Modeling and Analyzing Flight Control Software of Unmanned Aerial Vehicle Using UML and B Method

Jiufu Liu, Jianyong Zhou, Chunsheng Liu, Zhong Yang, Zhisheng Wang
College of Automation, Nanjing University of Aeronautics and Astronautics,Nanjing,China
Email:liujiufu2@yahoo.com,zhoujianyong4@126.com, liuchsh@nuaa.edu.cn,
yangzhong@nuaa.edu.cn,wangzhisheng@nuaa.edu.cn

Yifan Zhang
Institute of Information Engineering, Henan University of Animal Husbandry Economy, Zhengzhou, China.
Email:xxxwljys@126.com

***Abstract*—B is a formal method which enables the automatic generation of an executable code through successive refinements from an abstract specification. Unified Modeling Language (UML) specifications can be formally verified by analyzing the corresponding B specification, and integration of UML specifications and B method can overcomes the drawbacks of UML. In this paper the Class diagram of the flight control system is presented and each class operation is mapped to a B abstract machine. The flight control software behaviors are presented in the form of statecharts The B method is adopted to translate the statecharts into B specification of flight control software.Using UML and B method, flight control is refined and failure management is added. Finally proof obligations are presented to ensure the safety for the ertical control of artificial navigation of UAV.**

***Index Terms*—Flight control software;UML;Class operation; Statechart;B method;Refinement**

I. INTRODUCTION

Now designers of embedded systems[1] are faced with enormous challenges from the complexity due to the increase of design content, explosion of new features, ambiguous design parameters and evolving customer requirements. These challenges far surpass the technological capabilities of even the most excellent design teams. To our surprise, most of complex systems are still specified in a textual format. How to generate complete and unambiguous specifications is the most important problem for us to develop high quality software.

In conventional methods, specification errors are often identified only during final integration and system testing, but lots of studies have shown that the costs of correcting the errors during the integration is over 10 to 1000 times more than correcting them during specification.

Statemate is the most comprehensive graphical modeling and simulation tool for the rapid development of complex embedded systems on the market. Statemate creates a visual, graphical specification that represents the intended functions and behavior of the system being specified. This specification may be executed (graphically simulated) so the system engineer can explore what-if scenarios to determine if the behavior and the interactions between system elements are correct.

The graphical models are easily created using intuitive semantics and graphical notations that support a concurrent engineering process.

The modeling language is based on standard engineering diagrams with some UML diagrams[2]. Standard engineering diagrams include data and control flow diagrams, structure diagrams, truth tables, flowcharts and control law block diagrams. UML diagrams include use case diagrams, sequence diagrams and Statecharts. These diagrams, input via a menu-driven user interface, enable effective detailed communication of the specification to the engineers tasked with implementing and testing the system.

However, the fact that UML lacks a precise semantics is a serious drawback of object-oriented techniques based on UML. To remedy the drawback, the B language[3] is considered to be a promising approach to formalize UML specification. By formalizing UML specifications in B, one can use B powerful support tools like AtelierB, B-Toolkit to detect and analyze inconsistencies and defects inside UML specifications.

In this paper, a new approach for mapping statechart diagrams into B is presented, and only the event-based part of the statechart diagrams is considered. In Section 2 the B method is introduced. In Section 3 , Section 4, Section 5 and Section 6 a case study for flight control software is discussed. In Section 3 the Class diagram of the flight control system is presented. In Section 4 the statechart diagrams and B method are used to model one flight control system. In Section 5 we refine the a flight control system, accordingly new states and. events are added. In Section 6 proof obligations are proposed. Finally, in Section 7, conclusions complete our presentation.

doi:10.4304/jsw.9.4.800-806

II. B METHOD

B[3-5] is a formal software development method that covers a software process from specification to implementation. The B notation is based on set theory, the language of generalized substitutions and first order logic. Specifications are composed of abstract machines that are similar to modules or classes. Each abstract machine consists of a set of variables, invariance properties relating to those variables and operations. The state of the system, i.e. the set of variable values, is only modifiable by operations which must preserve the invariant.

To express the post-conditions of operations, B method provides the generalized substitutions, which can be used to specify the non-determinism (at abstract specification level) and also the determinism (at implementation specification level). This is a notablely different from Z and VDM, which use only logic expressions. The generalized substitutions provide a more familiar frame to specifiers by integrating the essential methodological aspects, such as invariant and refinement.

