Design and Experimental Validation of UAV Control System Software Based on the TMO Structuring Scheme

Hansol Park[1], Moon Hae Kim[1], Chun-Hyon Chang[1], Keechon Kim[1], Jung-Guk Kim[2], and Doo-Hyun Kim[1,*]

[1] Computer Science Department, Konkuk University, Seoul, Korea
{parkhs, mhkim, chchang, kckim, doohyun}@konkuk.ac.kr
[2] Hankuk University of Foreign studies, Korea
jgkim@hufs.ac.kr

Abstract. The technologies for designing and validating computer-control systems subject to challenging timing and reliability requirements have been advancing slowly. One such type of systems are unmanned aerial vehicle (UAV) control systems. The functional complexity of UAV control systems is steadily increasing. Enabling the design of such complex systems in easily understandable forms that are amenable to rigorous analysis is a highly desirable goal. In this paper, we discuss our experimental application of the Time-triggered Message-triggered Object (TMO) structuring scheme to the design of a UAV control system. The TMO scheme enables high-level structuring together with design-time guaranteeing of accurate timings of various critical control actions with significantly smaller efforts than those required when using lower-level structuring schemes based on direct programming of threads, UDP invocations, etc. An experimental 2-step validation of a UAV control system is also discussed. The first step was to validate the system by use of an environment simulator and then real flight tests were involved only in the second step.

1 Introduction

The technologies for designing and validating computer-control systems subject to challenging timing and reliability requirements have been advancing slowly. One such type of systems are unmanned aerial vehicle (UAV) control systems. Enabling the design of such systems in easily understandable forms that are amenable to rigorous analysis is a highly desirable goal. [5, 6, 8, 10]

The TMO (Time-triggered Message-triggered Object) model formalized earlier by Kim and his collaborators [1, 2] has been found to be a sound real-time object model that can be used for various types of hard and soft real-time distributed computing applications. With the TMO model, both functional and timing behaviors of a system can be specified in explicit and natural easy-to-understand forms.

To support execution of TMOs, several engines have been developed in the form of middleware layered on a few widely used OS platforms. Representative cases are TMOSM [3, also see http://dream.eng.uci.edu] on MS Windows XP Windows CE,

* Corresponding Author: New Millennium Hall 1203, School of Internet and Multimedia Engineering, Konkuk University, Kwangjin-Gu, Seoul, 143-701, Korea.

R. Obermaisser et al. (Eds.): SEUS 2007, LNCS 4761, pp. 192–201, 2007.

and Linux, LTMOS [4] on Linux, and Konkuk TMOSM/Linux on Linux. In the work reported in this paper, we used TMOSM/Linux made by Konkuk University.

In this paper, we propose a TMO-based high-level design and implementation method for the real-time embedded software parts of UAV control systems. The main goal here is to improve the software engineering productivity and the software reliability to a significant extent. Improvements are sought for in various phases of engineering UAV control software such as design, implementation, and testing.

Our UAV control system was validated by use of an environment simulator in the first step. Real flight tests were involved only in the second step. FlightGear was used a virtual flight environment, named Hardware-In-the-Loop (HIL) system, of which components can be replaced by actual hardware without rendering the remainder of the simulator inoperable [7].

In Section 2, as backgrounds, the basic structure of the TMO model, the type of UAVs considered, and the features of FlightGear are described briefly. In Section 3, we present the design of an UAV control system based on the TMO model. In Section 4, our implementation and flight simulation are described. Finally, in Section 5, we conclude with a suggestion for future works.

2 Backgrounds

2.1 TMO Structuring Scheme

We use the TMO model as a fundamental building-block and TMOSM (TMO Support Middleware) as the execution engine for our experiments [2, 3]. TMO is a natural, syntactically minor, and semantically powerful extension of the conventional object(s) [2]. Especially, TMO is a high-level real-time computing object. Member functions (i.e., methods) are executed within specified windows in the domain of global time. Such timing requirements are specified in natural intuitive forms with no esoteric styles imposed.

