Spacecraft Sun Acquisition Software Component Identification and Description

Xingsong Zhao[1, a, *], Bin Gu[1, b], Xiaogang Dong[1, c], Xiaofeng Li [1, d], Ruiming Zhong[1, e]

Beijing Institute of Control Engineering, Beijing, China, 100190[1]

zxssdu@126.com[a]; gubinbj@sina.com[b]; dongxiaogang@263.net[c]; li_x_feng@126.com[d]; ruiming_zhong@163.com[e]

Abstract. Software components are important reusable assets for spacecraft embedded software. Component development is an important task current now. Based on FODA (Feature-Oriented Domain Analysis), this paper analyzes the spacecraft software domain, identifies common requirements and variability requirements, defines basic components, aggregation components, and parameter CFG component, proposes a component interface determination method based on data flow diagram, proposes the design method of mode-state management behavior components, and gives detailed descriptions of a gyro data acquisition component and a mode-state management component. This paper effectively extracts 15 reusable components of sun acquisition software and has been applied in multiple projects.

Keywords: FODA, spacecraft embedded software, software reuse, software components

1 Introduction

Spacecraft embedded software presents the characteristics of increasing complexity, frequent requirement changes, shortened development cycles, and improved reliability. Safety, efficiency, and quality of software development need to be strengthened urgently. The low reuse rate of software assets is one of the main bottlenecks restricting the efficiency and quality of spacecraft software development, mainly because spacecraft software generally adopts customized methods for development, and lacks effective methods on description of requirements specification, refinement of domain commonality and variability.

Domain engineering is an important approach to build reusable software assets, and consists of three main activities: domain analysis, domain design, and domain implementation. Software engineering research at Carnegie Mellon University proposed the feature-oriented domain analysis (FODA) method [1] in 1990. It is widely used in commercial and defense fields. Software product line technology [2-3] is an extension of the FODA method. Literature [4] describes the technical evolution of industrial product lines under concurrent systems. Software component [5] is a key element of software product line technology.

A software component is an entity with a certain independent function, and its definition is not unique. Components focus on specific functions, and component-based software development is an important technique for software reuse. Literature [5] describes component development method in a collection of software product line.

Component-based software engineering (CBSE) is a software engineering approach that decomposes a system's functions based on separated concerns. This approach defines and produces many loosely coupled individual components, and then combines the components into a system. The literature [6] proposes a method for component classification. F Primer is an application of component development architecture in embedded software, but does not provide a component decomposition method. A large number of similar functions exist in different projects of spacecraft embedded software, and they can be developed as a component library. Document [7] describes the reusable component measurement methods in existing component systems. Literature [8] describes software refactoring techniques based on FODA tools.

In the literatures above, the problem of how to identify components in spacecraft software of a tightly coupled functions and how to effectively apply component technology in spacecraft software has not been solved. This paper analyzes the domain characteristics of the spacecraft sun acquisition software, including functional features, behavioral features, and data features, develops the reusable components, identifies the relationship between the components, and builds software assets in the spacecraft field.

2 Sun acquisition Mode (abbreviated as SAM)

The sun acquisition mode belongs to the safety critical functions of the spacecraft control system and is used to ensure the energy supply of the spacecraft in emergency. The main function of this mode is to achieve stable state of the satellite in a sun-oriented attitude, which is usually entered during the entry phase or in case of spacecraft fault.

Common components of spacecraft control systems include: sensors (for spacecraft attitude measurement, mainly sun sensors, earth sensors, star sensors, gyroscopes), controllers (controllers connect on-board sensors and actuators), actuators (generate reaction thrust or control torque, mainly thrusters, momentum wheels, CMG, magnetic torques).

In the sun acquisition mode, the control system calculates the attitude of the spacecraft using the data of the gyroscope and sun sensor, and uses the thruster and momentum wheel to control the spacecraft rotating along the pitch or roll axis of the spacecraft, which eventually allows the sun sensors to find the sun and maintain an attitude pointing to the sun.

