A Hybrid Model-Based Realization to Deploy the Controller for a Micro Unmanned Aerial Vehicle

Ngo Van Hien¹, Pham Gia Diem¹, Nguyen Phu Hung²

¹ Hanoi University of Science and Technology (HUST), Hanoi, Vietnam
² National Center for Technological Progress (NACENTECH), Hanoi, Vietnam

Emails: hien.ngovan@hust.edu.vn, diem.phanggia@hust.edu.vn, hungnp.ite@gmail.com, * Corresponding author

Abstract— This paper introduces a hybrid control model, which is based on the Model-Driven Architecture (MDA) approach combined with the real-time Unified Modeling Language (UML). Extended Kalman Filter (EKF) algorithm and hybrid automata, in order to conveniently deploy controllers of micro Unmanned Aerial Vehicles (UAVs) such as the Quadrotor UAV (Q-UAV). This model also creates a capsule-based collaboration pattern, which can permit the designed control components to be customizable and reusable in new application developments of various UAV typed Vertical Take Off and Landing (VTOL). The paper brings out stepwise the dynamics and control architecture of a Q-UAV for control inputs that are then combined with the specialization of MDA features as follows: the Computation Independent Model (CIM) is defined by the specification of use-case model together with the EKF algorithm and hybrid automata to define the implementation analysis for control; the Platform Independent Model (PIM) is then designed by specializing the real-time UML features including main control capsules that depicts in detail structures and behaviors of the controller; the detailed PIM is subsequently transformed into the Platform Specific Model (PSM) by object-oriented open-source platforms to rapidly deploy the Q-UAV controller. Based on this proposed model, a trajectory-tracking controller was developed and tested that permits a Q-UAV to reach and follow the desired reference trajectory.

Index Term— Micro Unmanned Aerial Vehicle (UAV); Quadrotor UAV Control; Hybrid Automata; Extended Kalman Filter (EKF); Real-Time UML; Model-Driven Architecture (MDA).

Abbreviations

<table>
<thead>
<tr>
<th>CIM</th>
<th>CLF</th>
<th>DoF</th>
<th>EKF</th>
<th>GPS</th>
<th>HA</th>
<th>HDS</th>
<th>IB</th>
<th>IDE</th>
<th>ICGB</th>
<th>IMU</th>
<th>LQ</th>
<th>LOS</th>
<th>MBSE</th>
<th>MDA</th>
<th>MDE</th>
<th>OMG</th>
<th>PID</th>
<th>PIM</th>
</tr>
</thead>
</table>

I. INTRODUCTION

For a long time, Unmanned Aerial Vehicles (UAVs) have been more extensively known in many military applications. Actually, micro UAVs, especially Quadrotor UAVs (Q-UAVs), are also being developed rapidly for use in civilian applications, e.g. environmental monitoring, forest protection, wildfire detection, traffic monitoring, building, power line and bridge inspection, emergency response, crime prevention, search and rescue, mapping, surveillance, etc. [1-4].

The autonomy architecture of general UAV basically consists of three main sub-systems as follows: the guidance sub-system, navigation sub-system and control system. All three of these sub-systems have their own individual tasks to complete, yet must also work cooperatively in order to reliably allow an UAV to complete its objectives. The problem of designing autonomous flight controllers for Q-UAVs is equally challenging because these controllers are closely connected with the dynamic models. Hence, the Q-UAV controller must take into account models with discrete events and continuous behaviors that can be considered as a Hybrid Dynamic System (HDS) of which global behaviors can be modelled by Hybrid Automata (HA) [5-7].

Besides, the low development costs must be taken into account in the application construction. Thus, the reusability is also needed to be considered in the new development lifecycle of various UAV applications. According to the Object Management Group (OMG) [8], the Unified Modeling Language (UML) has been standardized for analyzing and designing visually system components in the software industry. In addition, the System Modeling Language (SysML) [9] is a UML profile for systems engineering that has been also standardized by OMG. SysML is used for the analysis, design, verification and validation of industrial systems in various domains. Nevertheless, both UML and SysML lack the constructs for modeling precisely timing and communication evolutions between the control objects for real-time and embedded systems, e.g. the Q-UAV controller. Furthermore, OMG have also standardized the Model-Driven Architecture (MDA) [10] whose main idea is to separate the
specification of system operations from the details of the way that system uses the capabilities of its platform. MDA supports a framework for, and enables tools to be provided for: specifying a system independently of the platform that supports it; specifying platforms; choosing a particular platform for the system; and transforming the system specification into one for a particular platform. On the other hand, the Model-Based Systems Engineering (MBSE) has been formalized by INCOSE [11, 12] to support the modeling of system requirements, design, analysis, verification, and validation artifacts in the development lifecycle of complex systems. Thus, MDA defines a general system approach, which focuses on the use of rigorous visual modeling techniques throughout the whole of development lifecycle, and provides a skeletal solution, whose details can be the MBSE specialization to systems engineering applications. The applications of the about model-driven approaches for the implementation of industrial control systems can be found in [13-22]. For these reasons, the MDA’s features can be specialized by using the real-time UML version [23-26] in order to model in detail the analysis and design artifacts for real-time and embedded control systems, e.g. the Q-UAV controller. This version also includes the ‘capsules, ports, protocols, connectors’ concepts that can be adapted by specializing a set of control capsules to model in detail behaviors and structures of the Q-UAV controller.