Refinement[6] [7] is the term given to the process of taking a specification through a sequence of design steps towards implementation. In general, a distinction is frequently made between procedural or algorithmic refinement, in which only the algorithmic component of an operation is refined. The other form of refinement is data refinement, in which the state of the machine is changed, that is, a set of variables is chosen to model the behaviour. The B method supports both procedural and data refinements.

A characteristic of the B method is that it has been designed to be easily automated. The generation of proof obligations (of the invariant preservation and of refinement correctness) obeys the simple rules that can be easily implemented in a piece of software. Furthermore, the support tools, such as AtelierB and B-Toolkit, provide utilities to discharge automatically and interactively the generated proof obligations.

Finally, beside the refinement, B also provide the structure like INCLUDES, IMPORTS, USES and SEES so that abstract machines can be composed in various ways. Thus, large systems can be specified in a modular way, possibly reusing parts of other specifications.

III. UML MODEL OF FLIGHT CONTROL SOFTWARE

Due to the complexity of flight control and mission management of unmanned aerial vehicle(UAV), the requirements of flight control system, such as the function, the reliability ,the adaptation and the cost, become higher and higher.The flight control system interacts with angle gynoscope, angle speed gynoscope, GPS, remote control/measure, elevator, rudder, left aileron, right aileron, engine. The Figure1. represents the context diagram of the flight control system component.

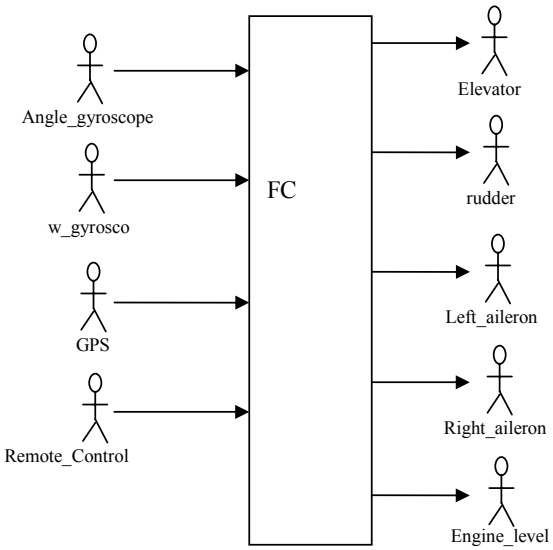

Figure 1. the flight control system component

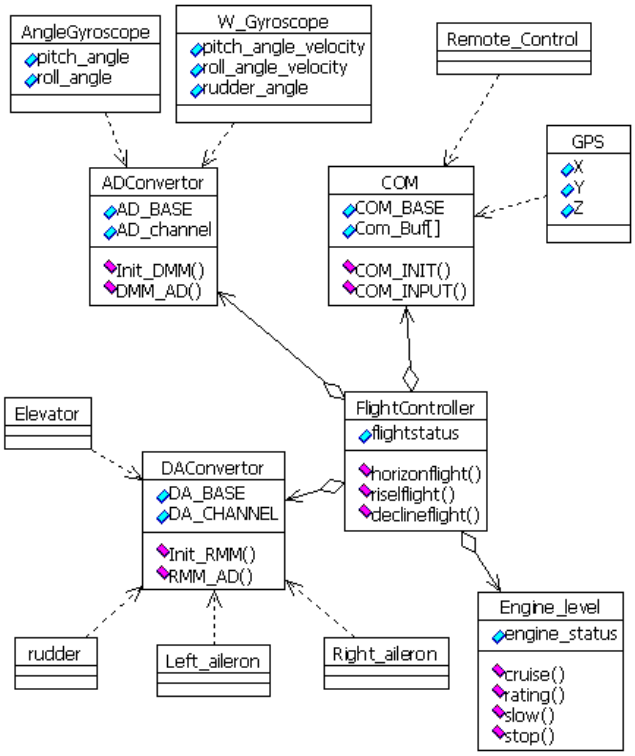

Figure 2. Class diagram of the flight control system

The Class diagram of the flight control system is shown in Figure2.[5-11]. The flight control system is modeled by an aggregation,which is composed of the classes:FlightController, ADconvertor, DAConvertor, COM,Engine. The class ADconvertor is depended by the class anglegynoscope and the class W_ Gynoscope(angle speed gynoscope). The class DAconvertor is depended by the class elevator ,the class rudder, the class left_aileron(left aileron) and the class right_aileron(right

aileron). The class COM(serial port) is depended by the class GPS and the class remote_control/measure(remote_control/measure).