SAM consists of five sub-modes: rate damping (RDSM), pitch search (PASM), roll search (RASM), cruise search (CSM), and stop control (NOCTRL). The process of each sub-mode includes five steps: data acquisition, data processing, attitude estimation, attitude control, and control data output. The types, quantities, and installation locations of the sensors and actuators configured by each project are different, and the functions of each sub-mode are also different, which leads to differences in the specific processing of the SAM sub-mode, as shown in Table 1 for details.

Table 1. Sub-mode process description table

No.	SAM Process	Sub-mode	Main functions and mode transfer conditions	Parts used
1	Data Acquisition Data Processing Attitude calculation Attitude control Mode Management	RDSM	control the three-axis angle velocity of the star to a certain range, and transfer to PASM when success or timeout	Gyroscope, Sun Sensors, Thruster
2		PASM	control the satellite to rotate along the pitch axis at a certain angle velocity, search the sun and turn into CSM when success, or turn into RASM when time out	Gyroscope, Sun Sensors, Thrusters, Momentum Wheel
3		RASM	control the satellite to rotate along the scroll axis at a certain speed, and turn into CSM when success, or the first time out, turn into PASM, more than 2 time out turn into NOCTRL	
4		CSM	continuously track the sun, control the attitude error of the satellite, turn into other mode by ground commands	
5		NOCTRL	stop control, actuator no longer output, turn into other mode by ground commands	

This paper will analyze and identify functional components, behavioral components, and data ports from the aspects of traditional requirements analysis: functional requirements, behavioral requirements, and data requirements.

3 Functional components

The FODA method is used to analyze and identify common and variable features in the domain. Features describe the functional and quality characteristics of a system. The FODA feature analysis method performs feature decomposition according to the top-down hierarchy to generate a feature tree. The feature tree presents system features according to a hierarchical organization. A feature can be decomposed into multiple sub-features, which can be mandatory, optional, or replaceable. In this paper, the commonality and variability of different spacecraft SAM software are identified by functional feature decomposition, and the functional components of the system are extracted based on the feature decomposed.

3.1 Domain Analysis meta-model

The meta-model of the feature model is given in the literature [3], describing two basic types of relationships between features: refining relationship and constraint relationship, as detailed in Figure 1 [3].

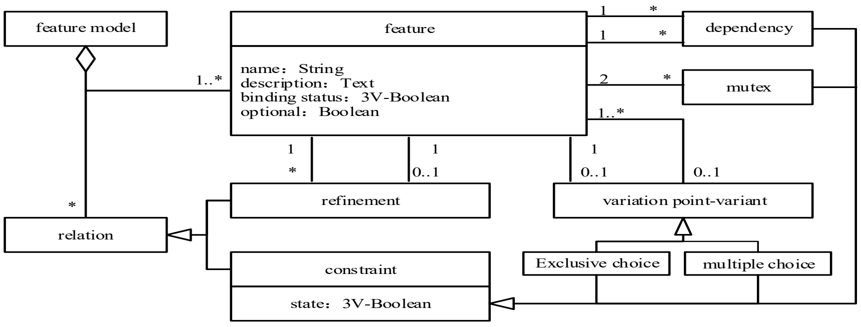

Fig. 1. Feature model meta-model [3]

3.2 Feature tree of SAM software

Based on the FODA feature meta-model, the functional feature tree of SAM software is constructed. With the sun acquisition function as the root node, the functional requirement feature decomposition is carried out, and each feature is refined one by one in a left-to-right, top-down manner until it is decomposed to the smallest functional unit. The advantage of a tree diagram is that it can be used both to describe the relationship between decomposition and aggregation, and to represent the order of calls between function-points through the order in which the tree nodes are accessed. The functional feature tree of SAM software is shown in Figure 2. (notes: SADA in Figure 2 is abbreviation of Solar Array Drive Assembly)

