Hindawi
International Journal of Aerospace Engineering
Volume 2020, Article ID 8862864, 17 pages
https://doi.org/10.1155/2020/8862864

Research Article

An Object-Oriented Systems Engineering Point of View to Develop Controllers of Quadrotor Unmanned Aerial Vehicles

Ngo Van Hien,[1] **Van-Thuan Truong**,[1] **and Ngoc-Tam Bui**[2,3]

[1]*School of Transportation Engineering, Hanoi University of Science and Technology, Hanoi 10000, Vietnam*
[2]*School of Mechanical Engineering, Hanoi University of Science and Technology, Hanoi 10000, Vietnam*
[3]*College of Systems Engineering and Science, Shibaura Institute of Technology, Tokyo 135-8548, Japan*

Correspondence should be addressed to Ngo Van Hien; hien.ngovan@hust.edu.vn,
Van-Thuan Truong; thuan.truongvan@hust.edu.vn, and Ngoc-Tam Bui; tambn@shibaura-it.ac.jp

Received 30 March 2020; Revised 15 June 2020; Accepted 7 July 2020; Published 5 August 2020

Academic Editor: Paolo Castaldi

The aerospace industry needs to be provided with system solutions to technologically challenging and mission-critical problems. Based on the industrial control point of view, development engineers must take costs and existing standards into account in order to effectively design, implement, and deploy control systems with reasonable costs. The customization and reusability are important factors associated with the production of new applications in order to reduce their costs, resources, and development time. In this work, the Model-Driven Architecture (MDA)/Model-Based Systems Engineering (MBSE) approach combined with the real-time Unified Modeling Language (UML)/Systems Modeling Language (SysML), Unscented *Kalman* Filter (UKF) algorithm, and hybrid automata is specialized to obtain a hybrid control model in order to conveniently deploy controllers of Quadrotor Unmanned Aerial Vehicles (Q-UAVs). This hybrid control model also provides a real-time capsule pattern, which allows the designed elements to be customizable and reusable in new applications of various multirotor UAVs of the Vertical Take-Off and Landing (VTOL) type. The Q-UAV dynamics and control architecture are combined with the MDA/MBSE specialization as follows: the Computation Independent Model (CIM) is defined by specifying the use-case model together with the UKF algorithm and hybrid automata to intensively gather the control requirements. The Platform Independent Model (PIM) is then designed by specializing the real-time UML/SysML's features to obtain the main control capsules, ports, and protocols, together with their dynamic evolution. The detailed PIM is subsequently transformed into the PSM by open-source platforms to rapidly implement and deploy the Q-UAV controller. The paper ends with trial flights and deployment results that show good feasibility to be used for a trajectory-tracking controller of a low-cost Q-UAV. In this case study, the Q-UAV controller is implemented with the simulation model in the *OpenModelica* tool. The obtained simulation results then allow the main control elements and their properties to be defined, as well as building the implementation libraries in the *Arduino* environment based on C++ language to quickly perform the realization model in the *ATMEGA32-U2* and *STM32 Cortex-M4* microcontrollers.

1. Introduction

Unmanned Aerial Vehicles (UAVs) have been used in many military applications for a long time. Currently, UAVs, especially quadrotor UAVs (Q-UAVs), are also being developed rapidly in civilian applications to enhance economic effectiveness, e.g., online shipping in e-commerce, precision agriculture, environmental monitoring, mapping, and surveillance. The Q-UAV is one of the micro-UAV types of which operating modes are capable of Vertical Take-Off and Landing (VTOL) and hovering and horizontal flight, so they can handle turbulence due to, for example, wind more easily and are easier to design and realize by using a compact airframe [1–3].