As depicted in Fig. 1, the basic TMO structure consists of four parts; 1) **Spontaneous Method (SpM)**: A time-triggered (TT) method which is triggered when the real-time clock reaches specific values determined at design time and specified in AAC (Autonomous Activation Condition) as the time-windows for execution. 2) **Method (SvM)**: A method similar to the conventional service method which is triggered by service request messages from clients. 3) **Object Data Store (ODS)**: The set of data members which may be partitioned into ODS segments (ODSSs), each of which is a basic unit of storage that can be exclusively accessed by a certain TMO method at any given time or shared among executions concurrent of TMO methods (SpMs or SvMs). 4) **Environment Access Capability (EAC)**: A list of entry points to remote object methods, logical communication channels, and I/O device interfaces.

2.2 Unmanned Aerial Vehicle (UAV) Control System

An unmanned aerial vehicle (UAV) is an aircraft with no onboard pilot. UAVs can be remote controlled aircrafts (e.g. flown by a pilot at a ground control station), or aircrafts that can fly autonomously with pre-programmed flight plans or dynamically adaptive control systems. Although the UAV control system consists of many components, there are two main components: 1) a ground station component which

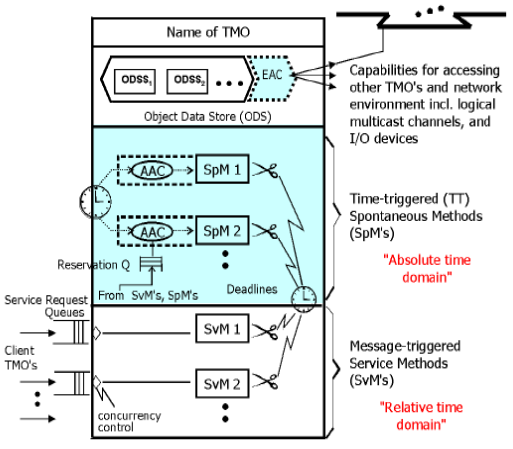


Fig. 1. Structure of TMO model (courtesy of [2, 3])

provides telemetry feedback to the operator and allows him/her to control the aircraft, and 2) an on-board flight system component of the vehicle.

In a UAV, the flight system component (embedded controller) operates some sensors for sensing attitude and position and uses the data in controlling the pose of the UAV on the flight. In our experimental UAV, a gyroscope and a GPS were used to enable precise navigation. In the ground, the operator can monitor the current states of the UAV and issue commands to the running UAV via the ground station component. The ground station component and the UAV flight system component must be connected via communication channels so that they may monitor and command each other.

Our UAV discussed in this paper is designed to fly autonomously with the autopilot. This autopilot requires precise timing of sensor operations. If the timing is badly missed, the UAV would be endangered.

2.3 FlightGear Flight Simulator Project

FlightGear is an open-source flight simulator project which provides flight simulators running on Windows, Linux and Mac platforms. The goal of the FlightGear project is to create a sophisticated flight simulator framework for use in research or academic environments, for the development and pursuit of other interesting flight simulation ideas, as well as for an end-user application. FlightGear provides a multitude of features as follows:

- High Degree of Freedom: FlightGear is open-source.
- Cross Platform: FlightGear runs on many different operating systems.
- Multiple Flight Dynamic Models: Three primary flight dynamics models (FDMs), LarcSim, JSBSim and yasim, are available.
- Moderate Hardware Requirements: Commercial-off-the-shelf (COTS) personal computer components are sufficient for running FlightGear.
- Extensibility: FlightGear can run a simple simulation on a single laptop or drive a sophisticated, realistic, immersive, multi-screen simulation.
- Network Access: A wide variety of external interfaces are available.

3 Experimental System Design and Implementation

3.1 Hardware Architecture

As depicted in Figure 2, the UAV control system consists of a Flight System component, a Ground Station component, and a Communication component. The Flight

System component has three parts, embedded controller, sensors, and actuators. In this project, an RF-Ethernet communication device which used radio frequency for Ethernet protocol was used as the communication component. Flight simulator was set to generate all kinds of sensor data and receive actuator signals for virtual flight control.

3.1.1 Flight System

(1) Sensors

A gyroscope and a GPS receiver were used as the sensors of Flight System. The gyroscope was used to provide attitude data. The gyroscope was set to generate roll, pitch, and yaw data and sends the data to the Embedded Controller, which was connected to the gyroscope via a serial port. The GPS receiver was used to produce geometric location data periodically. The GPS provided NMEA standard location data via a serial port for the Embedded Controller in the Flight System.