This study is focused on the development of a control model integrated the Q-UAV dynamics using the MDA approach combined with the real-time object paradigms, e.g. the real-time UML, the specialization of HA features and the Extended Kalman Filter (EKF) algorithm, which can permit us to conveniently implement the Q-UAV controller. This model also allows the designed and implemented control elements to be closely customizable and re-usable in the realization of new applications for various UAV types capable of VTOL. In this study, the dynamics and control architecture of a Q-UAV are also adapted for the control inputs that are then combined with the specialization of MDA features composed of the Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM); the control system permits a Q-UAV to track the desired reference trajectory in the Cartesian space. In detail, the CIM includes the use-case model specialized with an implemented function block diagram, the supplemented EKF algorithm and HA to precisely capture the requirement analysis for a Q-UAV controller; the PIM is built on the identified CIM by specifying the real-time UML to create a capsule-based collaboration pattern to entirely design the control components. The detailed PIM elements is then converted into PSMs by open-source platforms such as Arduino [27] in order to rapidly realize and deploy the Q-UAV controller. Finally, a trajectory-tracking controller of an application of Q-UAV was deployed and tested.

The paper is structured as follows: The second section brings the related works that have inspired us to define a model-based realization for Q-UAV controllers. The dynamic model and control architecture of a Q-UAV are introduced in the third section. The fourth section presents the details of a proposal of executable MDA process to conveniently realize Q-UAV controllers, including the CIM, PIM and PSM components. Following this proposed control model, in the fifth section, it is applied to a case study. Conclusions and future work are reported in the final section.

II. RELATED WORK

In present design of complex systems, there were many applications that have used the standard control methods combined with soft computing approaches to make them more effective for controllers of such systems [28-35]. These were also applied to the construction of Q-UAV controllers, for example, Lazim, Husain, Basri, et al. [2] have introduced an enhanced disturbance observer for improving the robustness of autonomous quadrotor flight, which was combined with the presence of wind disturbances; their study was achieved by integrating the artificial intelligence via Radial Basis Function Neural Network (RBFNN) to predict the bounded disturbance estimation error produced by the standard disturbance observer of controllers. A robust controller has been proposed by Liu, Li, Zuo, et al. [36] to address the attitude control problem for uncertain quadrotors with input delays; their designed controller included a nominal controller to achieve desired tracking for the nominal system and a robust compensator to achieve the robust stability of the uncertain system with input delays. The discrete-time Sliding Mode Control (SMC) scheme was applied to design the discrete-time flight controllers for the position and attitude tracking control of a Q-UAV that was presented in [37]; under the discrete-time controllers, the six Degrees of Freedom (DoF) respectively converge to their desired values, the position and velocity tracking errors of all states converge to zero, i.e. the sliding manifolds converge to their sliding surfaces; the obtained simulation results are then predicted for the real model of the Q-UAV and other complex environments. A hybrid feedback control strategy has been implemented by Smith III and Sanfelice [38] that unites two state-feedback controllers; a transit controller capable of steering or transitioning the UAV to nearby the waypoint and a loiter controller capable of steering this vehicle about a loitering radius; for this application of hybrid feedback control, Lyapunov functions and hybrid systems theory have employed to establish stability properties of the defined loitering set of points. A hierarchical control strategy based on adaptive radical basis function neural networks and double-loop Integral SMC (IntSMC) has been developed by Li, Wang, Tan, et al. [39] for tracing the position and attitude of Q-UAVs that were subjected to sustained disturbances and parameter uncertainties; capabilities of online adaptive estimating of the unknown uncertainties and null tracking error were proved then by using the Lyapunov stability theory; their simulation results were compared also with traditional PD/IntSMC algorithms and with the backstepping/nonlinear $H_{\infty}$ controller that permit to verify the effectiveness and robustness of their proposed control laws. Various controllers based on Lyapunov theory, PID, Linear Quadratic (LQ), Backstepping and SMC techniques were implemented to the control design of a miniature Q-UAV, and were compared for attitude control that could be found in [40].

