class, is designed to implement the system detection function. The system detection includes self-detection and device detection. When the system status is abnormal, the Task BIT class sends messages to error management class to deal with it.

Error management class: The Task Erro Manager class belongs to the error management class. After receiving the message queens of reporting errors from other classes, it is called to treat the errors of computers or devices.

Figure 2. Class diagram of management software.

6. Detail Design of Class Model

After finishing the design of framework, the classes are designed to meet the requirements of the software. The designing idea of the abstract data class, device interface class and task class is detailed in this section.

A. Abstract Data Class

On account of large amount, abundant information, the status data, tele control and telemetry data are hard to deal with. In the software system, the abstract data class is used to store data and provide the operations, reading or writing. The abstract data class packages the data structure, hides the detail, provides the unified interfaces and makes the software easy to be maintained.

Take the SJZYC Frame class for an example. The telemetry data is packaged in this class, which includes 22 telemetry data frames. The structure of the telemetry data frame is not steady, and may be changed frequently. For this reason, the key of the design is to locate the data and improve the maintainability.

The attribute includes 22 data classes and 22 selection functions. The 22 data classes are numbered in sequence from 1 to 22, and the number is named Frame Number (FN). The data in the data class is also numbered in the same way with the number named Data Number (DN). Each data class has a selection function with the DN as the parameter. The selection function locates the data through the DN and return the address of the data to other functions.

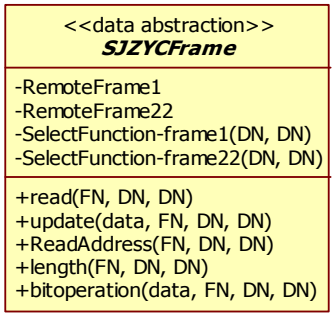

Figure 3 Class diagram of SJZYC Frame class.

The operation functions include reading function, updating function, reading address function, length function and bitwise operation function. These functions are external interfaces of the class.

Take the reading function for example. The FN and DN are the parameters of this function. When the reading function is used, it calls the selection function by the switch statement and the FN. The selection function called by the reading function locates the data by DN and return the pointer of the data. Finally the reading function reads the data to accomplish the operation. Others operation functions use the same way to locate the data. This selection structure locates the data by selection function and number, and realizes the encapsulation of information.

B. Device Interface Class

In the software design, device interface class is usually used to implement the functions related to external devices. It hides the details of device information, so the software system cannot be affected by replacing devices. The operations of the device interface class include initializing, reading or writing.

Take the Device_ALT1_Interface class for instance. This class is related to the Serial Port device. The operation functions include the alt_ COM_ ini function and alt_ get data function. The alt_ COM_ ini function is designed to initialize the device and the interior variable. The alt_ get data function is used to read data from the serial ports.

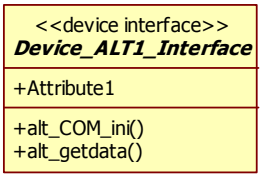

Figure 4. Class diagram of Device_ ALT1_ Interface class.

C. Task Class

The task class is designed to meet the task requirements of software system. Take the TaskReadData_HighCAN1 class for example. It is a data exchange class and calls the Device_ CAN_ Interface class to receive relevant data. The operation function of this class receives and stores the data in the temporary format, then converts the data from temporary storage format to standard storage format. After accomplishing the conversion, according to the type of the data, the homologous message is sent to data manipulation class. Then data manipulation class implements the operation on the data. We evaluated this increase on the 48 UML models and found that five degrees of neighbors involve between several hundred and several thousand model elements depending on the model. It was not practical for a designer to consider these many model elements to understand the impact of a single design change. These models were very diverse in domain and size. Figure 5 depicts the sizes of the models which cover the entire spectrum from small to very large models of up to 36,000 model elements.

