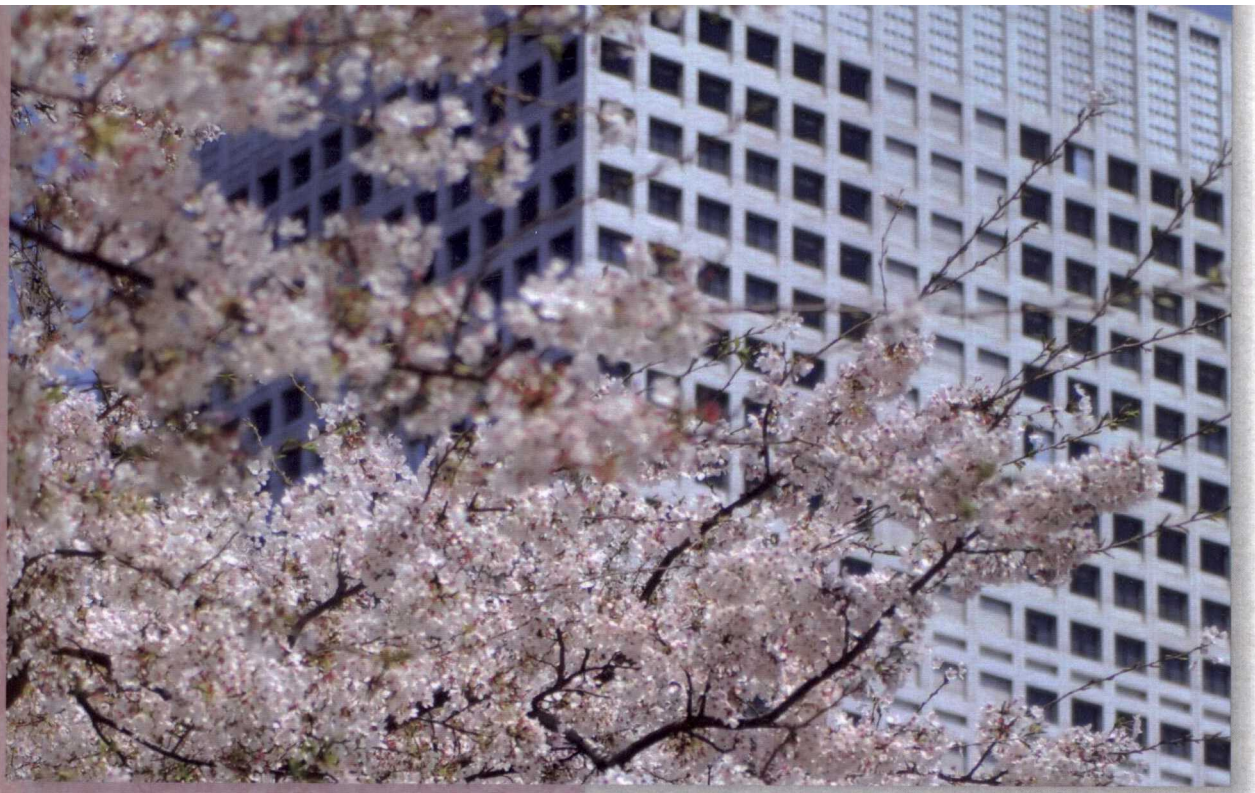

IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing

IEEE computer society

IEEE

DEOSC

EMBLIX

Proceedings of the

12th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing

17-20 March 2009 / Tokyo, Japan

CPS Conference Publishing Services

Los Alamitos, California

Washington • Tokyo

IEEE computer society

Session 3: Panel (I): Model Driven Design and Organic Computing - Contradictory or Synergetic Approaches to Overcome the Embedded Software Crisis

Moderator: Uwe Brinkschulte
Panelists: Dietmar Fey, Mike Hinchey, Tomoji Kishi, Fabrice Kordon, Peter Puschner

Session 4: National-Level Movements in Software Technology R&D (II)