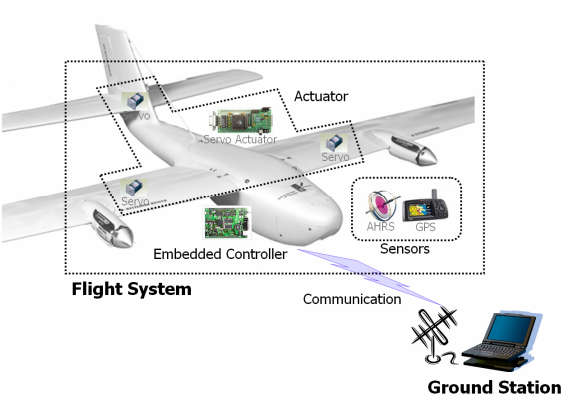


Fig. 2. Constitution of UAV Control System

(2) Actuator

A Servo-Actuator that was an actuator of servo motors controlled by PWM (Pulse Width Modulation) signals in the Flight System was assembled by us. The Servo-Actuator was designed to receive servo-control data from the Embedded Controller via a serial port and convert the data to PWM signals and use the signals to control directly-connected servo motors. Atmel's Atmega128 chip was used for the Servo-Actuator and the converting software was newly developed.

(3) Embedded Controller

The Embedded Controller was built using an ARM-based small computer system. This small computer system has a 400 MHz XScale processor, and 64MB of RAM. An extended Serial I/O board is installed in the Embedded Controller to provide four serial ports needed to communicate with the gyroscope and GPS devices. A 64MB built-in NAND flash serves as non-volatile storage and suffices to keep embedded Linux kernel 2.4.18 and UAV controller software based on TMOSM/Linux.

3.1.2 Ground Station

Ground Station is a kind of control unit which monitors the state of the UAV and provides a flight-path to the Embedded Controller in the Flight System. Ground Station supports GUI for monitoring and sends location data which include a flight-path to the UAV. Ground Station uses the RF-Ethernet device for communicating with the UAV.

3.2 Modeling of a UAV Control System Based on TMO Model

As depicted in Fig. 2, the Embedded Controller in the Flight System is connected to Sensor Module, Servo Actuator, and Ground Station. In this section, we present the TMO-structured design of our UAV control system.

3.2.1 UAV TMO : A TMO-Structured Design of the Embedded Controller

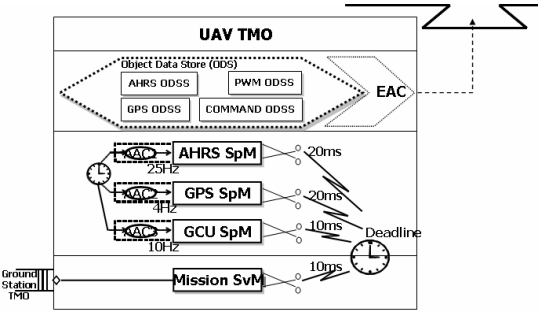


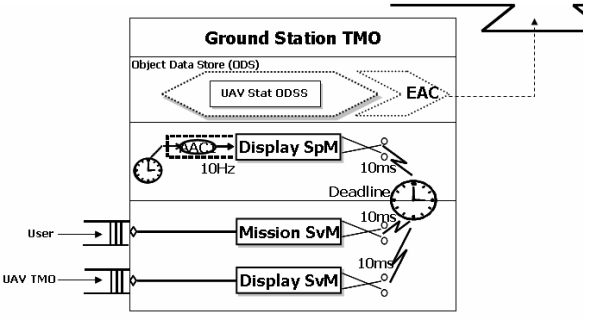
Fig. 3. Design of the UAV TMO

The UAV TMO is a design of the Embedded Controller object in the Flight System. As depicted Figure 3, the UAV TMO has AHRS(Attitude and Heading Reference Systems) SpM and GPS SpM because AHRS and GPS produce sensor data at different rates. The functional requirements for each SpM and SvM in the UAV TMO are defined in Table 1.

