Model Based Development of Spacecraft OBDH Software

Zhenhui Dong(✉), Yahang Zhang, Peiyao Yang, Yiming Liu, and Xuan Chu

Institute of Spacecraft System Engineering, Beijing, China
564760683@qq.com

Abstract. In view of the difficulties faced by the current document driven development mode of software, this paper systematically introduces the research work of the model driven development mode of spacecraft OBDH(On-Board Data Handling) subsystem software. The business requirements of the OBDH subsystem software are modeled by Simulink, forming the OBDH subsystem software model architecture. After completing the model design of the basic functions such as remote control, telemetry, bus management and satellite time management, the model is simulated and verified, and the code is automatically generated. Finally, the hardware in the loop test is completed. This research has established a complete business chain of "requirement design - requirement modeling - model design and model in the loop - code automatic generation - hardware in the loop" for the spacecraft OBDH subsystem software, which has laid a foundation for model driven development of future OBDH software expansion functions.

Keywords: Model · Software · Spacecraft · OBDH

1 Introduction

In the traditional software development process, the design results of each stage are presented in the form of documents and transferred between stages, thus forming the document-based software development mode. However, with the increasing complexity of spacecraft software, the traditional document-based software development mode is facing more and more obvious difficulties, including backward representation, lack of early verification means, and difficulty in reuse.

For the above reasons, model driven software development has gradually attracted attention in recent years. Model Based Development (MBD) [1] refers to a software development method that takes Model Driven Architecture (MDA) [2] as the guiding ideology, takes models as the core of software development, and guides software understanding, design, construction, development, operation, maintenance and modification. Model driven development also makes the information transfer between each stage of software development process from the past documents to models, avoiding the huge pressure and island phenomenon caused by the need to synchronize design documents in software development.

R. Li et al. (Eds.): MOBILWARE 2022, LNICST 507, pp. 117–124, 2023.
https://doi.org/10.1007/978-3-031-34497-8_10

2 Design Idea

The functions of the OBDH subsystem software include managing the whole satellite bus network, telemetry, remote control, program control and satellite management [3]. The software is generally a multi process software based on the real-time operating system [4]. The OBDH subsystem is a typical discrete system and a strong real-time multitask system. Operations with high real-time requirements usually need to be handled in the interrupt processing program.

There are two main goals for the development of model driven OBDH software: 1) Build a universal spacecraft OBDH software architecture model and model the main business requirements. At the same time, the trusted model library is accumulated on the basis of software components to form organizational assets; 2) Find a model-based software development process suitable for spacecraft OBDH software, open up the whole process development link of model-based software development, build a prototype system of model-based software development mode, and objectively evaluate the impact of model-based software development mode on spacecraft OBDH software development efficiency and quality.

The overall design idea is divided into the following four parts. Each research content is progressive in order. The above research content is used as input, and the results are provided to the next research content, thus forming a research process of progressive and clear research purpose.

2.1 General Functional Requirements Modeling

The goal of conventional functional requirements modeling of spacecraft OBDH software is to quickly determine the logic and basic functions of the requirements object, establish the connection between the requirements and the model, and do not need to go deep into the internal details of the model. At this stage, it is required to quickly verify whether the data interface, input and output between the requirements and the complete system are reasonable, and determine the main business functions. Conventional function requirement modeling of spacecraft OBDH software mainly completes the abstraction and definition of business functions, and serves as the input and basis for subsequent modeling. The specific business logic and model architecture will be implemented in the subsequent stages after the rationality of the system layer is verified.

2.2 Model Architecture Design

After completing the modeling of the conventional functional requirements of the spacecraft OBDH software, this research starts the design and development of the spacecraft OBDH software model architecture, and develops a universal OBDH software architecture model that includes telemetry, remote control, bus communication management, satellite time management, memory read and other functions common to the spacecraft OBDH software. Based on the actual project needs and the technical characteristics of OBDH software, this research has built three different forms of software architecture models: 1) general software architecture model, in which component software can be

integrated. The architecture has complete functions, and the model is simulated and verified. During the process of building the model architecture, the special model library for OBDHs is extracted and generated; 2) The general software architecture model and the self-developed general test software are connected through TCP/IP network, supporting telemetry downlink, receiving remote control commands, and supporting bus data simulation; 3) The general software architecture model and the underlying driver and operating system interface are jointly compiled through Makefile, and then the binary executable file can be directly downloaded to the hardware device or virtual simulation test software (such as the software test platform) for running.