Session 5A: Scheduling and Resource Management

2009 IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing

HELISCOPE Project:
Research Goal and Survey on Related Technologies

Doo-Hyun Kim, Kodirov Nodir
and Chun-Hyon Chang
School of Internet & Multimedia Engineering
Konkuk University
Seoul, Korea
e-mail: doohyun@konkuk.ac.kr

Jung-Guk Kim
College of Information and Industrial Engineering
Hankuk University of Foreign Studies
Korea
e-mail: jgkim@hufs.ac.kr

***Abstract*—In this paper, we introduce the target scenario of our HELISCOPE project with the emphasis on the four modes for *flying-camera* task. We also explain the control points that are considered as the interfaces where the on-flight embedded software and mechanical implementation of the unmanned helicopter are contacted for the autopilot necessarily used in performing such as various modes of *flying-camera* tasks. Being related to real-time issues, we are developing a real-time S/W platform based on TMO (time-triggered and message-triggered object) scheme. We, in this paper, explain the clues in becoming to use the TMO mode with the survey of other on-flight real-time S/W platform researches.**

Keywords: disaster response, unmanned helicopter, on-flight software, real-time embedded software platform

I. INTRODUCTION

Recently, according the progress of global warming, the magnitude and frequency of natural disasters tend to be increased. Similarly, industrial disasters are also getting huger due to the progress of industrialization. The disaster prevention has already become a nation-wide issue and prepared by the government through the disaster management which consists of four phases: mitigation, preparedness, and response and recovery phases. The HELISCOPE project is to develop on-flight computing system, embedded S/W and related services for unmanned helicopter that shall be used especially in the phase of disaster response and recovery intensively.

In this paper, we introduce the target scenario of our HELISCOPE project with the emphasis on the four modes for flying-camera task. We also explain the control points that are considered as the interfaces where the on-flight embedded software and mechanical implementation of the unmanned helicopter are contacted for the autopilot necessarily used in performing such as flying-camera tasks[3-6].

In addition, the on-flight embedded S/W intrinsically requires real-time features for the computations related to getting current status of attitude, heading and global position, computing the commands for the next movement, communicating with GCS(Ground Control System), and feeding the commands out to the servo actuators. These series of computations should be guaranteed to be finished with every specific period suitable for sustaining safe and stable flight. Being related to the real-time issue, we have started to develop a real-time S/W platform based on TMO (time-triggered and message-triggered object) scheme. We, in this paper, also explain the clues in becoming to use the TMO mode with the survey of other on-flight real-time S/W platform researches.

II. TARGET SCENARIO AND FLYING-CAMERA TASKS

A. Target Scenario

Although the unmanned helicopter, HELISCOPE, can be uses in diverse aspect of the disaster management phases, we have special emphasis on the usage of HELISCOPE in the phases of disaster response because it requires immediate and interactive use on-the-spot. The HELISCOPE is to be included in the mobilization of the necessary emergency services and service equipments or can be used as one of first responders. Immediately upon its arriving at the spot of disaster, the HELISCOPE takes off and shoot the still pictures and moving images to send GCS through mobile internet such Wibro(Wireless Broadband) [1] and HSPA(High Speed Packet Access) [2]. In addition to these first stage tasks, the HELISCOPE performs various tasks related to taking detailed video shooting by using the capabilities of unmanned helicopter such as hovering, point navigation, and waypoint navigation.

B. Research Challenges

The HELISCOPE project ultimately aims to develop flying-camera services that provide aerial video streams without any limitations of distance and geographical difficulties as well as to provide GUI-based easy-to-use GCS by integrating GIS, mobile internet and user-centered interaction devices. As shown in TABLE I, the intelligent aspects such as controlling beyond the line of sight over reliable mobile internet, self-establishment of flying-prohibited area by sensing thermal level, power dissipation

1555-0885/09 $25.00 © 2009 IEEE
DOI 10.1109/ISORC.2009.49

and secured area, etc are also considered as functionalities of surprising level.