IV. STATECHART_B SPECIFICATION OF FLIGHT CONTROL SOFTWARE

As the core of flight control system, flight controller(FC) is a typical intricate embedded real-time hybrid system .It covers diversified discrete and continuous control algorithm which are used to implement the control and management function including data collection, control law, logic and scheduling judgment, equipment monitoring, autonomic navigation, automatic takeoff and landing, long distance communication, etc..

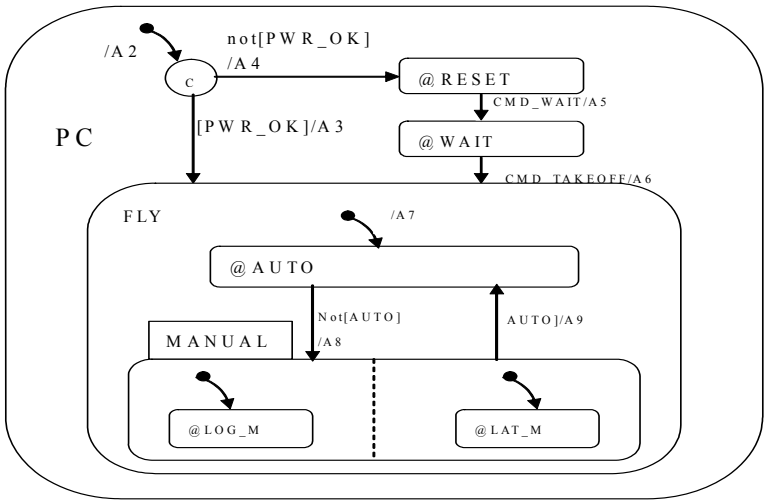

Figure 3. Definition of main control module PC

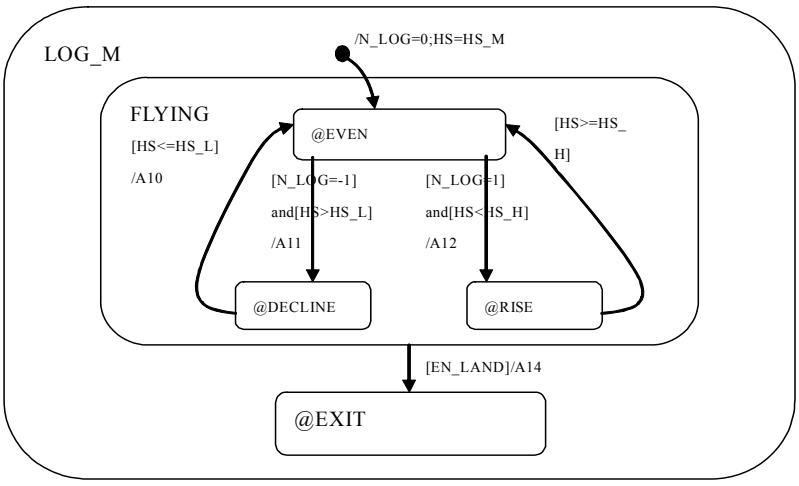

Figure 4. Vertical control of artificial navigation

Main control module PC's detailed implementation is shown in Figure.3.Main control module PC describes the state and state transition of UAV flight control and equipment management including autonomic navigation/remote control navigation, engine control, mission execution etc[8-15]. Statemate can be used to rapidly design and validate complex systems level products through a unique combination of graphic modeling, simulation, code generation, documentation generation, and test plan definition. As a result, Statemate has emerged as the standard for high-end embedded systems development within the medical, automotive, aerospace, and defense industries. Figure.3 is drawn with the tool Statemate.

After the UAV flight control system enters PC , the UAV flight control system will firstly make certain the transition route of different power pattern through condition juncture. If exceptional broken power

comes(condition PWR_OK not true),the UAV flight control system will directly restore flight state RESET ,and execute the reset operation. If UAV flight control system receives the “ wait ” command(CMD_WAIT event),It will transfers to wait state ; in the same way, if UAV flight control system receives the “ takeoff ” command(CMD_TAKEOFF event),the system will transfer the sub-state AUTO of FLY, then complete the running and climbing behavior.