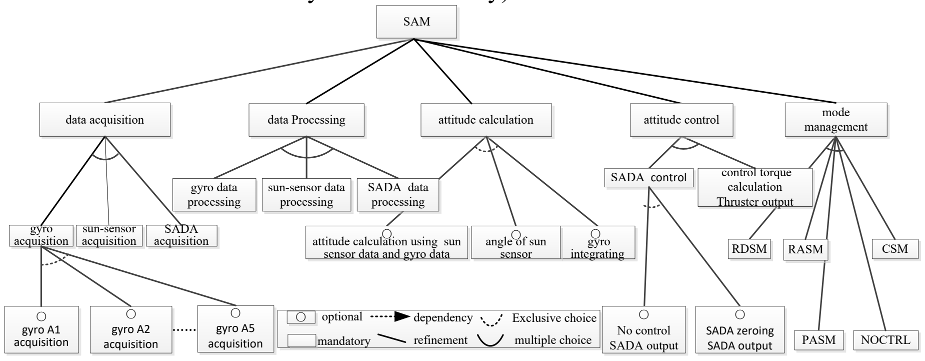

Fig. 2. Functional feature tree of SAM software

3.3 Functional Components of SAM

A component is an entity that encapsulates one or more program modules. Components emphasize encapsulation, and use ports for interaction. The three basic elements of a component are components, interfaces, dependencies [10].Component diagrams in UML consist of components, interfaces, relationships, ports, and connectors [10]. The bottom-level features of the feature tree represent the minimum functional unit (usually the element in the data dictionary) in the domain as the basic components. Based on the feature tree, this paper extracts software components as follows:

(1) Basic component: According to the refinement results, the basic data of a minimum functional unit that needs to be provided for use by multiple components is developed as an independent component and is called a basic component.

(2) Aggregation component: If the child nodes under the same parent node are closely coupled, instead of creating a child node component, a parent node component is created, called an aggregate component.

(3) Parameter CFG component: If there are only differences in configuration parameters between components, they can be extracted into one component.

Based on the feature tree, five typical projects of different platforms (A1, A2, A3, A4, and A5) are selected as the research objects, the common characteristics and variable characteristics of SAM software of each project are specifically analyzed. A matrix analysis table is established based on the four elements (system composition, functional logic, request service interface, service interface). Table 2 shows the feature analysis results of the five projects.

Table 2. Table of commonality and variability analysis of SAM requirements (GEO satellite as an example)

Projects / Elements	**A1, A2, A3, A4, A5**
System Composition	Gyro: 5 projects with different gyro-type, quantity, installation, communication protocol. Thrusters: 5 projects with the same thruster product type, but with different quantity and installation
	Sun Sensors: 5 projects with the same product type, quantity, installation, and communication protocol. SADA: 5 projects with the same product type, quantity, installation and communication protocol.
Functional Logic	Data acquisition: 5 types of gyro data acquisition, one kind sun sensors data acquisition, one kind SADA data acquisition. Data processing: 5 types of gyro data processing, 3 types of attitude calculation. Attitude control: 5 types of control torque calculation, 2 types of SADA control. Model management: 5 types of RDSM, PASM, RASM, CSM and one kind of NOCTRL.
Control output	Thruster output: 5 types of thruster combinational logic.
	SADA control: 2 types.
Component Extraction	Each minimum functional unit as one basic component according to the result of refinement. In accordance with the aggregation relationship, two SADA controls at are both SADA controls and are aggregated into one SADA control component.

	A tight coupling between the 5 sub-modes' management according to the aggregation relationship, aggregated into one component, as detailed in section 4.
Total number of recognition Components	5 gyro data acquisition components; 3 data processing components; 1 SADA control component, 5 control torque calculation components, and 1 sub-modes management component (see Section 4 for details).

3.4 Component data port

A dependency is said to exist between two components if there is data interaction between the two, or if one component calls an interface provided by the other component. The feature tree does not give the input and output interfaces of each component after feature decomposition. Based on the feature tree, this paper uses the UML's data flow diagram (DFD) description to analyze the dependencies between sun acquisition software components, exporting the corresponding input and output interfaces. Data storage represents the interaction data, see Figure 3 for details.