TABLE I. LEVELS OF CHALLENGE

Level	Items
Basic Level	- Safe Take-off and Landing - Waypoints Navigation and Hovering - Auto-repatriation - Manual-control by RC
Primary Level	- Integration of GIS and Waypoints - Video-streaming over mobile internet, i.e., Wibro and HSPA - Joystick remote control of Flying-Camera
Surprising Level	- Controlling beyond Line of Sight over reliable mobile internet - Self-establishment of Flying-Prohibited Area(Sensing thermal level, power dissipation and secured area, etc) - Integrated Visualization of USN, GIS, UAV

C. *Four Modes of Flying-Camera Tasks*

The on-flight equipment including camera, i.e., *camera block,* is responsible to accomplish such flying-camera tasks as taking overall/exact view of remote or hazardous location or shots of target object. In order to accomplish those tasks, the camera block can interact with FCC(Flight Control Computer) to compute certain commands directed to servo actuator of camera such as gimbals. This control can be divided to four modes:

- Mode 1: control by Joystick. As shown in Figure 1, camera can be controlled manually via joystick, where in this mode camera is able to track target object, which is located in camera's view area.

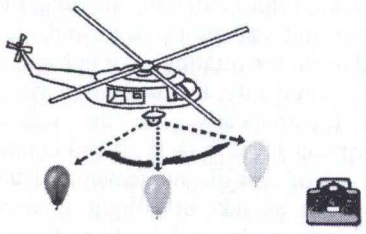

Figure 1. Joystick-Search mode

- Mode 2: Object can be steady also, while human wants to take a shot of target object from different sides. Here, as we said previously and as shown in Figure 2, camera itself cannot see all sides of steady target, while interacting with FCC camera will be able see target object from all sides.

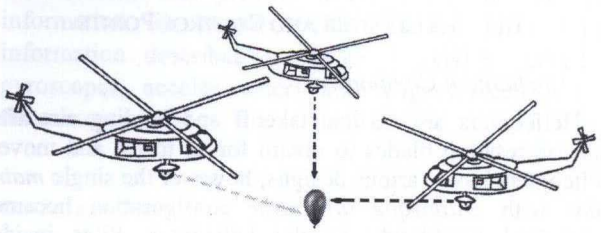

Figure 2. Target-Around mode

- Mode 3: Object moves from one location to another. Here camera to be able to fully track target object camera servos interacts with FCC. As shown in the left part of Figure 3, camera is not able track target if it moves to out of view location, which is marked as "out of view", hence in order to track object in that area, camera servos interacts with FCC and FCC giving commands to helicopter servos helicopter itself moves to appropriate position, (marked as "visible").

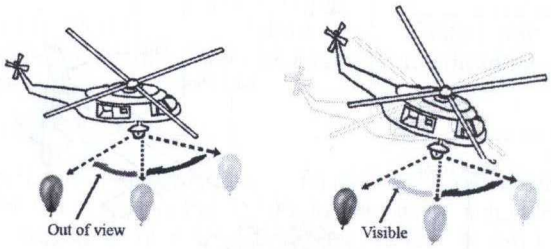

Figure 3. Target-Watch mode

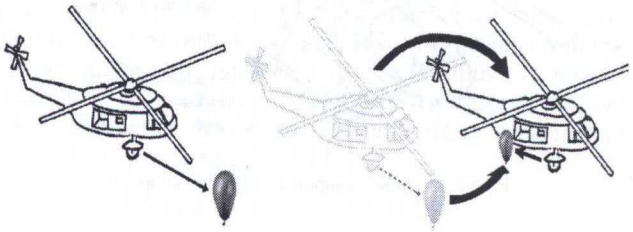

Figure 4. Target-Tracking mode

- Mode 4: Helicopter and object moving. In the last case both of helicopter and target object can move, however every time camera has to track target object, keeping it in the visible area. This task is more complicated from other modes. As shown in Figure 4 (left to right) in first phase object located in visible area to camera and in second phase target object moved from the first position to the second, which is out of camera's visible area and now horizontally far from camera itself. In this case camera servos interacts with FCC and computing appropriate commands to helicopter

servo actuators, it changes its longitude and latitude in order to point to target object properly.