In main control module PC, A2~A9 denote concomitant compound actions respectively; The state LOG_M and the state LAT_M are used to describe the horizontal UAV control function and the vertical UAV control function respectively under artificial navigation. We take LOG_M as example in Figure.4 ,when the system enters horizontal flight state EVEN under default,at this time the variant HS takes normal altitude value;When the vertical command comes(variant N_LOG changes),system will transfer to the declining state DECLINE or the rising state RISE （N_LOG is -1 and 1 respectively）,and the system will give the constant control value through the action A11or A12 to switch to constant altitude integral computing.

While the UAV declines to super low altitude HS_L or UAV rises to limited high altitude HS_H, the UAV flight control system will automatically transfer to the horizontal flight state EVEN,this will assure the safety of the vertical flight of UAV. Furthermore, the transition of state DECLINE and state RISE must pass though state EVEN. the three states ,state DECLINE, state RISE and state EVEN, are described by their own sub-state respectively to implement the smooth conversion of given pitch angle values.

The module FLYING of Figure.4 is used to control vertical artificial navigation. Figure.5 shows the B specification of Flying module. In Figure.5 a abstract machine FLYING is constructed, and three states of EVEN, DECLINE and RISE are defined. Three operations of DO_EVEN, DO_DECLINE and DO_RISE denote the horizontal, declining and rising flight operation respectively. $\delta_z = \delta_{z1}$, $\delta_z = \delta_{z2}$, $\delta_z = \delta_{z3}$, $\delta_z = \delta_{z4}$ and $\delta_z = \delta_{z5}$ denote some given pitch angle values. To implement each operation the relevant conditions including pre-condition and post-condition must be satisfied.

In Figure.5, each operation of the B specification of Flying module satisfies not only both pre-condition and post-condition but also satisfies the branch selection conditions. So the software developed with the B method is reliable, robust and safe.

V. Refinement of Flight Control Software

An important feature provided by the Event B formalism is the ability to stepwise refine specifications. Refinement is a process that transforms an abstract, non-deterministic, specification into a concrete, deterministic, system that preserves the functionality of the original specification. During the refinement process new features that are suggested by the requirements are represented by new variables added to the system. Simultaneously, events are refined to take the new features into account.

```
                    MACHINE FLYING
SETS
          STATUS={ DECLINE,EVEN, RISE}
CONSTANTS
                          HS_M
PROPERTIES
                  HS_M∈ HS_L..HS_H
VARIABLES
                          N_LOG,HS
INVARIANT
          N_LOG={-1,0,1} ∧ HS∈ HS_L..HS_H
INITIALIZATIONS
                  N_LOG=0  HS= HS_M
OPERATIONS
      7.1  DO_EVEN=
            PRE
                s∈  STATUS ∧  HS∈ HS_L..HS_H
            THEN
                Select  N_LOG=0 ∧ HS=HS_M  then
                    N_LOG=0 | |
                    δz = δz1
                when  N_LOG=-1 ∧ HS！=HS_L  then
                    N_LOG=0 | |
                    δz = δz2
                when  N_LOG=1 ∧ HS=HS_H  then
                    N_LOG=0 | |
                    δz = δz3
             Else skip end
          End;
     7.2DO_DECLINE=
          PRE
                s∈  STATUS ∧  HS∈ HS_L..HS_H
          THEN
                Select  N_LOG=0 ∧  HS!=HS_L  then
                    N_LOG=-1 | |
                    δz = δz4
          Else skip end
        End;
     7.3DO_RISE=
          PRE
                s∈  STATUS ∧  HS∈ HS_L..HS_H
          THEN
                Select  N_LOG=0 ∧  HS!=HS_ H then
                    N_LOG=1 | |
                    δz = δz5
           Else skip end
        End;
```

Figure 5. B specification of Flying modul

In Figure.6, vertical control of artificial navigation refined by adding new states and transitions is shown. We add the state SUSP which denotes suspending state and the state Failed which denotes the system is failed.

When refining a system the behaviour of the system in a more detailed manner is modeled. Before adding more detailed behaviour we need to reveal more detailed state space. This can be done using hierarchical states to

introduce sub-states within a state. In Figure.7, the refinement of failure management of vertical control of artificial navigation is shown, which is the second refinement of Figure.6. The state SUSP is split into sub-states such as state anglegynoscope_failed, state W_Gynoscope_failed, state rudder_failed, state left_aileron_failed, state right_aileron_failed, state state GPS_failed, state unkown, state kownn, state fixed. At the same time the old events is refined and the new events are added betwe en the substates.