2.3 Function Module Encapsulation and Model Library Construction

Model library organizes many functional models according to a fixed structure, and effectively manages and uses each functional module through a unified model base management system. From the perspective of software, each functional model in the model library is a model architecture that can be reused and assembled into an application system [5, 6]. It can be seen that the common model base is an important achievement of unified standardization construction. Through the modeling of business function requirements and the transformation of the component interface of OBDH software, a universal model library of OBDH software is formed. In the subsequent modeling work, the model library can be directly used in Simulink or call module encapsulation through C Caller to facilitate users to build new OBDH software architecture models.

2.4 Automatic Generation of Software Code

After building the spacecraft OBDH software model, the model-based spacecraft OBDH software code generation is carried out. Specifically, code generation based on Simulink model can use Simulink's own RTW tool to generate intermediate files in TLC format first, and then generate target C or C + + files according to the target file type configured by the user. At the same time, combined with the compiler set by the user, the target file can be compiled to generate an executable file [7].

During the code generation process of Simulink, for the configuration of peripheral models, some options in the sub model need to have the same configuration, such as optimization options and code generation format. However, for the configuration in Code Generation, such as whether to generate reusable code, and the way to generate reusable code peripheral interfaces, you only need to configure in the peripheral model, while the configuration in the internal sub model has no impact on the results. To ensure the consistency, effectiveness and efficiency of generated code, the generated code needs to be configured locally and follow certain configuration specifications. It includes simulation configuration, hardware environment configuration, overall configuration, report generation configuration, code annotation format configuration, interface configuration, etc. At the same time, in order to improve the readability, migration and reusability of generated code configuration, code style configuration, code template configuration, data type replacement configuration, and memory configuration are also required.

3 Model Based Development of OBDH Software

3.1 Software Requirements Modeling

This work adopts the architecture design tool System Composer in Simulink software, which is used for requirements modeling in the early stage of the project. First, use the Requirement Editor to sort and decompose the requirements, then use System Composer to model the software requirements, and establish the link between the requirements and the model. The correspondence between the requirement model and the design model can be realized by dragging the requirement items directly, and the completeness of the requirement realization can be verified. Figure 1. Shows the requirement model framework design and interface design.

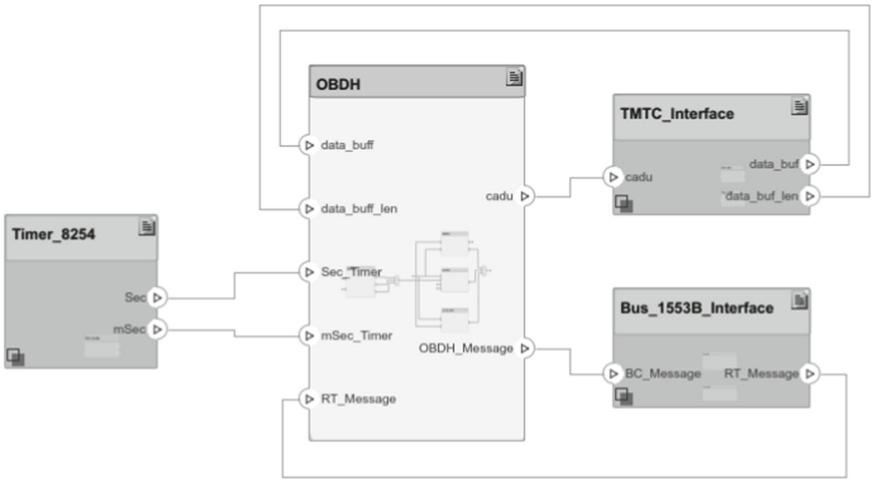


Fig. 1. Requirements model framework and external interface

3.2 Functional Model Architecture

The universal software function model architecture of OBDH is implemented by using Simulink/Stateflow toolbox, Simulink model library and Matlab M function. This research only models and generates code for the application layer of the OBDH software, and conducts some special processing and integration for the underlying hardware driver and the interface between the application software and the operating system. In addition, considering that spacecraft OBDH software has generally established an organizational asset library of software components, this part of code can be directly integrated into the entire software model without modeling and code generation. The general software architecture model includes telemetry, remote control, bus communication management, satellite time management, memory read and other requirements. Figure 2. Shows the overall architecture of OBDH model.