Following the above standard control methods, the designed control elements could be difficult to customize and
reuse for realizing controllers of different UAV types of which operating modes are capable of VTOL, and for deploying appropriately into various software and hardware platforms. To achieve this goal, the MDA approach can be implemented by using the real-time UML to conveniently perform the whole of development lifecycle focused on control systems such as the Q-UAV controller. The three main goals of MDA are portability, interoperability and reusability through architectural separation of concerns. Here, the portability allows the same solution to be realized on new or multiple platforms; the interoperability creates systems that can easily integrate and communicate with other systems and use a variety of resource applications: the reusability builds solutions that can be reused in many different applications in different contexts. Based on the above model-driven approaches, Ragavan, Shamugavelan, Ganapathy, et al. [41] have proposed the Bond Graph based Unified Meta-Modeling Framework (BG-UMF) to lead the complexity in model transformation, analysis, validation, verification and automatic code generation, which are focused on the conceptual design and development of executable models for large engineering systems. Herrera, Posadas, Peñil, et al. [42] have introduced the complex UML/MARTE (Modeling and Analysis of Real-Time and Embedded Systems) methodology for design space exploration of embedded systems, which was based on a novel combination of Model-Driven Engineering (MDE), Electronic System Level and design exploration technologies. This framework could enable capturing the set of possible design solutions, that is, the design space, in an abstract, standard and graphical way by relying on the real-time UML profile. In [43], the advantages of model-based techniques have explored to automate the implementation of the supervisor and its integration into the final real-time control system from the simulation dynamic model. This model-driven engineering approach has been followed in order to provide a framework to support the development cycle of the control system for mode switched processes. It uses information from the model of the process and from the control loops designed and using model transformation techniques that the supervisor is automatically generated. In particular, the use of real-time UML and model-driven approaches for the design of real-time embedded software and systems can be found in [13-22].

Summarizing this section, we cited the structural implementations for Q-UAV controllers and the model-based methodologies for real-time and embedded systems that could be used to create a hybrid model-based realization to conveniently deploy the Q-UAV controller.

\[
\begin{align*}
\dot{x} = & \left( \sin \psi \sin \phi + \cos \psi \sin \theta \cos \phi \right) \sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{xi} - \frac{1}{2} C_{\psi A} \rho \| \dot{x} \|^2 |
\end{align*}
\]

\[
\begin{align*}
\dot{y} = & \left( -\cos \psi \cos \phi + \sin \psi \sin \theta \cos \phi \right) \sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{yi} - \frac{1}{2} C_{\theta A} \rho \| \dot{y} \|^2 |
\end{align*}
\]

\[
\begin{align*}
\dot{z} = & mg - \left( -\cos \psi \cos \phi \right) \sum_{i=1}^{4} T_i
\end{align*}
\]

\[
1_{xx} \dot{\psi} = \dot{\theta} \psi (1_{yy} - 1_{zz}) + 1_{l_y} \Omega_r + l (1_{l_y} + 1_{zz}) - h (\sum_{i=1}^{4} H_{yi}) + (-1)^{i+1} \sum_{i=1}^{4} R_{mxi}
\]

\[
1_{yy} \dot{\theta} = \dot{\phi} \psi (1_{zz} - 1_{xx}) - 1_{l_y} \Omega_r + l (1_{l_y} - 1_{zz}) - h (\sum_{i=1}^{4} H_{yi}) + (-1)^{i+1} \sum_{i=1}^{4} R_{myi}
\]

\[
1_{zz} \dot{\phi} = \dot{\phi} \psi (1_{xx} - 1_{yy}) + 1_{l_y} \Omega_r - (\frac{1}{2}) \sum_{i=1}^{4} Q_i + l (1_{l_y} - H_{yy}) + h (1_{l_y} - H_{yy})
\]

Where: \(I_\alpha, I_\beta, I_\gamma\) and \(l_\alpha\) are inertia moments; \(\phi, \theta, \psi\) are respectively Roll, Pitch, Yaw (RPY) angles; \(J_r\) presents the rotor inertia; \(H\) is a set of hub forces; \(R_m\) is a set of rolling moments; \(T_i\) presents the thrust force \((i = 1, 4)\); \(\Omega\) is the overall residual propeller angular speed; \(A_c\) is fuselage area; \(C\) is the propulsion group cost factor; \(\rho\) is the air density; \(Q\),

III. OVERVIEW OF DYNAMIC MODEL AND CONTROL ARCHITECTURE OF A Q-UAV

A. Q-UAV Dynamic Model for Control

Q-UAVs have four fixed-pitch propellers in cross configuration as shown in Fig. 1. There are four rotors with fixed angles which represent four input forces \([T_1, T_2, T_3, T_4]\) that are basically the thrust generated by the propellers’ angular speeds \([\omega_1, \omega_2, \omega_3, \omega_4]\). Modeling the rigid body dynamics aims at finding the differential equations that relate system outputs (position and orientation) to its inputs (force and torque vectors). From the large field of guidance, navigation and control of aerial vehicles in [40, 44-50], the 6 DoF dynamic model of an UAV in body coordinate frame can be written in general form (1).

\[
\mathbf{M} \ddot{\mathbf{v}} + \mathbf{C}(\mathbf{v})\mathbf{v} + \mathbf{D}(\mathbf{v})\mathbf{v} + \mathbf{g}(\eta) = \tau(\mathbf{v}, \mathbf{u})
\]  

(1)

Where: \(\eta = (x, y, z, \phi, \theta, \psi)\) is the position (NED: North, East and Down) and orientation (Euler: Roll, Pitch and Yaw angles); \(\mathbf{v} = (u, v, w, p, q, r)\) is the velocity and angular velocity; \(\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A\) is a mass matrix, which denotes the 6x6 system inertia matrix containing, \(\mathbf{M}_{RB}\) - the generalized constant inertia matrix, and \(\mathbf{M}_A\) - the aerodynamic added mass inertia matrix; \(\mathbf{C}(\mathbf{v})\) is the 6x6 Coriolis matrix; \(\mathbf{D}(\mathbf{v})\); \(\mathbf{g}(\eta)\) is the 6x1 vector of gravitational effects; \(\tau(\mathbf{v}, \mathbf{u})\) is the vector of resultant force and moment acting on the aerial vehicle, and \(\mathbf{u}\) is the control inputs, e.g. the rotational speed of the motors related to the generated thrusts. The detailed dynamics for control of UAVs can be seen in [44, 47].