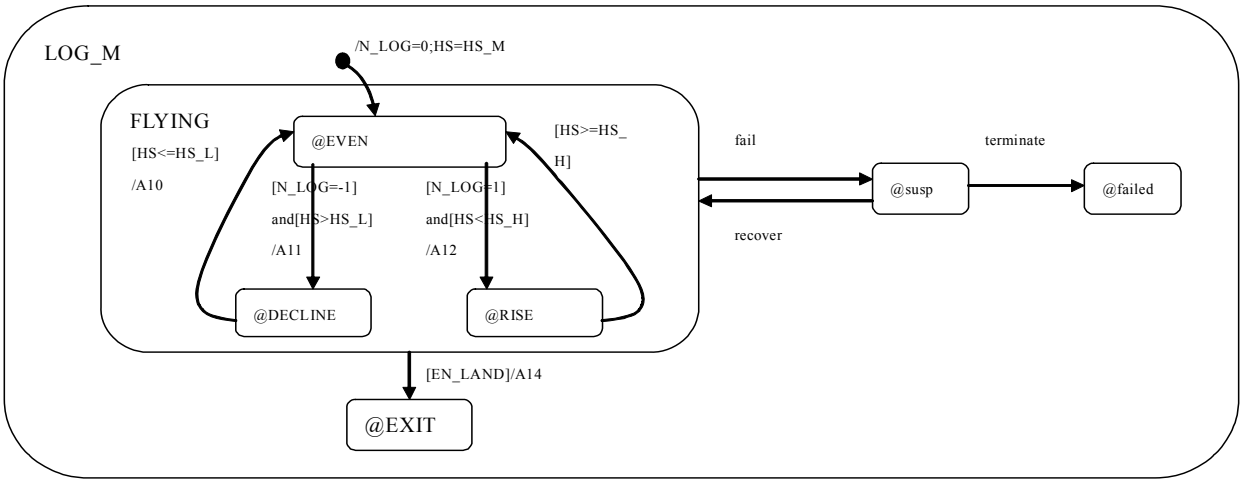

Figure 6.Vertical control of artificial navigation refined by adding new states and transitions