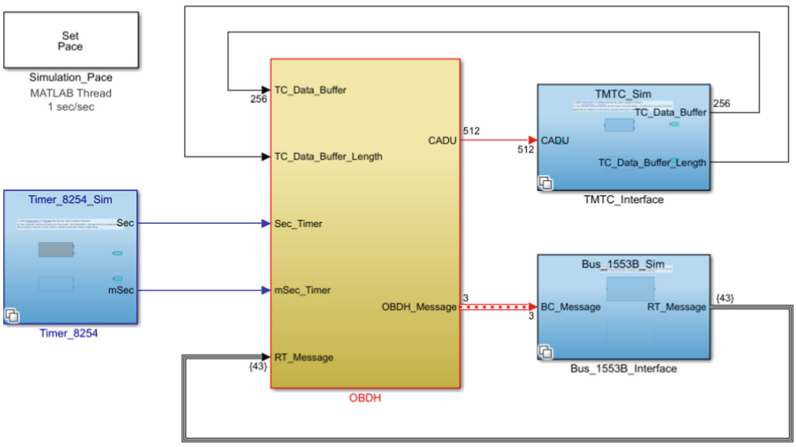


Fig. 2. Overall architecture of OBDH software model

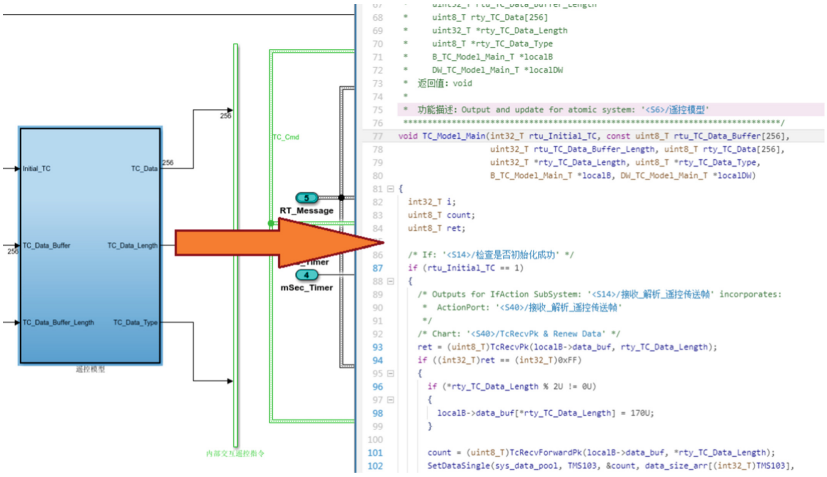
3.3 Generate Code

Model driven software development needs to establish code generation configuration items according to the requirements of code generation. In this research, two different configuration sets are established, namely, the configuration set for simulation and the configuration set for code generation, so that the functions of model simulation and code generation are realized in a set of models. Users can browse and modify the contents of these two configuration sets through the Model Explorer tool.

After the simulation and verification of the OBDH software model is completed, C code can be generated. The generated C code can view the generated report and track the relationship with the model, as shown in Fig. 3. After the source code is generated according to the user's needs, the generated source file can be packaged, which facilitates the generation of source code to other compilation environments for compilation, running and debugging operations.

The automatically generated code is integrated with some handwritten codes (including software components, general library functions, and hardware drivers). By specifying and setting the main functions generated by different subsystems to achieve multitask scheduling, configuring Makefile files, adding macros required for compiling models, and finally jointly compiling the automatically generated code and handwritten code and downloading them to the hardware environment or virtual software testing environment for running.

In this study, the C code generated by model driven method totaled 5813 lines, including 12 C source files and 16 header files. There were 82 function modules in total, accounting for 21% of the engineering code (about 22000 lines of component, hardware driver and other handwritten code). There are two reasons why the code generated by the model accounts for a relatively low proportion in the total code. One is that this research mainly aims at technical verification, so only some software functions are used as examples for modeling and code generation. The other is that some models in this research are only used for model simulation and verification, not for code generation.




```
*     uint8_T rty_TC_Data[256]
*     uint32_T *rty_TC_Data_Length
*     uint8_T *rty_TC_Data_Type
*     B_TC_Model_Main_T *localB
*     DW_TC_Model_Main_T *localDW
* 返回值: void
*
* 功能描述: Output and update for atomic system: '<S6>/遥控模型'
*/
void TC_Model_Main(int32_T rtu_Initial_TC, const uint8_T rtu_TC_Data_Buffer[256],
                   uint32_T rtu_TC_Data_Buffer_Length, uint8_T rty_TC_Data[256],
                   uint32_T *rty_TC_Data_Length, uint8_T *rty_TC_Data_Type,
                   B_TC_Model_Main_T *localB, DW_TC_Model_Main_T *localDW)
{
  int32_T i;
  uint8_T count;
  uint8_T ret;

  /* If: '<S14>/检查是否初始化成功' */
  if (rtu_Initial_TC == 1)
  {
    /* Outputs for IfAction SubSystem: '<S14>/接收_解析_遥控传送帧' incorporates:
     *  ActionPort: '<S40>/接收_解析_遥控传送帧'
     */
    /* Chart: '<S40>/TcRecvPk & Renew Data' */
    ret = (uint8_T)TcRecvPk(localB->data_buf, rty_TC_Data_Length);
    if ((int32_T)ret == (int32_T)0xFF)
    {
      if (*rty_TC_Data_Length % 2U != 0U)
      {
        localB->data_buf[*rty_TC_Data_Length] = 170U;
      }

      count = (uint8_T)TcRecvForwardPk(localB->data_buf, *rty_TC_Data_Length);
      SetDataSingle(sys_data_pool, TMS103, &count, data_size_arr[(int32_T)TMS103],
```