III. HELICOPTER AND CONTROL POINTS

A. Mechanical Components

Helicopters are vertical takeoff and landing aircrafts that use rotating blades to obtain forces to lift and move. Helicopters have various designs; however the single *main rotor* with *antitorque tail rotor* configuration became recognized worldwide as the helicopter. Pilot inside helicopter has a number of control mechanisms to accomplish different tasks during flight and for flight.

In Figure 5, main aerodynamic components of helicopters are described, where they consist from rotors and stabilizers:

- main rotor, which generates thrust and maneuverability to the aircraft;
- tail rotor, which counteracts the helicopter body torque reaction;
- vertical and horizontal stabilizers, which contribute to maintain the aircraft in normal fight position (leveled off and nose forward);

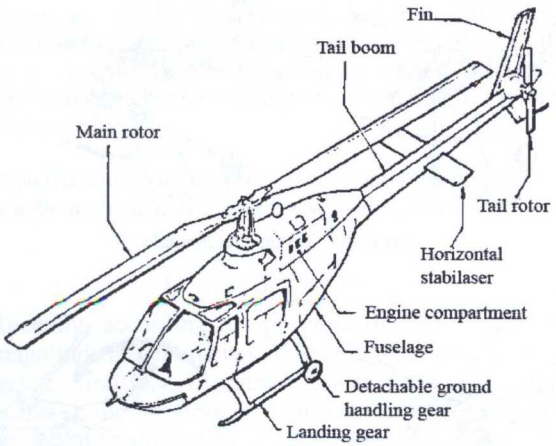

Figure 5. Main components of helicopters [5]

TABLE II. SIX DOF OF HELICOPTER

Action	Name	Velocity
movement along x	Longitudinal	Longitude velocity
movement along y	Lateral	Lateral velocity
movement along z	Altitude	Altitude velocity
movement around x: ZY plane	Roll	Yaw rate
movement around y: XZ plane	Pitch	Pitch rate
movement around z: XY plane	Yaw	Roll rate

B. Six Degree of Freedom

Six degrees of freedom(DOF) explained in TABLE II were named considering a three axes system with origin in the helicopter center of gravity. Helicopter can move along and around these three axes using its servo actuators and generating velocity named appropriately.

C. Control points and Swashplate

The vertical movements (take off and landing) depend on the relation between the aircraft weight and the lift force generated by the rotating blades. If this force is greater than the weight, the helicopter accelerates up; if it is less than the weight, it accelerates down; and if they are equal, the aircraft remains hovering at a constant altitude. In order to increase and decrease the intensity of the lift force the pilot acts on the "collective pitch control" or simply "collective".

The horizontal movements (longitudinal and lateral) occur when there is a horizontal force component. In order to generate this horizontal component, the lift force is inclined towards the desired direction. To control this inclination the pilot acts on the "cyclic pitch control", or simply "cyclic".

The "lateral cyclic" tilts the lift force left or right sideward, and the "longitudinal cyclic" tilts the lift forward and backward. These horizontal movements start with the helicopter's body inclination to the desired direction. So, if a forward motion is desired, the aircraft tilts the nose down. If the movement is to the right, the aircraft rolls right.

The pilot changes the aircraft azimuth acting on the "tail rotor collective control", or simply "tail". This control alters the force generated by the tail rotor, whose function is to react to the helicopter turning tendency [4].

The above four operational commands summarized in Table 2 are control points for the flight and are carried on through the device called swashplate that translates the pilot's commands into motion of the main rotor blades. Because the main rotor blades are spinning, the swashplate is used to transmit the pilot's commands from the non-rotating fuselage to the rotating rotor hub and main blades.