Systems engineering is an approach that has been widely accepted in the aerospace and defense industry to provide system solutions to technologically challenging and mission-critical problems. The solutions often include

the use of hardware and equipment, software, data, people, and facilities [4]. In addition, the low development costs must be taken into account in the construction of applications. Thus, reusability must also be considered in the new development lifecycle of various applications for UAVs, such as the VTOL. According to the Object Management Group (OMG) [5], Unified Modeling Language (UML) has been standardized for analyzing and designing visual system components in the software industry. In addition, System Modeling Language (SysML) [6] is a UML profile for systems engineering that has been also standardized by OMG. SysML is used for analyzing, designing, verifying, and validating the development artifacts of industrial systems in various domains. Nevertheless, both UML and SysML lack the constructs for precise modeling of timing and communication evolutions between the control objects for real-time and embedded systems, e.g., the Q-UAV controller. Furthermore, the OMG has also standardized the Model-Driven Architecture (MDA) [7, 8] whose main idea is to separate the specifications of system operations from the details of the way that a system uses the capabilities of its platform. On the other hand, Model-Based Systems Engineering (MBSE) has been formalized by INCOSE [9, 10] to support the modeling of system requirements, design, analysis, verification, and validation artifacts in the development life cycle of complex systems.

In addition, the problem of designing autonomous flight controllers for Q-UAVs is equally challenging because these controllers are closely connected with the dynamic models [11–13]. Hence, the Q-UAV controller can be considered to be a model that combines discrete and continuous behaviors named the Hybrid Dynamic System (HDS), and it can be modeled by hybrid automata (HA) [14–16].

Following on from the above points, the MDA/MBSE's features can be specialized by combining them with the real-time UML [17–20] together with the SysML (i.e., real-time UML/SysML) to depict the analysis and design components of control systems in detail. This study focuses on the development of a hybrid control model that integrates the Q-UAV dynamics specialized for the MDA/MBSE together with real-time object paradigms and the HA, which permits us to conveniently implement the Q-UAV controller. This model also allows the designed control components to be closely customized and reused for deploying new control applications of various UAV types capable of VTOL. In this work, the Q-UAV dynamics and control architecture are combined with the specialization of MDA/MBSE 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 trajectories. In detail, the CIM consists of the use-case model defined closely with an implemented function block diagram, the supplemented Unscented *Kalman* Filter (UKF) algorithm, and HA to accurately meet the control requirements for a Q-UAV controller. Based on the defined CIM, the PIM is designed by specifying the real-time UML/-SysML that allows a real-time capsule pattern to be created to model the real-time control elements with their timing evolutions in detail. The detailed PIM components are then transformed into PSMs by using open-source platforms such as *OpenModelica* [21] and *Arduino* [22] in order to rapidly implement and deploy the Q-UAV controller. Finally, a trajectory-tracking controller of an application of Q-UAV is deployed and used to test the *ATMEGA U2* and *STM32-Cortex-M4* microcontrollers, which are linked with the Inertial Measurement Unit (IMU) *MPU6000* with a working frequency of 100 Hz [23] and GPS *Ublox Neo 6M* with a working frequency of 10 Hz [24]. This controller is trialed through different experimental scenarios and is compared to existing solutions.

In this study, we followed the method of our author introduced in Van Hien et al. [25]. The main contributions of this paper are summarized as follows:

(1) The MDA/MBSE approach combined with the real-time UML/SysML, UKF algorithm, and hybrid automata is specialized to conveniently analyze, design, implement, and deploy the Q-UAV controller

(2) The main designed control capsules can be customized and reused for various UAV types capable of VTOL

(3) The open-source platforms are used to rapidly implement and deploy the controller

(4) A trajectory-tracking controller of a low-cost Q-UAV is developed and tested

The rest of this paper is organized as follows: The second section introduces the related works that inspired us to build an MDA/MBSE-oriented development life cycle for Q-UAV controllers. The Q-UAV dynamic model and control architecture are introduced in the third section. The fourth section presents the details of MDA/MBSE-based development to conveniently realize and deploy Q-UAV controllers, including the CIM, PIM, and PSM components. In the fifth section, this specialized model is applied to a case study. Conclusions and future work are shown in the final section.