Fig. 1. A Q-UAV model.
presents the drag moment; h and l are respectively vertical
distance and horizontal distance: propeller center to Center Of
Gravity (CoG); x, y and z define the position in body
coordinate frame.

In this study, we proposed the following assumptions: the
hub forces and rolling moments are ignored and thrust and drag
coefficients are considered as constants in order to comply with
the real-time constraints of the embedded control loop. Thus,
the control evolution of Q-UAV can be rewritten in state-space
form $\dot{x} = f(x, u)$ with $u$ inputs vector and $x$ state vector.

$$u = [u_1, u_2, u_3, u_4]^T$$  (3)

Here, $u_i$ is the control input ($i = \{1,4\}$; $x$ is a 12-dimensional
state vector for describing the motion of Q-UAV that is
written as follows:

$$x = \begin{bmatrix} x_1 \ y_1 \ z_1 \ \phi_1 \ \psi_1 \ \theta_1 \ v \ w \ p \ q \ r \end{bmatrix}^T = \begin{bmatrix} x \ y \ z \ \phi \ \psi \ \theta \ v \ w \ p \ q \ r \end{bmatrix}^T$$  (4)

B. Proposal of Control Architecture for a Q-UAV

As the previously stated, there are three main sub-systems
within the physical structure of a Q-UAV proposed as follows:
the guidance sub-system is used for providing the desired path
for the Q-UAV track; the navigation sub-system is responsible for executing an estimation of the current state of the Q-UAV;
and the control sub-system is used for calculating and applying
the control forces and moments to conduct this Q-UAV. These
sub-systems have their own individual tasks to complete, yet
must also work cooperatively in order to reliably allow a Q-
UAV to complete its missions even in the presence of unknown
environmental disturbances. Fig. 2 shows out a functional
diagram, which captures how these sub-systems interact.
Furthermore, the control sub-system of any Q-UAV composes of a sub-block of control law and a sub-block of control
allocation. The first is responsible for generating the
generalized forces and moments in 6 DoF based on current and
desired states, while the second is responsible for distributing
this generalized forces and moments amongst the actuators of Q-UAV in order to realize the navigation task allocation.

![Fig. 2. Functional block diagram for presenting the autonomy architecture of Q-UAV.](image)

From the above described Q-UAV dynamic model and
general control architecture together with characteristics of HDS in [7, 51, 52], we find that controllers of Q-UAVs are
HDS whose dynamic behaviors can be modeled by HA [5-7].
These control systems have the continuous/discrete parts and
their interactions such as the motional components, e.g. horizontal transferring, VTOL, rotation, roll, pitch and yaw,
and external interacting events from the guidance and
navigation sub-systems, and environmental disturbances. In
this study, we are interested in developing the trajectory-
tracking controller of Q-UAVs, so we can use this hybrid
dynamic model to find out the control algorithms with a
specific guidance law such as the Line-Of-Sight (LOS) guidance implemented in [53-56].

IV. AN EXECUTABLE MDA PROCESS TO DEVELOP A Q-
UAV CONTROLLER

Starting from the above adapted Q-UAV dynamics and
control architecture and real-time UML features, we propose here an executable MDA process (Fig. 3) to develop the Q-
UAV controller, which includes three main models as follows:
CIM, PIM and PSM.

![Fig. 3. Executable MDA process for developing Q-UAV controllers.](image)

i) In the CIM, the use case model is specialized by the
implemented functional block diagram, EKF algorithm and
HA to closely capture the requirements analysis for a Q-UAV
controller.

ii) The PIM is built up by specifying the real-time control
algorithm, ports, protocols and their timing evolutions together
with the EKF algorithm and HA evolution in order to model
the precise behaviors and structures of Q-UAV controller.

iii) The PIM is then converted into the PSM by Object-
Oriented (OO) open-source specific platforms such as Arduino
[27] in order to rapidly realize and deploy the controller.

In the implementation model, there are also transformation
rules, which allow the designed PIM to be converted into the
PSM. All of artefacts and activities in CIM, PIM, PSM and
model transformations will be gone into detail in the next sub-
sections.

A. CIM for a Q-UAV Controller

The goal of the CIM is to entirely model the problem in
business terms and without getting into the solution or how
it might be implemented. In the CIM, object collaborations with
the real-time UML, which are based on the use case model
together with the specifications of continuous behaviors
including the implemented functional block diagram, Integral
Backstepping (IB) technique and EKF algorithms and HA
in order to closely capture the requirements analysis for a Q-
UAV controller.