Due to its complexity, the navigation system is usually decomposed hierarchically into the following levels: mission interpreter level, task level and control level. The *mission* is a set of specific movements of the helicopter, called *tasks* such as take-off, flight forward, hovering, changing azimuth angle, and landing. Using the control points, the *control level* implements the control of the helicopter chosen by the task level.

D. Unmanned Helicopter

1) Flight Control Computer: In case of manual control of helicopter human pilot controls all measurement and rotors of helicopter, while in autonomous control human control is replaced with FCC (flight control computer), which controls entire flight system. FCC generates commands to all four control points shown in TABLE III in given frequency; however,

to calculate next step output, it needs to know current status of helicopter. While Camera is used to accomplish flying-camera tasks such as taking a shot of target object, current status information of helicopter is inputted from GPS(Global Positioning System) and AHRS(Attitude Heading Reference System), GPS provides information about Longitude, Latitude, Altitude and their velocity as mentioned in first three rows of TABLE II. AHRS is 3-axis sensors that provide heading, attitude and yaw information for aircraft, which is last three sensor information described in Table 3. AHRS consist of gyroscopes, accelerometers and magnetometers on all three axes.

TABLE III. HELICOPTER CONTROLS AND EFFECTS

Control Points	Directly controls	Primary effect	Secondary effect	Used in forward flight	Used in hover flight
Cyclic (lateral)	Varies main rotor blade pitch left/right	Tilts main rotor disk left and right through the swashplate	Induces roll in direction moved	To turn the aircraft	To move sideways
Cyclic (longitu-dinal)	Varies main rotor blade pitch fore/aft	Tilts main rotor disk forward and back via the swashplate	Induces pitch nose down or up	Control attitude	To move forwards/ backwards
Collective	Collective angle of attack for the rotor main blades via the swashplate	Increase/decrease pitch angle of rotor blades causing the aircraft to rise/descend	Increase/decrease torque and engine RPM	To adjust power through rotor blade pitch setting	To adjust skid height/ vertical speed
Tail rotor	Collective pitch supplied to tail rotor blades	Yaw rate	Increase/decrease torque and engine RPM (less than collective)	Adjust sideslip angle	Control yaw rate/heading

2) Ground Control System: Flight system including FCC is connected with GCS(Ground Control System) through communication block. The GCS is used to monitor the state of helicopter itself and manipulate camera to accomplish flying-camera tasks. Sensor information which provided by sensors of flight system is used to monitor helicopter status.

The flying-Camera tasks are given by human to accomplish the purposes such as taking overall/exact view of remote or hazardous location or shots of target object. Camera is used as flying-camera tool and it has its servos, while human can control its servos by *joystick*. Joystick is camera servo control point, while it can turn to right/left and rotate camera lens pointing to given target object. Camera is not able to point to target object if it moves out of camera's view. In order to achieve full view ability, camera servo works with FCC together and FCC computes servo command based on GPS, AHRS input and Camera servo input. As shown in Figure 6, human controls *camera joystick* which goes through communication to FCC and this command are used together with GPS, AHRS sensor information as input to FCC. This interaction of FCC and camera servo is used to accomplish more flexible flying-camera tasks.

IV. REAL-TIME SW PLATFORMS

The on-flight S/W embedded on the FCC intrinsically requires real-time computations related to getting attitude, heading and global position information from sensors, computing the commands for the next movement and feeding the commands out to the servo actuators, and communicating with GCS(Ground Control System). These series of computations can be decomposed into several threads executing chunks of corresponding codes as shown in Figure 7, and should be guaranteed to be finished within every specific period suitable for sustaining safe and stable flight; usually 20ms~50ms. Although it is validated in the design phase, the guarantee of the deadline is mostly dependent on the execution platform. In this chapter, we try to summarize the existing research efforts with simple comparison provided in TABLE IV.

A. CORBA-Based Middleware for UAVs

The OCP has middleware layer which provides the software layer isolating the application from the underlying target platform. It provides services for controlling the execution and scheduling of components, inter-component communication, and distribution and deployment of application components onto a target system [7, 8].