Table 1. Functional requirements of UAV TMO

Method Name	Functions	Deadline
AHRS SpM	1. Acquire AHRS packets from gyroscope every 40ms 2. Analyze and parse the AHRS packet and write parsed attitude data to AHRS ODSS 3. Read AHRS data, GPS data, and command data from each ODSS. 4. If GPS data and Command data are empty (initial condition), just use AHRS data. 5. If the current GPS data are not found, produce such data via extrapolation (because GPS internal clock is slower than AHRS). 6. Find flight-path from command data and compare it with the current GPS data. 7. Calculate actuator control signals using PID algorithm to follow given flight-path. 8. Write AHRS, GPS and actuator control data to PWM(Pulse Width Modulation) ODSS.	20ms
GPS SpM	1. Acquire GPS packets every 250ms 2. Analyze and parse the GPs packet and write parsed geometric position data to GPS ODSS	20ms
GCU SpM	1. Read the status data from PWM ODSS every 100ms 2. Send the status data to the Ground Station TMO using a Gate in EAC	10ms
Mission SvM	1. Receive command data from Ground Station TMO 2. Write the command data to COMMAND ODSS	10ms

3.2.2 Ground Station TMO



Ground Station TMO includes Display SpM and Display SvM for monitoring and Mission SvM for mission planning as depicted in Fig. 4. The functional requirements for Ground Station TMO are defined in Table 2.

Fig. 4. Design of the Ground Station TMO

Table 2. Functional requirements of Ground Station TMO

Method Name	Functions	Deadline
Display SpM	1. Read all data of the UAV from the UAV State ODSS and Command ODSS every 40ms 2. Send all data to the user application for display	10ms
Mission SvM	1. Read command data which include waypoints of the UAV from the call parameters supplied by the user application 2. Send the data to the UAV TMO in Flight System	10ms
Display SvM	1. Receive the status data of the UAV from Flight System 2. Write the data to UAV State ODSS	10ms

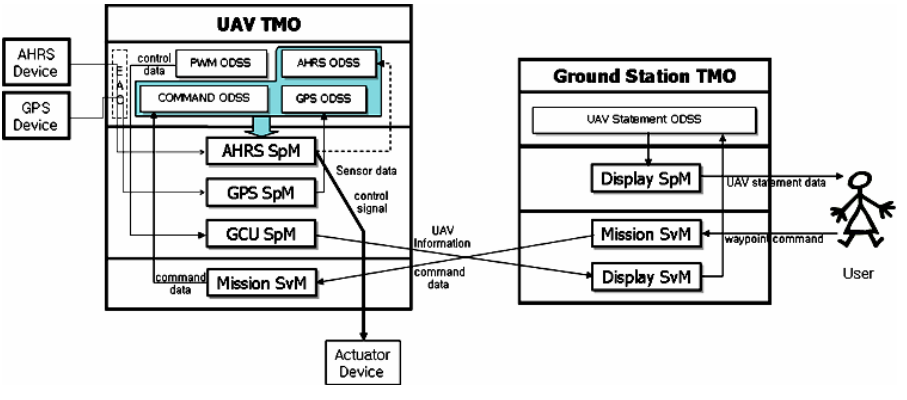


Fig. 5. Interaction of the UAV TMO with the Ground Station TMO

Fig. 5 depicts interactions including data flows between UAV TMO and Ground Station TMO. GCU SpM in UAV TMO requests a service to Display SvM in Ground Station TMO for displaying UAV status data, while Mission SvM in Ground Station TMO requests a service to Mission SvM in UAV TMO for accepting command data.