2. Related Work

2.1. Structural Approaches for Q-UAV Controllers. In the present design of complex systems, many applications use standard control methods together with soft-computing approaches to make them more effective for controllers of such systems [26–30]. These have also been applied to the construction of Q-UAV controllers; for example, some of the main control methods and assessments for Q-UAV applications are summarized in Table 1.

The assessment results of the above control approaches suggest the use of a combination of PID and backstepping, the so-called Integral Backstepping (IB) technique. This is used to implement the continuous evolution of the Q-UAV controller by a functional block diagram in the fourth section.

In addition, the above traditional control models are based on structural approaches that focus on database-driven architecture. Hence, the designed control components

Table 1: Some of the used control methods and assessments for Quadrotor Unmanned Aerial Vehicle (Q-UAV) applications.

Used control methods	Assessment of Q-UAV control applications
Lyapunov theory [31, 32]	This is proved to be very reactive, especially for the yaw angle control of a Q-UAV. However, stabilization in the direct neighborhood of the equilibrium point was not rigid enough to permit hover flight.
Proportional–Integral–Derivative (PID) controller [33–35]	It is proved to be well adapted to the Q-UAV when flying near hover. It was possible using this technique to successfully perform the first autonomous flight. But the PID controller was only able to control the Q-UAV in near hover and absence of large disturbances.
Linear Quadratic (LQ) regulator [36–38]	This displayed average stabilization results. However, it was shown to be less dynamic than the PID controller.
Backstepping technique [39–42]	The ability of this technique to control the orientation angles in the presence of relatively high perturbations is very interesting.
Sliding-mode technique [42–49]	This did not provide excellent results when used alone. The switching nature of the controller seems to be ill adapted to the dynamics of Q-UAVs. Thus, it could be combined with other control techniques, such as the backstepping or radical basis function neural networks, to improve the control performance for Q-UAVs.

could be difficult to customize and reuse for realizing new controllers of different UAVs of the VTOL type and for deploying them appropriately into various software and hardware platforms. To achieve this goal, model-driven approaches can be specialized to conveniently perform the whole development lifecycle focused on control systems such as the Q-UAV controller.

2.2. Model-Driven Approaches and Applications. As previously stated, the MDA has been standardized by OMG for system design, development, and implementation. The three main goals of the MDA are portability, interoperability, and reusability through an architectural separation of concerns. Furthermore, Model-Based Systems Engineering (MBSE) is the formalized application of modeling that is used to support system requirements and for design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases [9, 10]. MDA is a system development paradigm that emphasizes the use of rigorous visual modeling techniques throughout the system development life cycle (SDLC), and MBSE is a specialization of MDA that applies MDA principles and best practices to systems engineering applications [25, 50]. Following the above model-based approaches, Ragavan et al. [51] 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 et al. [52] introduced the complex UML/MARTE (Modeling and Analysis of Real-Time and Embedded Systems) methodology to design embedded systems, which brought out a new integration of Model-Driven Engineering (MDE), electronic systems, and space exploration technologies. This platform could capture the set of possible design solutions by using the UML and the standard MARTE profile. Many industrial applications based on the real-time UML/SysML and model-driven approaches for complex control systems can be found in [8, 25, 50, 53–56].

In particular, the demands of system complexity and design productivity for embedded control systems must be managed by simplifying and reusing the design. Thus, these systems need to be verified as early as possible in the development process to reduce the cost and effort in the context of MBSE [57]. To achieve these goals, the UML profile for *SystemVerilog* (UMLSV) [58, 59] and a novel Natural Language Processing (NLP) framework [60] were developed to completely model the design and verification requirements of systems.

In this section, we cited the structural realizations for Q-UAV controllers and the model-based approaches for industrial control systems that could be used to create a hybrid control model based on the MDA/MBSE approach and real-time UML/SysML to conveniently deploy the Q-UAV controller.