The OCP follows different approach than traditional control systems development process [7], providing the tools to map the design into the OCP's component architecture, going through these steps:

- The Controls designer takes the design generated from Matlab/Simulink and uses the OCP's tool to add to the design additional information about worse case execution times, allowable rates of

execution, scheduling type (Hard or Soft), physical layout, and data types;

- The OCP's tool generates an XML representation of the system;
- The XML Description of the system is provided as an input to the OCP's tool which generates the OCP component framework;

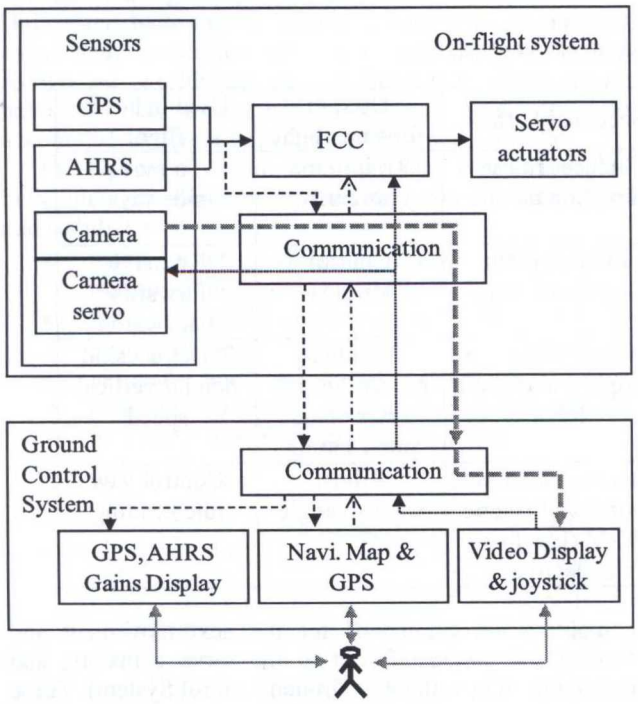

Figure 6. FCC and GCS interaction

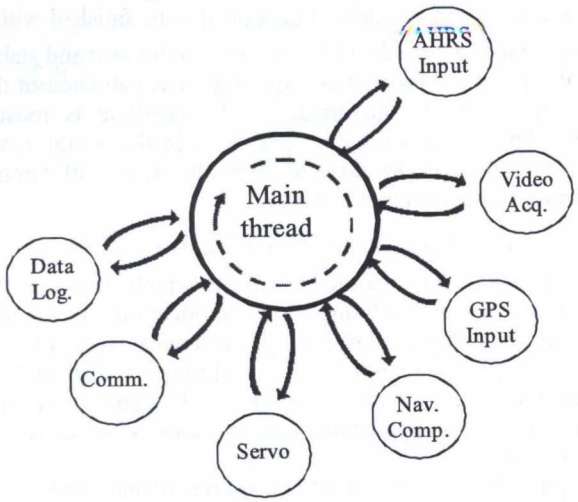

Figure 7. Sub-tasks of the computation in FCC

Finally the designer adds RT Workshop generated code to the OCP's component framework.

The OCP has its heritage in the mission avionics domain, which is a hard RT domain with rates typically up to 40 Hz. In contrast, the flight controls domain has much higher rates especially in the area of low-level inner loop control. This imposes more stringent requirements on the OCP middleware.

OCP provides a framework for encapsulating hardware specific tasks and exposing them to the application through a clean interface – the Timer Service. It provides a generic interface for creating both hardware and software timers. Hardware timers are the most accurate timers available through the controls API for component triggering, and they are triggered by one of the finite number of hardware timers resident on the processing hardware. Software timers, also available through the controls API, are less accurate than hardware timers, but are used to emulate hardware timers in software [8].

B. Giotto

Giotto is a domain-specific high-level programming language for control applications. A Giotto program explicitly specifies the exact real-time interaction of software components with the physical world, say differently Giotto program supervises the interaction between software processes and the physical world, but does not itself transform data [9]. All computation is encapsulated inside the supervised software processes, which can be written in any non-embedded programming language, such as C and using that high level Giotto program, Giotto compiler automatically generates timing code that ensures the specified behavior on a given platform [10].