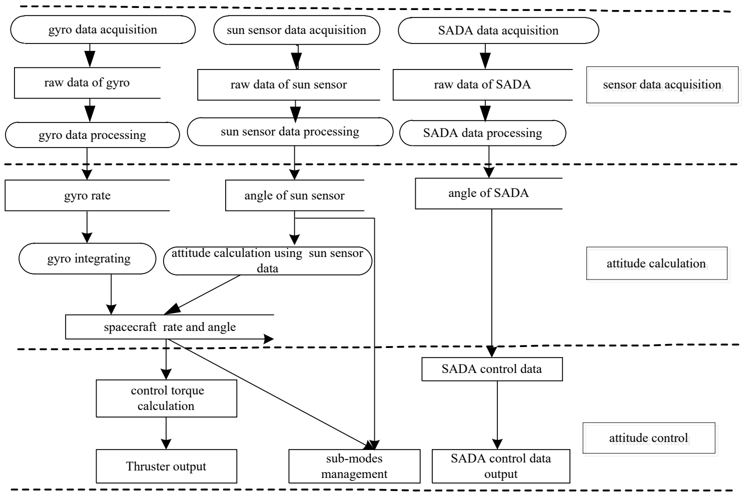

Fig. 3. Data flow diagram of SAM software

4 Behavior components

Behavior describes the state and transition of a system, and state machines are a common example of behavior modeling, the basic elements of which are states and state transitions. For SAM software, each sub-mode shows the current working state of the system, and one sub-mode is transferred to another sub-mode after meeting certain conditions, and the management components of SAM software sub-modes can be described by a state machine.

For 5 project samples above, the sub-modes management functions of them are similar, the request interface and service interface of them are same, and the only difference of them is parameter-values of transition conditions. According to the aggregation relationship, the sub-modes management of the 5 projects can be abstracted into an independent component, which can be expressed by a state machine.

The UML state diagram in Figure 4 depicts the SAM mode management component, and the sub-modes state transition conditions in the figure are as follows:

Condition C1 (RDSM →PASM): the absolute values of the three-axis angular rates w1 , w2 , w3 are continuously tw less than TH or the duration time of the RDSM sub-mode is greater than tRDSM;

Condition C2 (PASM→RASM): the sun cannot be found in continuous time tRASM in PASM;

Condition C3 (RASM→PASM): the sun cannot be found in continuous time tPASM in RASM;

Condition C4 (PASM →CSM): the sun is found and lasts longer than tsun;

Condition C5 (RASM →CSM): the sun is found and lasts longer than tsun;

Condition C6 (RASM→NOCTRL): the sun is still not found after 2 times of the PASM;

Condition C7 (NOCTRL→ End): ground remote control command.

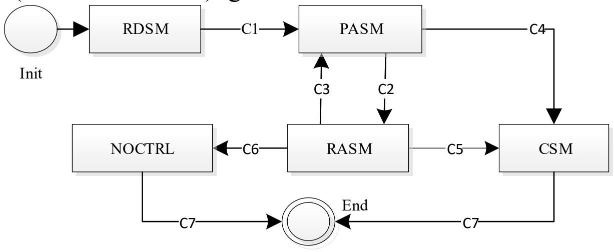

Fig. 4. Sun acquisition sub-modes state diagram

The finite state machine (FSM) is a five-tuple, and the finite state machine M = (Q, Σ,δ, q 0,F), and the corresponding finite state machine in Figure 4 is described as follows.

(1) States set Q = {RDSM, RASM, PASM, CSM, NOCTRL};

(2) Input set Σ = {Condition 1, Condition 2, ..., Condition 7};

(3) State transition function δ=(Q x Σ->Q), for q∈Q, c∈Σ, next state q'= δ(q, c);

(4) Initial state q0 = {Init};

(5) End state F = {End}.

The 5 project sub-modes management finite state machines, differ only in state transfer condition parameters' values of tw, TH, tRDSM, tRASM, tPASM, tsun are different. The sub-mode management behavior component is described in Section 5.2.

5 Component description

This paper selects gyro data acquisition Function component and sub-modes management behavior component as examples of component descriptions.