3.3 Implementation

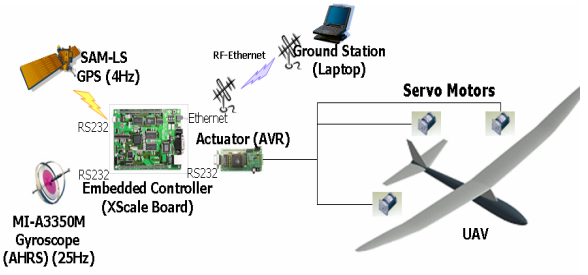


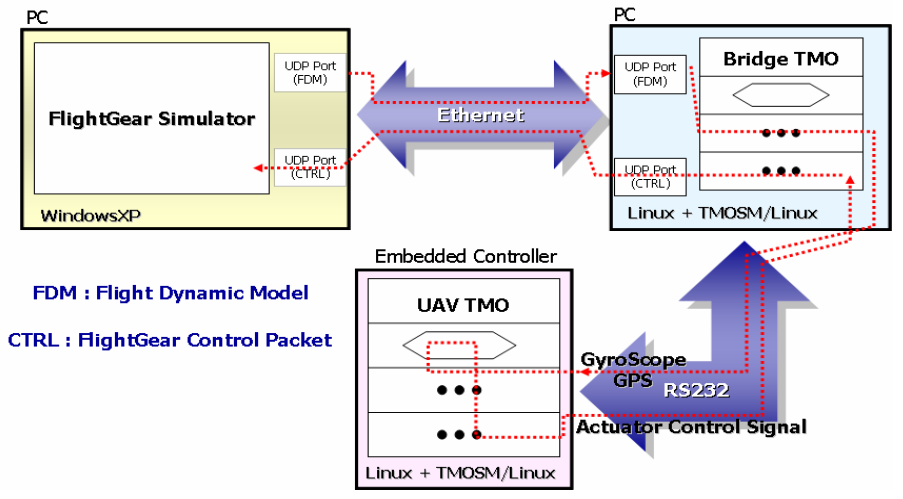
Fig. 6. UAV control system architecture

For our experimental implementation of a UAV control system, 25Hz gyroscope, 4Hz GPS, XScale-based Embedded Controller, and AVR-based Actuator were used to compose the Flight System. PID algorithm was used in our autonomous UAV control system. We used a general-purpose laptop computer as the Ground Station for monitoring the status of the UAV. Fig. 6 presents the hardware structure of the implemented UAV control system. The TMO-structured real-time computing software in both Flight System and Ground Station turned out to be remarkably easier to read, analyze, and maintain in comparison to the initial version of the software that had been designed 4 years ago and composed of threads, sockets, and thread-priorities.

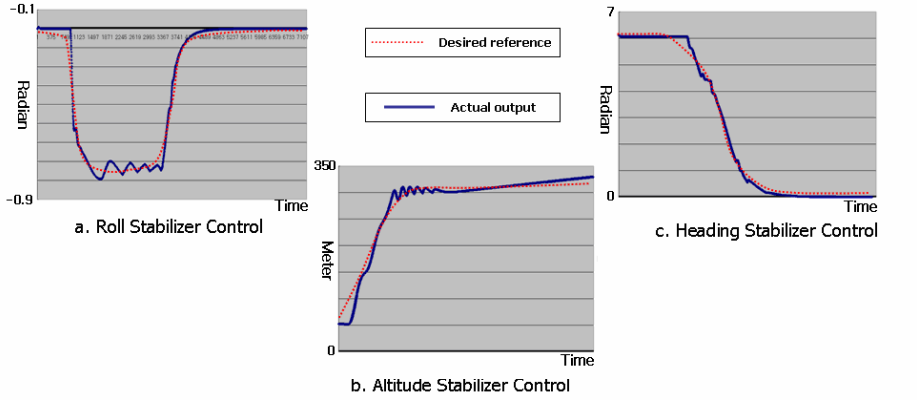
4 Validation

4.1 Step 1: Test with an Environment Simulator

In our experiments, we deployed a hardware-in-the-loop simulator (HILS) centered around FlightGear and depicted in Fig. 7(a). In HILS, a Bridge TMO converts a state data packet coming out of FlightGear to a packet of Sensors (Gyroscope, GPS) in the format that can be accepted by UAV TMO without any further conversion. Moreover, it converts a control signal coming out of UAV TMO to a control packet that can be accepted by FlightGear. We tested roll, altitude, and heading stabilizer control of our UAV control system on the HILS. Fig. 7(b) shows the results. The desired control references are displayed as dotted lines and the actual responses of the embedded controller in the UAV are displayed as bold lines. These response data were obtained while the UAV in simulation changed the altitude and heading into the direction toward the destination. Fig. 7(b) has three different graphs which show changes in the stabilizer of roll, altitude, and heading. When the UAV has to change heading direction toward the destination (See Fig. 7(b)-c), roll and pitch (altitude stabilizer) values will also be changed until heading is set in the direction toward the destination (See Fig. 7(b)-a, b). During this period, the UAV encounters an unstable situation, and the embedded controller should adjust the attitude of the UAV for stable flight. The results of each stabilizer control indicate that the actual response data obtained are close to the desired references for stable flight. By comparing the figures, we can conclude that the exhibited behavior of the simulated Flight System is close to the desired references. These results attest to the accuracy of our embedded control system implemented with the TMO structuring techniques and tools and the usefulness of the hardware-in-the-loop simulator.