Giotto uses embedded software model, which separates the platform-independent from the platform-dependent issues, this facilitates code generation by partitioning the task into two steps. In the first step, called program generation, a given mathematical model is transformed into an embedded software model which is entirely independent of any execution platform and it specifies only the reactivity (i.e., real-time response) of the system relative to a physical environment. In the second step, called compilation, the software model is transformed into executable code for a target platform, while in this step ensures the schedulability (i.e., real-time execution) of the system in a specific execution environment [10]. A Giotto program specifies only the reactivity of the functionality programs – i.e., when they are invoked, and when their outputs are read – but not their scheduling.

C. TMO-UAV

The TMO (Time-triggered Message-triggered Object) is 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.

TMO is natural, syntactically minor, and semantically powerful extension of the conventional objects, while it can be thought as a high-level real-time computing object. TMO has member functions, which are executed within specified windows in the domain of global time [11].

In case of real-time distributed computing applications , TMO model can be used as a fundamental building-block,

while TMOSL (TMO Support Library) as development API for TMO programmers and TMOSM (TMO Support Middleware) as execution engine for real experiments [11].

The basic TMO structure consists of four parts:

- Spontaneous Method (SpM): A time-triggered method which is triggered when the real-time clock reaches specific values determined at design time.
- Service Method (SvM): A method similar to the conventional service method which is triggered by service request messages from clients.
- 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).
- Environment Access Capability (EAC): A list of entry points to remote object methods, logical communication channels, and I/O device interfaces.

In order to develop real-time distributed computing applications with TMO model, first each functional block of distributed application is divided into separate TMO. And they will interact with each other by using TMO methods, hence accomplishing given task in given time.

In TMO model, whole system works by interaction of separate TMOs (UAV TMO and Ground Station TMO – in case of UAV control system) and each TMO consists from sequence of tasks, called TMO methods. In TMO, the designer guarantees and advertises execution time-windows bounded by start times and completion times. Special features of TMO methods enable the designer of each TMO to provide guarantee of timely service capabilities of the object [12].

D. TMO-HILS

TMO-HILS is an architecture of HILS (Hardware-In-the-Loop Simulation), which support multi-vehicle simulation. The TMO (Time-triggered Message-triggered Object) is a real-time object model that can be used for hard and soft real-time applications and parallel computing. With the TMO model, both functional and timing behaviors of a system can be explicitly specified. By using the TMO, building the HILS is simplified dramatically on design and implement. Universality and integrity are the characteristics of this HILS, which is able not only to test the different embedded systems respectively, but also to test them in one virtual environment simultaneously.

TMO-HILS system supports decentralized and distributed test of multiple real-time systems which are based on TMO. The TMO-HILS simulation system provides the user with great confidence in the mission execution for real tests and dramatically reduces the risk of costly errors [12].

TMO-HILS system consists of Virtual Environment, V-O Interface (Virtual Environment - Objects), Simulation Engines and Objects. The TMO-HILS provides the Virtual Environment in which the objects may be tested with some missions to simulate some real test environment. The objects connect with Virtual Environment not directly, but via an interface, named V-O interface, which consists of Bridge TMO. The Simulation Engines module provides the powerful support to whole TMO-HILS system with generating and analyzing the real data of devices.

V. CONCLUSIONS

In this paper, we introduced the target scenario of our HELISCOPE project with the emphasis on the four modes for flying-camera task. We also explained the control points that are considered as the interfaces where the on-flight embedded software and mechanical implementation of the unmanned helicopter are contacted for the autopilot necessarily used in performing such flying-camera tasks. Being related to real-time issues, we also introduced S/W platforms including TMO model and execution engine.

While GIOTTO and CORBA-UAV provide outstanding features such as domain-specific high-level language, timing check in the code generation time, isolation of application from target platform by using components, TMO is expected to provide a consistent scheme that can be applied in the real-time specification, the implementation, simulation and execution phases.