The main use case model of a Q-UAV controller is defined
as shown in Fig. 4 combined with an example of trajectory-
tracking scenarios and local state machine of the “Track a
desired trajectory" use case, which are respectively shown in Fig. 5a and Fig. 5b. In Fig. 5a, the "loop(5)" fragment is typical value in practice of LOS guidance, which can be found in [57].

Where:
- MDS is the Measurement-cum-Display System consisting of the guidance and navigation systems, because both of them essentially act as a signal supplier for the controllers of Q-UAV;
- AES is the Air Environment System including disturbances generated by the weather;
- The “Track a desired trajectory” use case is performed for tracking the desired trajectory for the vehicle to follow;
- The “Ensure safety” use case is realized for maintaining system safety when one of its component fails or supplied power is low or bad weather;
- The “Configure” use case permits users or maintainers to configure and update control parameters for starting up the controllers;
- The “Maintain” use case is performed for maintaining the whole system that includes activities, e.g. the error identifications and corrections of the whole of physical Q-UAV or periodical maintenance.

![Diagram](image)

Fig. 4. Use case diagram for capturing the main control requirements of the Q-UAV.

In the use case model, it is necessary to set up industrial maneuvering constraints, e.g. the maximum tilted angle, velocity, altitude and other safe flying modes of the developed Q-UAV in order to ensure the operational safety of this system. In addition, an implementing functional block diagram must be built for modeling continuous dynamics of the Q-UAV controller; because the real-time UML lacks the constructs for depicting internal continuous models for each state on the state machine diagram. Thus, we propose an implemented functional block diagram to gather the internal continuous behaviors of the Q-UAV controller as shown in Fig. 6.

Where:
- **Desired trajectory** and Take off/Landing are events, which are used for providing respectively the desired position and altitude to the blocks of position and altitude control.
- $\omega_i$, $i = 1, 4$ are desired rotational speeds, which are applied to the 4 motors.

![Diagram](image)

Fig. 5. A desired trajectory-tracking scenario (a) and local state machine (b) for performing the “Track a desired trajectory” use case.

- $\Sigma T$ and $r_{\theta \phi \psi}$ are the overall output forces and moments acting on the Q-UAV.

![Diagram](image)

Fig. 6. A functional block diagram for implementing the continuous behaviors of the Q-UAV controller.

In this study, the IB technique combined with navigation filters such as the EKF algorithm [58, 59] are used for
estimating altitude, position, attitude and velocity control. The IB expansions combined with the CLF for Q-UAV controllers were well known by many Q-UAV control applications, for instance [60, 61]; the PID regulators can be applied to the functional block of Motor Control in order to reduce the inertial and delay time caused by the physical Q-UAV actuators in the whole system evolution. The application details of EKF algorithm for a Q-UAV controller will be presented in Section V.

In the CIM, Hybrid Automata (HA) are specified to present the mathematical implementation model of the Q-UAV controller consisting of terms as the Situations, Continuous State Variables, Event, Transition, Global Continuous Behavior and Invariants. A HA of the Q-UAV controller \((H_{Q-UAV})\) is established by data as follows:

\[
H_{Q-UAV} = (Q, X, \Sigma, A, Inv, F, q_0, x_0)
\]  

(5)

Here, \(Q\) is a set of operation modes called the situations of \(H_{Q-UAV}\), e.g. the modes of Hovering, Transferring, Taking off, Landing, etc.; \(q_0 \in Q\) is the initial situation; \(X\) is the continuous state-space of continuous elements of the Q-UAV, e.g. the control blocks of motor, position, altitude, etc.; \(x_0\) is the initial value of this space; \(\Sigma\) is a set of external interacting events for triggering from the current situation to the reached situation, which is issued from the guidance and navigation systems, and environmental disturbances for the Q-UAV controller; \(A\) is a set of transitions between the situations that are linked the appropriate events \(e \in \Sigma, Inv\) is called the invariant term of the situation for verifying when the situation is \(q\), the continuous state must be then \(x = inv(q)\); \(F\) is the global continuous models issued from by the 6 DoF dynamic model in equations (1)-(4) and the implemented functional block diagram (Fig. 6); the global evolution of continuous state is performed when a new situation is reached in the operation modes of the Q-UAV. The detail of HA specialization and its realization hypotheses for the Q-UAV controller can be seen in the authors’ reports [1, 6].

B. PIM for a Q-UAV Controller

In the detailed design phase of system, we transform the identified CIM into PIM, which is based on the use case approach [8, 9], real-time UML profile [23-26] and IBM Rational Rose RealTime or IBM Rational Software Architect RealTime tools [62]. The goal of the PIM is to closely build up the real-time control capsules, ports, protocols and their timing evolutions, which allow designing the precise behaviors and structures of Q-UAV controller. From the above defined CIM components, we have developed the 5 main control capsules, which take part in the HA realization of the Q-UAV: the continuous part’s capsule, discrete part’s capsule, internal interface’s capsule, external interface’s capsule and Instantaneous Global Continuous Behavior (IGCB) capsule. Fig. 7 and Fig. 8 indicate the capsule-based collaboration pattern and its timing evolutions for the Q-UAV controller.