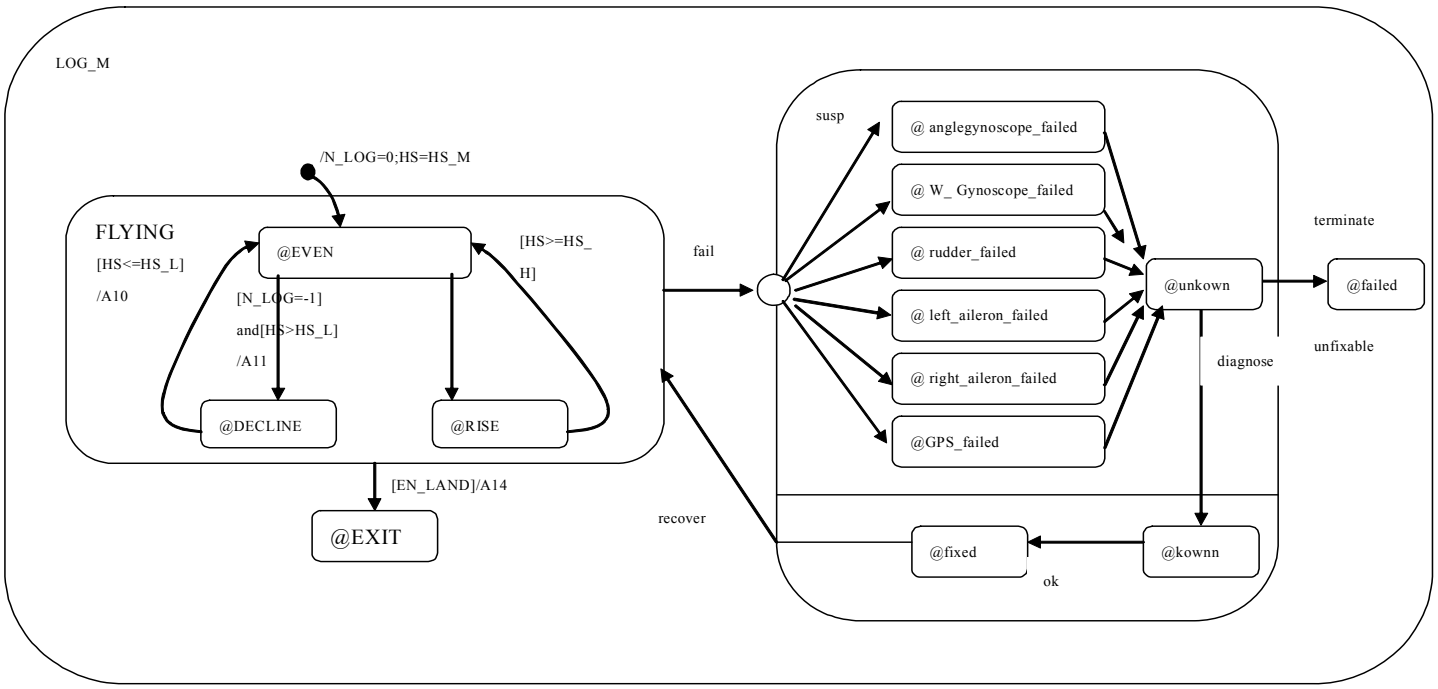

Figure 7. The refinement of failure management of vertical control of artificial navigation

VI. PROVEMENT OBLIGATIONS

At every stage of the specification, proof obligations ensure that operations preserve the system invariant. A set of proof obligations that is sufficient for correctness must be discharged when a refinement is postulated between two B components. Hence, by supporting proved refinement, B allows to go progressively from an abstract specification (non deterministic) to a deterministic specification that can be translated into a programming language (ADA, C or C++).

Analyzing the proof obligations is an efficient and practical way to detect errors encountered during the specification development.

Vertical control of artificial navigation have proof obligations to ensure safety.

When the command is the declining command, N_LOG=-1, the action $\delta_z = \delta_{z4}$ occurs, then the UAV declines, but during the declining period, the pre-condition and post- condition must both be satisfied and fulfilled. The needed proof obligations are as follows during the declining period.

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ N_LOG=0 ∧ HS!=HS_L ⇒ N_LOG=-1∈ STATUS

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ HS!=HS_L ∧ N_LOG=-1 ∧ $\delta_z = \delta_{z4}$ ⇒ HS∈ HS_L..HS_H

When the command is the rising command, N_LOG=1, the action $\delta_z = \delta_{z5}$ occurs, then the UAV rises, but during the rising period, the pre-condition and post-condition must both be satisfied and fulfilled. The needed proof obligations are as follows during the rising period.

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ N_LOG=0 ∧ HS!=HS_M ⇒ N_LOG=1∈ STATUS

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ HS!=HS_M ∧ N_LOG=1 ∧ $\delta_z = \delta_{z5}$ ⇒ HS∈ HS_L..HS_H

When the command is the even command, N_LOG=0, the even action occurs, then the UAV is in horizontal flight, but during the even period, the pre-condition and post-condition must both be satisfied and fulfilled.

There are three cases.

First, the UAV state is from the even state to the even state. The needed proof obligations are as follows during the even period.

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ N_LOG=0⇒ N_LOG=0∈ STATUS

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ N_LOG=0 ∧ $\delta_z = \delta_{z1}$ ⇒ HS∈ HS_L..HS_H

Second, the UAV state is from the declining state to the even state. The needed proof obligations are as follows during the even period.

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ N_LOG=-1 ∧ HS!=HS_L ⇒ N_LOG=0∈ STATUS

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ HS!=HS_L ∧ N_LOG=0 ∧ $\delta_z = \delta_{z2}$ ⇒ HS∈ HS_L..HS_H

Third, the UAV state is from the rising state to the even state. The needed proof obligations are as follows during the even period.