ACKNOWLEDGEMENT

This research was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute for Information Technology Advancement) (IITA-2008-C1090-0804-0015).

REFERENCES

[1] http://www.wibro.or.kr/new/overview01.jsp

[2] Nomor Research, "Technology of High Speed Packet Access", http://www.nomor.de/uploads/b0/2m/b02mwrVvIa5ZUtVrFeSP1w/Technology_of_HSPA.pdf

[3] R.W.Prouty, "Helicopter Performance, Stability and Control", Robert E. Krieger Publishing Company, Malabar, Florida, 1990;

[4] L.Collier, K.Thomas, "How to Fly Helicopters", TAB Books Inc., Blue Ridge Summit, 1986;

[5] E.H.J. Pallet, "Automatic Flight Control", Granada, London, 1983;

[6] www.wikipedia.org - Wikipedia, the free encyclopedia.

[7] Dr. James L. Paunicka, Dr. David E. Corman, Brian R. Mendel, "A CORBA-Based Middleware Solution for UAVs", *4th ISORC*, 2001, pp. 261.

[8] Tariq Samad, Gary Balas, *Software-Enabled Control*, CHAPTER 4, "Open Control Platform: A Software Platform Supporting Advances in UAV Control Technology," Wiley InterScience. pp 39-62.

[9] Thomas A. Henzinger, Benjamin Horowitz, Christoph Meyer Kirsch, "Giotto: A Time-triggered Language for Embedded

Programming" *Proceedings of the IEEE* Volume 91, Issue 1, Jan 2003 Page(s): 84 - 99.

[10] Thomas A. Henzinger, Christoph M. Kirsch, Marco A.A. Sanvido and Wolfgang Pree, "From Control Models to Real-Time Code Using Giotto," *IEEE Control Systems Magazine* 23(1):50-64

[11] Hansol Park, Moon Hae Kim, Chun-Hyon Chang, Keechon Kim, Jung-Guk Kim and Doo-Hyun Kim, "Design and Experimental Validation of UAV Control System Software Based on the TMO Structuring Scheme," *SEUS 2007, LNCS 4761*, pp. 419–428.

[12] J. Xu, H. Park, K. Jeong and C.H. Chang, "TMO-HILS Architecture for Real-Time Control System", *Int'l Conference on Control, Automation and Systems* 2007, Seoul, Korea, 2007, pp. 1855-1861.

TABLE IV. COMPARISON OF GIOTTO, CORBA-UAV AND TMO-UAV

	GIOTTO	CORBA UAV	TMO-UAV
Definition	Giotto is domain-specific high-level programming language for control applications.	CORBA-Based Middleware is an open, middleware-enabled software framework and development platform designed to analyze, develop, and test UAV control algorithms and embedded software.	The TMO is real-time object model that can be used for various types of hard and soft real-time distributed computing applications.
Timing features	Timing issues are provided in code generation time, which checked by Giotto compiler.	Uses hardware and software timers, which is provided Open Control Platform (OCP) framework.	The designer guarantees and advertises execution time-windows bounded by start times and completion times.
Compatibility	Can use code generated by MathWorks' Real-Time Workshop and C language objects.	Can use code generated by MathWorks' Real-Time Workshop and C language objects.	Uses TMOSL (TMO Support Library) which is developed using C++.
Additional features	Giotto program supervises the interaction between software processes and the physical world, but does not itself transform data.	The OCP has middleware layer which provides the software layer isolating the application from the underlying target platform.	TMO model can be used as a fundamental building-block, while TMOSL (TMO Support Library) as development API for TMO programmers and TMOSM (TMO Support Middleware) as execution engine for real experiments.

Published by the IEEE Computer Society
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314

IEEE Computer Society Order Number P3573
BMS Part Number: CFP09175-PRT
Library of Congress Number 2009900279
ISBN 978-0-7695-3573-9