Fig. 3. Model code tracking

4 Simulation Verification

The spacecraft OBDH software model is based on the Matlab/Simulink standard model development environment. Through the Intermediate communication model, the interface between the spacecraft OBDH software model and the traditional ground test software is connected. Based on the existing mature ground test software, the rapid verification of the model in the loop and hardware in the loop of the spacecraft OBDH software model is realized, and the entire model-based development process is fully connected. The environment of the spacecraft OBDH software based on this method is shown in Fig. 4.

This research establishes a complete process from requirements management, model architecture, model design, code generation and simulation testing for spacecraft OBDH software. The model test environment is built, and the model closed-loop simulation test is carried out, and the consistency between the automatically generated code and the running results of the model is verified. The technical indicators achieved in this study are shown in Table 1.

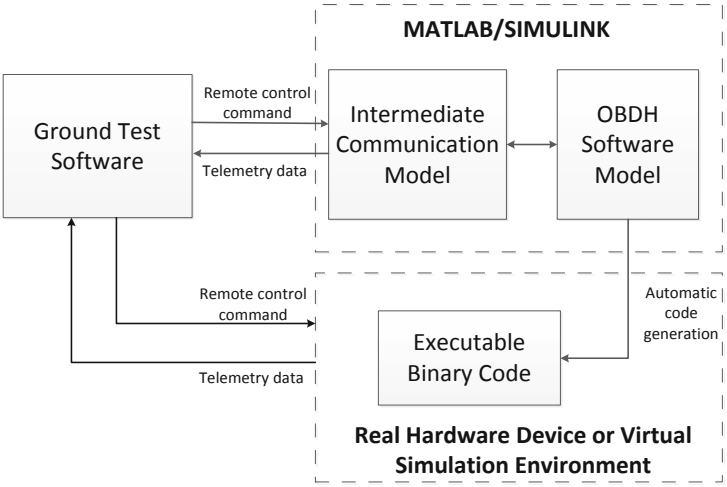


Fig. 4. Model-based development environment of OBDH software

Table 1. Technical indicators reached.

Index name	Achievement of indicators
Functional items covered by the model	A general model of OBDH software is established, which covers 5 types of OBDH software models and 3 types of software interface models
Automatically generate code	Automatic code generation time is within 30 s
Execution efficiency of automatically generated code	The SRAM used by the automatically generated code in the running process does not exceed 500KB; The average running time of the software after time equalization is less than 50 ms

5 Conclusions

This research changed the development mode of spacecraft OBDH software from the traditional handwritten coding development mode to the model driven development mode, and verified various technologies such as software requirement modeling, general architecture model, general model library, model simulation and verification, and software automatic generation. By connecting the existing general software test platform with the model, the efficiency of model simulation verification is improved. After model simulation verification, embedded software code is automatically generated, which can be directly run on the OBDH hardware equipment or virtual simulation platform after compilation. The simulation and verification of model can reuse existing organizational assets. This research has formed a complete model driven business chain of spacecraft OBDH software, laying a foundation for the subsequent model-based development of complex functions of OBDH software in the future.

References

1. Furong, L.: Model-based developing approach for airborne software of commercial engine. Process Autom. Instrum. **38**(6), 26–30 (2017)
2. Li, Y., Shenglin, G., Geng, C., Lei, L.: Model development environment research of embedded real-time software. Comput. Sci. **39**(z3), 226–229 (2012)
3. Xiaongwen, H., Meng, Z.: Application method of telecommand and telemetry packet utilization standard in spacecraft. Spacecraft Eng. **21**(3), 54–60 (2012)
4. He, X., Sun, Y.: engineering realization of software in central terminal unit of satellite data management system. Spacecraft Eng. **16**(5), 47–53 (2007)
5. Xiaogang, D., Jingsong, L., Dianyou, W., Chuan, L., Chaohui, C.: Model architecture based development method for spacecraft control software. Aerospace Control Appl. **47**(2), 55–62 (2021)
6. Wenquan, W., Kepu, S., Yong, W., Wei, X.: Airborne embedded software application based on MDA. Comput. Technol. Dev. **8**(23), 145–148 (2013)
7. Jiali, R., Haiyan, C.: Research of code auto-generation and integration for the embedded software. J. Taiyuan Univ. Technol. **44**(4), 518–521 (2013)