3. Q-UAV Dynamics and Control Architecture

3.1. Q-UAV Dynamic Model for Control. The aim of modeling the dynamics of Q-UAV is to find differential equations that relate a system's outputs (position and orientation) to its inputs (force and torque vectors). As determined from a large field of guidance, the navigation and control of aerial vehicles presented in [61–64], the 6 DoF dynamic model of a Q-UAV in the body coordinate frame can be written as equation system (1). Here, I_{xx}, I_{yy}, and I_{zz} are inertia moments; ϕ, θ, and ψ are, respectively, Roll, Pitch, and 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 ($i = \overline{1, 4}$); Ω_r is the overall residual propeller angular speed; A_c is the fuselage area; C is the propulsion group cost factor; ρ is the air density; Q_i presents the drag moment; h and l are, respectively, the vertical distance and horizontal distance of the propeller center to the Center of Gravity (CoG); and x, y, and z define the position in the body coordinate frame.

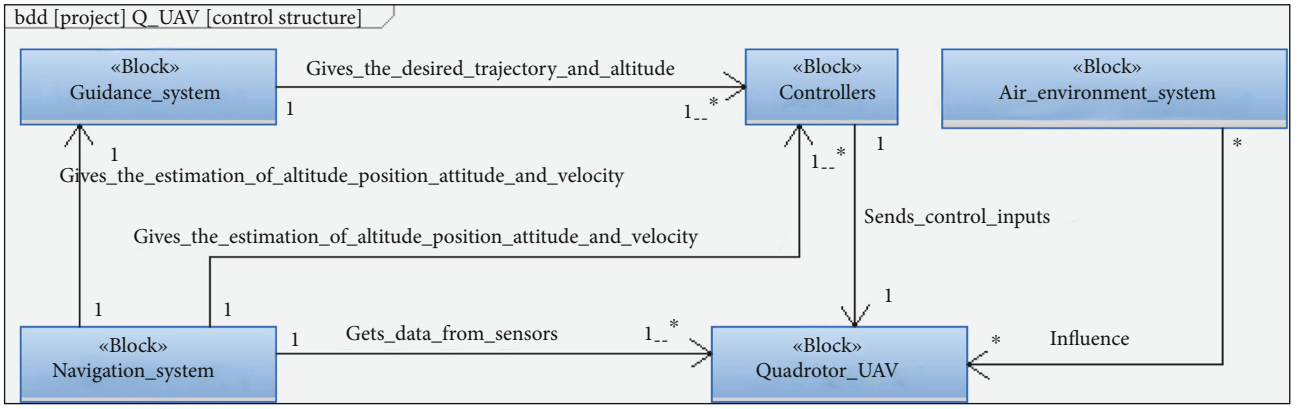

Figure 1: Block definition diagram of autonomy architecture for a Q-UAV.

In this study, we proposed the following assumptions: the hub forces and rolling moments were ignored, and thrust and drag coefficients were considered constants in order to comply with the real-time constraints of the embedded control loop. The evolution of the control system can be rewritten in state-space form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ with the $\mathbf{u}$ input vector and $\mathbf{x}$ state vector as equations (2)–(4).

$$\begin{cases} m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{xi} - \frac{1}{2}C_x A_c \rho \dot{x}|\dot{x}|, \\ m\ddot{y} = (-\cos\psi\cos\phi + \sin\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{yi} - \frac{1}{2}C_y A_c \rho \dot{y}|\dot{y}|, \\ m\ddot{z} = mg - (\cos\psi\cos\phi)\sum_{i=1}^{4} T_i, \\ I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) + J_r\dot{\theta}\Omega_r + l(-T_2 + T_4) - h\left(\sum_{i=1}^{4} H_{yi}\right) + (-1)^{i+1}\sum_{i=1}^{4} R_{mxi}, \\ I_{yy}\ddot{\theta} = \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - J_r\dot{\phi}\Omega_r + l(T_1 - T_3) - h\left(\sum_{i=1}^{4} H_{xi}\right) + (-1)^{i+1}\sum_{i=1}^{4} R_{myi}, \\ I_{zz}\ddot{\psi} = \dot{\theta}\dot{\phi}(I_{xx} - I_{yy}) + J_r\dot{\Omega}_r + (-1)^{i}\sum_{i=1}^{4} Q_i + l(H_{x2} - H_{x4}) + l(H_{y3} - H_{y1}), \end{cases} \tag{1}$$

$$\mathbf{u} = [u_1, u_2, u_3, u_4]^T. \tag{2}$$

Here, u_i is the control input ($i = \overline{1,4}$) that is described by a set of equations (3), ω_i is the propeller angular rate, and b and d are, respectively, the thrust and drag factors.