5.1 Gyro data acquisition component description

Analyzing the gyro data acquisition protocols of the selected 5 projects, the main features of gyro data acquisition are identified as follows: gyro-type, gyro-protocol (gyroFetchCmdBuffer, gyroDataBuffer), device driver (such as 1553B, UART)), waiting time for gyro data transfer (labeled as TConstraint). As a result, the component of gyro data acquisition is described as: GetGyroData(GyroType, gyroFetchCmdBuffer, gyroDataBuffer, device driver, TConstraint).Five components of gyro data acquisition in the selected 5 projects are showed as follows:

GetGyroData(GYROTYPE_A1, gyroFetchCmdBuffer1, gyroDataBuffer2, 1553B, 7ms);

GetGyroData(GYROTYPE_A2,gyroFetchCmdBuffer1, gyroDataBuffer2, 1553B, 7ms);

GetGyroData(GYROTYPE_A3,gyroFetchCmdBuffer1, gyroDataBuffer2, 1553B, 2ms);

GetGyroData(GYROTYPE_A4,gyroFetchCmdBuffer1, gyroDataBuffer2, UART, 5ms);

GetGyroData(GYROTYPE_A5, gyroFetchCmdBuffer1, gyroDataBuffer2, UART, 0ms);

Table 3 shows the component gyro data acquisition of GYROTYPE_A1, and Figure 5 gives its UML description.

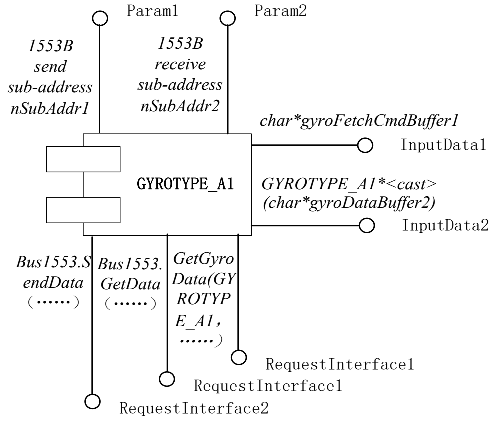

Fig. 5. GYROTYPE_A1 Component

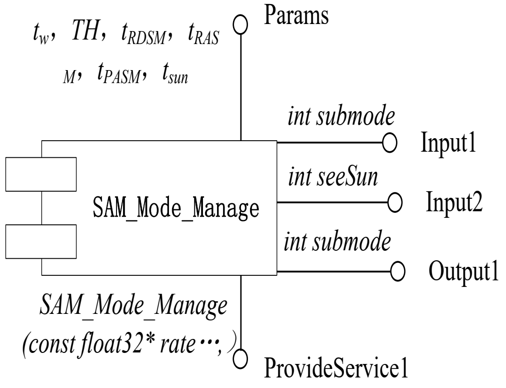
t_w, TH, t_{RDSM}, t_{RASM}, t_{PASM}, t_{sun}
Params
int submode
Input1
int seeSun
Input2
int submode
Output1
SAM_Mode_Manage
SAM_Mode_Manage (const float32* rate···,)
ProvideService1

Fig. 6. SAM_Mode_Manage Component

Table 3. GYROTYPE_A1 Component Description Table

Keyword: ***GYROTYPE_A1, 1553B***			
1. Functional behavior			
send *gyroFetchCmdBuffer1* ***to*** *Gyro_GYROTYPE_A1* ***by*** *B1553*			
delay 5ms			
receive *gyroDataBuffer2* ***from*** *Gyro_GYROTYPE_A1* ***by*** *B1553*			
2. Main performance			
$T_{Constraint} >= 5ms;$			
3. Interface			
-a input parameter			
Variable Name	Type	Dimension	Initial value
gyroFetchCmdBuffer1	*char**	None	*0x5555*
-b output parameters			
Variable Name	Type	Dimension	Initial value
*GYROTYPE_A1**	*struct**	/	*0*
4. Request Service			
Bus1553.SendData(int nSubAddr1,char B1553cmdBuffer1,int nLen1)*			
Bus1553.GetData(int nSubAddr2,char B1553dataBuffer1,int nLen2)*			
delay(int ms)			
5. Services Provided			
GetGyroData(GYROTYPE_A1, gyroFetchCmdBuffer1, gyroDataBuffer2)			
6. Configuration parameters			
1553B transmit sub-address nSubAddr1; *1553B receives sub-address nSubAddr2*			