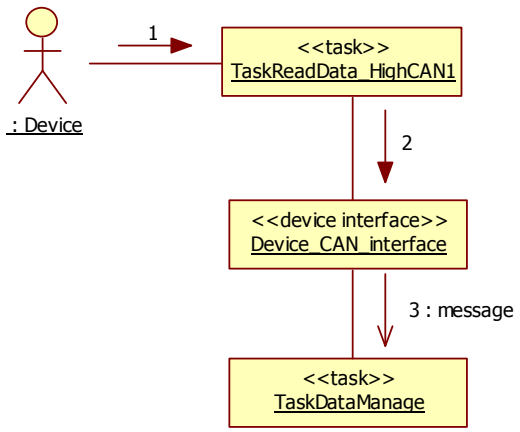

Figure 5. Collaboration diagram of the Task Read Data_ High CAN1 class.

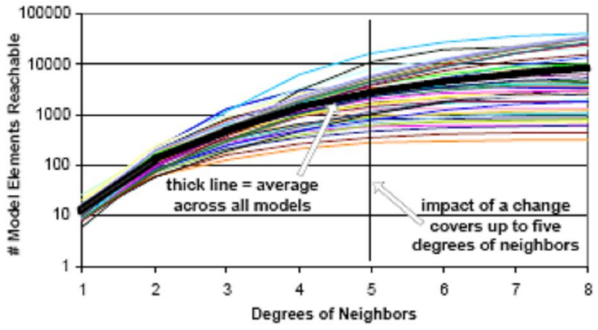

Figure 6. Exponential increase of number of model elements

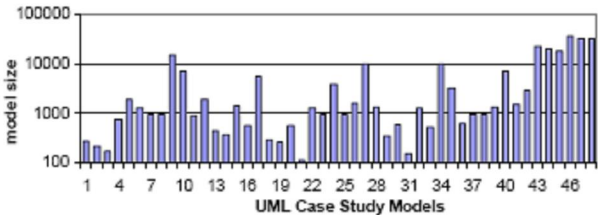

Figure 7. Sizes of the 48 UML M

7. Conclusion

Stratosphere airship is a new kind of flying vehicle which has attracted worldwide developing interests for its great potential. Because it needs to operate automatically for a very long time, **t**he UML is a standard visualization modeling language and its abundant diagrams are very helpful to describe and model the Object-Oriented system. This paper introduces the Object-Oriented design of the management software for the avionics system of stratosphere airship based on the UML. After analyzing the requirements of system, the framework of the software and detail class models are depicted. Although the detailed design of software is finished, there is still a lot of work to do. The future work is to optimize the design of the task class, improve the quality and reusability of codes, and accomplish the system testing.

Acknowledgements:

Authors are grateful to Science and Research Branch ,Islamic Azad University,Fars, Iran, for financial support to carry out this work.

Corresponding Author:
Afshin Shaabany
Science and Research Branch
Islamic Azad University
Fars
, Iran.
E-mail: afshinshy@yahoo.com

References

[1] Selic B, Gullekson G, Ward P. Real-Time Object-Oriented Modeling. John Wiley and Sons. New York. 1994.

[2] Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide. Addison-Wesley. 1999; 7-79.

[3] Guoshun S. Design of the UML modeling technology in software project. Software Guide. 2010; 9(8).

[4] Bikander M. Graphical Programing Using UML and SDL. IEEE Computer. 2000; 30-35.

[5] Kefu G. The research of application of UML., Journal of Wuhan Bioengineering Institute. 2006; 2(3): 2006.
[6] Hassan G. Designing Concurrent, Distributed , and Real-Time Applications With UML. 2004; 300-326.
[7] Bruce P.D. Real-time UML-Developing Efficient Objects for Embedded Systems. Addison Wesley. 1998.
[8] Ganssle J. Navigating through new development environments. Embedded Systems Programming. 1999; 22-30.
[9] Selic B. A generic framework for modeling resources with UML. IEEE Computer. 2000; 64-69.
[10] Atkinson. Meta-Modeling for Distributed Object Environments. First International Workshop on Enterprise Distributed Object Computing. 1997; 90-101.
[11] Jong G. D. A UML-based Design Methodology for Real-Time and Embedded Systems. Proceedings of the Design Automation and Test in Europe (DATE). 2002; 776-778.

3/5/2011