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ N_LOG=1 ∧ HS!=HS_M ⇒ N_LOG=0∈ STATUS

AUTO ∧ s ∈ STATUS ∧ HS ∈ HS_L..HS_H ∧ HS!=HS_M ∧ N_LOG=0 ∧ $\delta_z = \delta_{z3}$ ⇒ HS∈ HS_L..HS_H

VII. CONCLUSION

In this paper, a new approach for modeling UML diagrams in B is presented. The Class diagram of the flight control system is presented and each class operation is mapped to a B abstract machine, and the B specification is derived from UML statechart diagrams. By adopting the tool of Statemate, which support the virtual prototype technology, the detailed description of flight control software behaviors is presented based on statecharts. The B method is used to translate the statecharts into B specification of one flight control software. Using UML and B method, flight control is refined and failure management is added. The architecture of B specification derived from class and statecharts diagram is built. The UML specifications can be formally verified by analyzing the corresponding B specification. The integration of the UML specifications and B method overcomes the drawback of the UML.

REFERENCES

[1] http://www.ilogix.com/pdf/StatemateBrochure(V4).pd[M], 2003

[2] Chunyu Miao. Dynamic Slicing Research of UML Statechart Specifications. Journal of Computers, 2011,6, (4):792-798

[3] J. Abrial.The B Book- Assigning Programs to Meanings[M]. Cambridge University Press, 1996

[4] Manoranjan Satpathy.ProTest:An Automatic Test Environment for B Specifications[J].Electronic Notes in Theoretical Computer Science ,2005,111:113-136

[5] Akram Idani, Yves Ledru. Dynamic graphical UML views from formal B specifications[J]. Information and Software Technology, 2005,33: 1-16

[6] Amel Mammar.From a B Formal specification to an executable code : application to the relational database domain[J]. Information and Software Technology, 2006,48: 253-279

[7] H. Ledang. Integrating UML and B Specification Techniques.In the Informatik2001 Workshop on Integrating Diagrammatic and Formal Specification Techniques, Vienna (Austria), September 26, 2001.

[8] H. Ledang. Modeling class operations in B : a case study on the pump component. Technical Report A01-R-011, Laboratoire Lorrain de Recherche en Informatique et ses Applications, March 2001.

[9] H. Ledang. Modeling Class Operations in B: Application to UML Behaviral Diagrams. In ASE2001: the 16th IEEE International Conference on Automated Software Engineering, Loews Coronado Bay, San Diego (USA), November 26-29, 2001.

[10] H. Ledang. New Approach for Modeling State-Chart Diagrams in B.http://www.loria.fr/ledang/publications/state-chart-modeling.ps.gz, July 2001.

[11] Lina Chen. Automatic Test Cases Generation for Statechart Specifications from Semantics to Algorithm. Journal of Computers, 2011,6 (4):769-775

[12] Colin Snook, Marina Waldén. Refinement of Statemachines Using Event B Semantics. B 2007, LNCS 4355, 2006:171–185

[13] Jiufu LIU, Zhong YANG. UML and B method based analysis and refinement for flight control software of unmanned aerial vehicle. 2008 International Conference on

Computer Science and Software Engineering,2008, 1135 - 1141
[14] Jiufu LIU. Integration of statechart and B method based analysis and verification for flight control software of unmanned aerial vehicle. ACM SIGSOFT Software Engineering Notes, 2007, 32(2):1-4
[15] Xiao Liang, et al. Dynamic Modeling and Computer Simulation for Autonomous Underwater Vehicles with Fins. Journal of Computers, 2013 ,8 (4):1058-1064

Jiufu Liu is a researcher with Institute of Automation Engineering,Nanjing University of Areonautics and Astronautics,,P.R.China. He received his Ph.D. from Nanjing University of Areonautics and stronautics,,P.R.China.His research interests include software theory and formal methods,artificial intelligence.

Jianyong Zhou is a graduate student with Institute of Automation Engineering,Nanjing University of Areonautics and Astronautics,,P.R.China. His research interests include software testing and artificial intelligence.

Chunsheng Liu is a professor with Institute of Automation Engineering,Nanjing University of Areonautics and Astronautics,,P.R.China.His research interests include artificial intelligence,control theory.

Zhong Yang is a professor with Institute of Automation Engineering,Nanjing University of Areonautics and Astronautics,,P.R.China.His research interests include embedded and real-time software,control theory.

Zhisheng Wang is a professor with Institute of Automation Engineering,Nanjing University of Areonautics and Astronautics,P.R.China.His research interests include embedded and real-time software, artificial intelligence, human-computer interaction.

Yifan Zhang is a researcher with Institute of Information Engineering, Henan University of Animal Husbandry Economy,P.R.China. His research interests include software engineering, internet and information systems.