Here, the discrete part’s capsule consists of the situations \(Q\) and transitions \(A\) in HA of the Q-UAV controller; the continuous part’s capsule contains the continuous state-space \(X\); the IGCB’s capsule builds up concrete global continuous behaviors as \(f e^F\). \(f\) is derived from equations (1)-(4) and the implemented functional block diagram (Fig. 6) can be implemented in \(f\) for estimating the values of position, altitude, attitude and velocity of the Q-UAV; the internal interface’s capsule permits the Inv tool to generate internal events in the HA evolution; the external interface’s capsule is an intermediary, which receives or sends episodic events and periodic signals between the Q-UAV controller and their interacting systems such as the AES and MDS.

In Fig. 8, the messages exchanging between the main control capsules are synchronous, and the interval between two adjacent timeout messages indicates the sampling period (\(\Delta T\)) of the IGCB’s capsule. The external interface’s capsule receives period signals coming from external continuous components. It then gives the ContinuousElement message to the IGCB’s capsule so that the IGCB’s capsule can call all of the continuous elements corresponding to the concrete ‘IGCB: IGCB1’. During the call of the IGCB’s capsule, the external interface’s capsule can receive an event named ‘3: InputEvent’ issued from the MDS or AES, and gives this event named ‘e3: DetectedEvent’ to the discrete part’s capsule. The discrete part’s capsule then memorizes and later processes this event. If the IGCB’s capsule receives the LastContinuousElement message coming from the continuous part’s capsule, then it gives the ContinuousEvolution message to the continuous part’s capsule so that the internal interface’s capsule can receive all updated variables. The internal interface’s capsule then verifies the invariant \((x, e Inv(q))\) of the situation \(q_2\); in this case, there is a generated internal event. The internal interface’s capsule gives this event to the discrete part’s capsule that permits the IGCB’s capsule to identify the concrete ‘IGCB: IGCB2’ and give output signals to the external interface’s capsule. At the end of this sampling period, the external interface’s capsule gives the output event and control signals to the external environment of the Q-UAV operating with its concrete ‘IGCB: IGCB2’.

In addition, the reusability is very important to implement controllers for various UAV types because it allows reducing the cost of application deployments. The specializations, which permit the capsule collaboration of a developed Q-UAV to be customizable and reusable in the new control application for various UAVs typed VTOL, are summarized in Table I.
Table I

<table>
<thead>
<tr>
<th>Designed control capsules</th>
<th>Generic artifacts</th>
<th>Specialization rules</th>
<th>Specialized artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete capsule</td>
<td>The discrete part’s capsule is not changed in the overall design for the new controller of UAV typed VTOL.</td>
<td>- Non</td>
<td></td>
</tr>
<tr>
<td>Continuous part</td>
<td>The ports and protocols of this capsule are not changed in the overall design for the new controller of UAV typed VTOL.</td>
<td>The continuous part’s capsule is specialized by supplementing or cutting down continuous components (i.e. ( x )) that depend on the physical actuators installing on the new UAV typed VTOL. The states and their behaviors, which correspond to the supplemented/cut down continuous elements, are supplemented/cut down in/from the state machine of this capsule. The behavior of the new set of continuous elements will be used to redefine the concrete IGCBs (( f )).</td>
<td></td>
</tr>
<tr>
<td>IGCB</td>
<td>The state machine, ports and protocols of this capsule are not changed in the overall design for the new controller of UAV typed VTOL.</td>
<td>The specification of the IGCB’s capsule captures new IGCB model, which are formed by restructuring the new set of continuous elements according to the implemented functional block diagram.</td>
<td></td>
</tr>
<tr>
<td>Internal interface</td>
<td>The state machine and ports of this capsule are not changed in the overall design for the new controller of UAV typed VTOL.</td>
<td>The specialization of the internal interface’s capsule is performed by supplementing/cutting down in/from the new IGCB in the IGCB’s capsule if necessary new Inv term that correspond to new supplemented/cut down situations in/from the discrete part’s capsule of application.</td>
<td></td>
</tr>
<tr>
<td>External interface</td>
<td>The state machine, ports and protocols of this capsule are not changed in the overall design for the new controller of UAV typed VTOL.</td>
<td>The external interface’s capsule is specialized by supplementing or cutting down input or output events, which are issued from outside, (i.e. supplementing/cutting down these events in/from its protocols).</td>
<td></td>
</tr>
</tbody>
</table>

The capsule-based collaboration pattern shown in Fig. 7 and Fig. 8 are not changed in the overall design for new controllers of UAVs typed VTOL.