5.2 Description of SAM mode management components

As described in Section 4, the finite state machine describes the characteristics and component model of SAM mode management. Table 4 shows a detailed description of the SAM mode management component, and Figure 6 shows its expression in UML.

Table 4. SAM_Mode_Manage Component Description Table

Keywords: ***SAM_Mode_Manage, RDSM, PASM, RASM, CSM, NOCTRL***			
1. Main functions			
The state switching between the 5 sub-modes detailed in Figure 4			
2. Main performance			
Execution time <= 0.2ms;			
3. Interface			
-a input parameter			
Variable Name	Type	Dimension	Initial value
rate	float64*	rad/s	0
seeSun	int	None	0
submode	int	None	0
-b output parameters			
Variable Name	Type	Dimension	Initial value
submode	int	None	0
4. Request service			
None			
5. Provide services			
void SAM_Mode_Manage(const float32 rate, int seeSun, int submode)*			
6. Configuration parameters			
t_w , TH, t_{RDSM} , t_{RASM} , t_{PASM} , t_{sun}			

6 Conclusion

In this paper, the spacecraft sun acquisition software is taken as the research object, and the functional requirements are decomposed using FODA, the basic components, aggregation components and parameter configuration components are defined. This paper proposes the identification method of the components, the construction method of mode- management behavior component, and the interface recognition technology based on data flow. This paper extracts 15 reusable components of SAM software, which are applied in multiple models. The next step will be further research into software-reusable assets in other modes of the spacecraft control system, as well as component-based software synthesis techniques.

Acknowledgement

We would like to thank the National Natural Science Foundation of China major project "Basic Theory and Method of Intelligent Synthesis of Embedded Software (ZY0100270903010073)" for funding this paper.

References

1. Kang K C, Cohen S G, Hess J A, et al. Feature-oriented domain analysis feasibility study. Technical Report. Carnegie Mellon University, 1990.
2. U.Devi, N. Kesswani, A. Sharma. A Diverse View on Feature-Oriented Programming in Software Product Line. UGC Care Journal,2020, 40(18):1950-1957.
3. Zhang W, Mei H. Feature-oriented Software Reuse Technology—State of the Art (in Chinese). Chin Sci Bull (Chin Ver.), 2014, 59: 21-42.
4. D Hinterreiter, L Linsbauer, K Feichtinger, H Prähofer and P Grünbacher. Supporting feature-oriented evolution in industrial automation product lines. Concurrent Engineering: Research and Applications.2020, Vol. 28(4): 265–279.
5. Chen Qian. Feature-oriented component-based development of whole software product families using enumerative variability. Thesis, the of Manchester university, 2019.
6. B.Jalender, Dr.A.Govardhan. A Novel Approach for Component Classifications and Adaptation Using JALTREE Algorithm. IJCSNS International Journal of Computer Science and Network Security, 2022, Vol. 22 (2):115-122.
7. RLB Jr, TK Canham, GJ Watney, LJ Reder, JW. Levison. SSC18-XII-04 F Prime: An Open-Source Framework for Small-Scale Flight Software Systems. 32nd Annual AIAA/USU Conference on Small Satellites (Small Sat2018).
8. N Padhy, R Panigrahi & S C Satapathy. Identifying the Reusable Components from component-Based System: Proposed Metrics and Model. Information Systems Design and Intelligent Applications, 89–99(2019).
9. S. Malathi, P. Sudhakar. Implementation of Software Refactoring Using FODA Tool. International Conference on Communication and Electronics Systems (ICCES2018).
10. Li lei, Wang Yangting. Object-Oriented Technology and UML. POSTS & TELECOM PRES, 2010.