(a)



(b)

Fig. 7. Hardware-in-the-loop simulator and Flight Simulation output

4.2 Step 2: Real Flight Test

Fig. 8. UAV flight test in Korea on 18 June 2006

On 18 June 2006, we tested our UAV in a field in Korea as shown in the pictures in Fig. 9. Our Flight System communicated with the Ground Station via RF-Ethernet. The Flight System could control the UAV using the roll, altitude, and heading stabilizer for approximately 45min. However, unpredictable winds occasionally created dangerous situations in which the Flight System lost the stability for short periods.

5 Conclusion and Future Work

In this paper, we presented our experimental application of the TMO structuring scheme to the design and implementation of a UAV control system. The Flight System and the Ground Station were designed by application of the TMO scheme and implemented with TMOSM/Linux. We tested our UAV control system by means of timing behavior analysis, flight simulation, and real flight tests. We could confirm a reasonable-level performance of roll, altitude, and heading stabilizer control from flight simulations. Finally, we could confirm a reasonable-level performance during the test flight of 45min in a real flight environment. Therefore, these experimental research results clearly indicate the promising nature of the TMO structuring scheme in design and implementation of challenging real-time distributed computing software such as those needed in UAV control systems. The advantages of the TMO scheme over conventional low-level programming approaches involving composition of software with threads, sockets, and thread-priorities seem quite significant. Among several areas in which the UAV control system could be improved, fault-tolerance in UAV control systems is an item of top-priority to us for tackling in the near future research.

Acknowledgements. This research was conducted at the Software Research Center of Konkuk University headed by Moon-Hae Kim and supported by the MIC (Ministry of Information and Communication), Korea, under the University ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

References

1. Kim, K.H.: Object Structures for Real-Time Systems and Simulators, pp. 62–70. IEEE Computer Society Press, Los Alamitos (1997)
2. Kim, K.H.: APIs for Real-Time Distributed Object Programming, pp. 72–80. IEEE Computer Society Press, Los Alamitos (2000)
3. Kim, K.H., Ishida, M., Liu, J.: An Efficient Middleware Architecture Supporting Time-Triggered Message-Triggered Objects and an NT-based Implementation. In: ISORC, pp. 54–63 (1999)
4. Kim, H.J., Park, S.H., Kim, J.G., Kim, M.H.: TMO-Linux: A Linux-based Real-time Operating System Supporting Execution of TMOs. In: ISORC (2002)
5. Koo, T.J., Liebman, J., Ma, C., Sastry, S.: Hierarchical approach for design of multi-vehicle multi-modal embedded software. In: Henzinger, T.A., Kirsch, C.M. (eds.) EMSOFT 2001. LNCS, vol. 2211, Springer, Heidelberg (2001)

6. Koo, T.J., Liebman, J., Ma, C., Horowitz, B., Sangiovanni-Vincentelli, A., Sastry, S.: Platform-based embedded software design and system integration for autonomous vehicles. Proceedings of the IEEE 91(1), 198–211 (2003)
7. Sorton, E.F., Hammaker, S.: Simulated flight Testing of an Autonomous Unmanned Aerial Vehicle Using FlightGear. Institute for Scientific Research Inc., Fairmont, WV AIAA-2005-7083
8. Ippolito, C.: QSS Group, Inc., NASA Ames Research Center, An Autonomous Autopilot Control System Design for Small-Scale UAVs. Internal Report of CMIL in University of Carnegie Mellon: EAV-20051016
9. Lee, E.A.: Embedded Software – An Agenda for Research. UCB ERL Memorandum M99/63 in University of California at Berkeley
10. Matczynski, M.J.: A Distributed Embedded Software Architecture for Multiple Unmanned Aerial Vehicles. Master thesis, EECS, MI