C. PSM for a Q-UAV Controller

The about designed PIM is implemented to the PSM that is transformed from the above built PIM by using tools such as IBM Rational Rose RealTime or IBM Rational Software Architect RealTime [62] or Papyrus for Real-Time (Papyrus-RT) [63], in order deploy the Q-UAV controller. Here, IBM Rational’s leading role in defining the real-time UML is...
widely acknowledged, as is the pre-eminence of the *IBM Rational Rose RealTime* or *IBM Rational Software Architect RealTime* products for supporting the real-time UML design of industrial software systems. It combines a rich modeling environment with a code-oriented tool set to make up a comprehensive practitioner desktop for creating solutions in a variety of architectural styles, and targeted at specific runtime infrastructures. Besides, *Papyrus-RT* is an industrial-grade, complete open-source modeling environment for the development of complex, software intensive, real-time, embedded, cyber-physical systems; it provides an implementation of the real-time UML together with editors, code generator for C++ and a supporting runtime system. Many other important lifecycle artifacts also benefit from these tools (e.g. requirements lists, test cases and build scripts) to entirely cover development phases for the Q-UAV controller.

Hence, the above designed PIM can be subsequently transformed into the PSM (i.e. the realization model) by using various object-oriented Implementation Development Environment (IDE) in order to entirely deploy the Q-UAV controller with compatible embedded microcontrollers. In fact, this transformation model is quickly performed through the round-trip engineering (i.e. the forward and reverse engineering) of intermediate programming codes, e.g. the C++ codes, which are issued from the design models depicted in *IBM Rational Rose RealTime* or *IBM Rational Software Architect RealTime* or *Papyrus-RT* tools.

Furthermore, the above defined HA can be automatically implemented by using the State pattern described in [64, 65]. The State pattern in either of the following cases: i) an object’s behavior depends on its state, and it must change its behavior at run-time depending on that state. ii) operations have large, multipart conditional statements that depend on the object’s state. This state is usually represented by one or more enumerated constants. Often, several operations will contain this same conditional structure. The State pattern puts each branch of the conditional in a separate class. This lets us treat the object’s state as an object in its own right that can vary independently from other objects. Following this pattern, the detailed implementation structure and programming codes of HA could be seen in the author’s thesis [6] to improve the evolution of the Q-UAV controller.

V. APPLICATION

Based on the above proposed hybrid control model, we deployed a trajectory-tracking controller, which permits a Q-UAV to reach and follow the desired trajectories. Fig. 9 shows out the Computer-Aided Design (CAD) model of this Q-UAV for the case study. The main operational parameters of Q-UAV are summarized in Table II.

In this application, the EKF algorithm are used for estimating altitude, position, attitude and velocity control. Thus, a model of space-time discreteness is needed to implement the control evolution of the Q-UAV; following the Q-UAV dynamics in equations (1)-(4), the control evolution of system can be then written by equations (6).

$\begin{align*}
    x_k &= f_{k-1}(x_{k-1}, u_{k-1}) + w_{k-1} \\
    y_k &= h_k(x_k) + v_k
\end{align*}$

(6)

Here, $x$ is the vector of state variables at the $k^{th}$ instant of $x$; $u_k$ and $y_k$ are respectively the inputs and outputs of the system; $h_k$, $w_k$ and $v_k$ are respectively the measurement function, additive process and measurement noise.

Fig. 9. CAD model of Q-UAV for the case study.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>4.50 kg</td>
</tr>
<tr>
<td>Maximum payload</td>
<td>1.55 kg</td>
</tr>
<tr>
<td>Autonomous duration</td>
<td>25 minutes</td>
</tr>
<tr>
<td>Li-Po battery</td>
<td>22.2 V, 20000 mAh</td>
</tr>
<tr>
<td>Maximum take-off speed</td>
<td>5.0 m/s</td>
</tr>
<tr>
<td>Maximum horizontal transferring speed</td>
<td>7.5 m/s</td>
</tr>
<tr>
<td>Maximum altitude</td>
<td>500 m</td>
</tr>
<tr>
<td>Maximum radius of action</td>
<td>4900 m</td>
</tr>
<tr>
<td>Inertia moment on x-axis $I_{xx}$</td>
<td>35.14e-3 kg m²</td>
</tr>
<tr>
<td>Inertia moment on y-axis $I_{yy}$</td>
<td>35.14e-3 kg m²</td>
</tr>
<tr>
<td>Inertia moment on z-axis $I_{zz}$</td>
<td>11.18e-2 kg m²</td>
</tr>
<tr>
<td>Rotor inertia of motor $J_r$</td>
<td>36.24e-5 kg m²</td>
</tr>
</tbody>
</table>

The above defined state-space models were implemented by the EKF algorithm (Algorithm I) for predicting the motion states corresponding to the sensors installed on the Q-UAV, such as the Inertial Measurement Unit (IMU): *MPU6000* with working frequency 100Hz [66], and GPS: *Ublox Neo 6M* with working frequency 10Hz [67]. In Algorithm $I$, $P$ denotes an estimate; $P$ and $Q$ are respectively the post covariance matrices of process and measurement noise, assumed as zero mean stationary white noises with zero cross-correlation; the state is recursively estimated starting from the assumed initial conditions as follows: $\hat{x}_{0|0} = x_0$ and $P_{0|0} = 0_{12\times 12}$.

<table>
<thead>
<tr>
<th>Function EKF algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step EKF predict</strong></td>
</tr>
<tr>
<td><strong>Data</strong>: $\hat{x}_{k-1</td>
</tr>
<tr>
<td><strong>Result</strong>: $\hat{x}_{k</td>
</tr>
<tr>
<td>$F_{k-1} = \frac{\partial f_{k-1}}{\partial x} \Big</td>
</tr>
</tbody>
</table>

Algorithm I

EKF algorithm applying to the Q-UAV controller.
\[ \dot{x}_{k|k-1} = f_{k-1}(\hat{x}_{k-1|k-1}); \]
\[ p_{k|k-1} = F_{k-1} p_{k-1|k-1} F_{k-1}^T + Q_{k-1}; \]
end
Step EKF update
Data : \( \hat{x}_{k|k-1}, P_{k|k-1}, h_k(\cdot) \)
Result : \( \hat{x}_{k|k}, P_{k|k} \)
\[ H_k = \frac{\partial h_k}{\partial x} |_{x_{k|k-1}}; \]
\[ S_k = R_k + H_k P_{k|k-1} H_k^T; \]
\[ L_k = P_{k|k-1} H_k^T S_k^{-1}; \]
\[ e_k = y_k - h_k(\hat{x}_{k|k-1}); \]
\[ \hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k e_k; \]
\[ P_{k|k} = P_{k|k-1} - L_k S_k L_k^T; \]
end

In this case study, ATMEGA32-U2 and STM32 Cortex-M4 microcontrollers [27] have been used on the main board, and installed the control program by the Arduino’s IDE [27] based on C++ language. The installation of trial flights (Fig. 10) and test scenarios were mainly based on the local state machine (Fig. 5b). The trajectory-tracking controller of this Q-UAV was satisfied with performance requirements, e.g. the admissible control duration, transition and static errors.

![Fig. 10. Installation and trial flights for the Q-UAV controller.](image)

We illustrate here some of main test cases and obtained results for controlling the Q-UAV as follows:
- 1st test scenario: the Q-UAV autonomously reaches and follows a desired rectangle-shaped trajectory with heading angles about 90°, the maximum trajectory error is about 1.50m against each Way-Point (WP) position at these heading angles (Fig. 11).
- 2nd test scenario: the Q-UAV autonomously reaches and follows a desired quadrangular-shaped trajectory with heading angles about 120°, 30° and 150°, the maximum trajectory error is about 1.70m against each WP position at these heading angles (Fig. 12).
- 3rd test case: the Q-UAV is capable of Vertical Take Off and Landing (VTOL) with a maximum altitude of 480 m; the landed position error is 0.40m against the initial taken off position (Fig. 13).

Based on the comparison between the above obtained experimental data and the existing experimental results in the second author’s thesis [6] (the thesis has been supervised by the first author) in which the EKF algorithm has not used for implementing the same physical Q-UAV model, the trajectory-tracking controller of this Q-UAV was improved on the stabilized trajectory error, which was diminished 0.4m±0.6m. From the above case study, we also find that the EKF can provide a considerable improvement in the control implementation for UAVs typed VTOL.

![Fig. 11. Q-UAV reaches and follows a desired rectangle-shaped trajectory with heading angles about 90°.](image)

![Fig. 12. Q-UAV reaches and follows a desired quadrangular-shaped trajectory with heading angles about 120°, 30° and 150°.](image)

![Fig. 13. Trial VTOL for the Q-UAV.](image)

VI. CONCLUSIONS AND FUTURE WORK

The paper introduced a hybrid realization model to conveniently develop controllers for micro UAVs such as the Q-UAV. This model is mainly based on the specializations of MDA approach combined with the real-time UML, EKF algorithm and HA for analyzing, designing and deploying closely Q-UAV controllers. No single formalism or language of an engineering process can possible capture all the knowledge and information needed.
to solve complex control systems such as the Q-UAV controller. Hence, the Q-UAV dynamics and control architecture are adapted for the requirements analysis of control that are implemented by the specializations of MDA features including the CIM, PIM and PSM elements. In the CIM, the use case model is clearly defined, and is enclosed with the continuous behaviors, EKF algorithms and HA, in order to gather in detail the implementation requirements for a Q-UAV. The PIM is then converted into the PSM through the round-trip engineering in order to entirely implement the Q-UAV controller with compatible embedded microcontrollers. Following this proposed control model, a trajectory-tracking controller of a low-cost Q-UAV was developed and tested out at ATMEGA U2 and STM32-Cortex-M4 microcontrollers.

Using the above specified MDA approach, engineering developers will be more capable to manage the system complexity through the visual component models and their model transformations, e.g. the model transformations were performed through the round-trip engineering of the intermediate C++ codes in the case study. However, the real-time UML version lack the constructs for modeling internal continuous behaviors for each state on the state machine diagram, an implemented functional block diagram must be then defined in the CIM to model continuous behaviors for the Q-UAV controller with events issued from outside.

Finally, we applied the above proposed model just in the low-cost Q-UAV controller and intend to implement it in the new control applications for autonomous coordinated vehicles. Eventually, if we get positive feedbacks, we will investigate in the application strategy to extend it more effective, in order to create cooperative controllers for balancing search and target response in the coordinated team of a UAV typed VTOL and various autonomous water vehicles for ocean exploration.

REFERENCES