$$\begin{cases} u_1 = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2), \\ u_2 = b(-\omega_2^2 + \omega_4^2), \\ u_3 = b(-\omega_1^2 + \omega_3^2), \\ u_4 = d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2). \end{cases} \tag{3}$$

$\mathbf{x}$ is a 12-dimensional state vector that is used to describe the motion of Q-UAV and is written as shown in

$$\mathbf{x} = \left[x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}\right]^T. \tag{4}$$

A discrete state-space representation is required to model the Q-UAV controller in order to use a recursive estimation filter of motion states, e.g., the Extended *Kalman* Filter (EKF) or UKF; the developed system can be then described by a set of equations (5):

$$\begin{cases} \mathbf{x_k} = \mathbf{f_{k-1}}(\mathbf{x_{k-1}}, \mathbf{u_{k-1}}) + \mathbf{w_{k-1}}, \\ \mathbf{y_k} = \mathbf{h_k}(\mathbf{x_k}) + \mathbf{v_k}. \end{cases} \tag{5}$$

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

In fact, the EKF has been widely accepted as a standard tool in the control and machine-learning communities. In this study, we used the UKF to estimate the altitude, position, attitude, and velocity control of the Q-UAV. This addresses many of the approximation issues of the EKF and consistently achieves an equal or better level of performance at a comparable level of complexity. The performance benefits of UKF-based algorithms have been demonstrated in a number of application domains, including state estimation, dual estimation, and parameter estimation [65, 66]. There are a number of clear advantages to the UKF, e.g., the mean and covariance of the state estimate are calculated to second-order or better, as opposed to first-order in the EKF. This allows more accurate implementation of the optimal recursive estimation equations, which is the basis for both the EKF and the UKF. While equations specifying the UKF may appear more complicated than the EKF, the actual computational complexity is equivalent. More detail about the use of the UKF algorithm for the Q-UAV controller will be provided in the fourth section.

3.2. General Control Architecture for a Q-UAV. There are three main systems within the physical structure of a Q-UAV as follows: the guidance system is used to provide the desired path for the Q-UAV to track, the navigation system is responsible for estimating the current state of the Q-UAV, and the control system is used to calculate and apply the control forces and moments to conduct this Q-UAV. These subsystems have missions to be accomplished, yet they must also cooperatively operate to allow

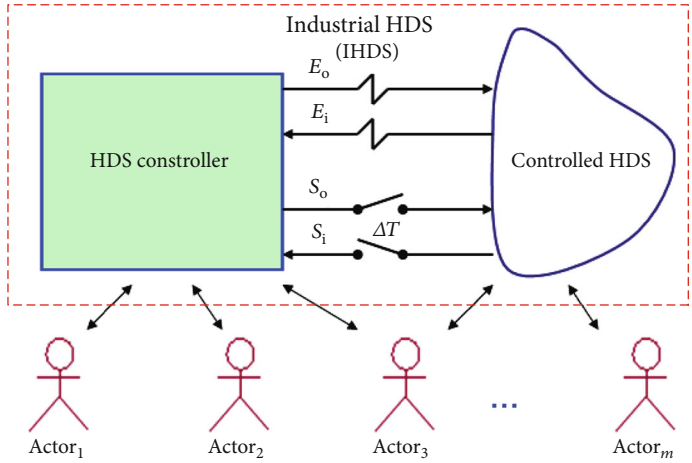

Figure 2: Block diagram of an industrial Hybrid Dynamic System (IHDS).

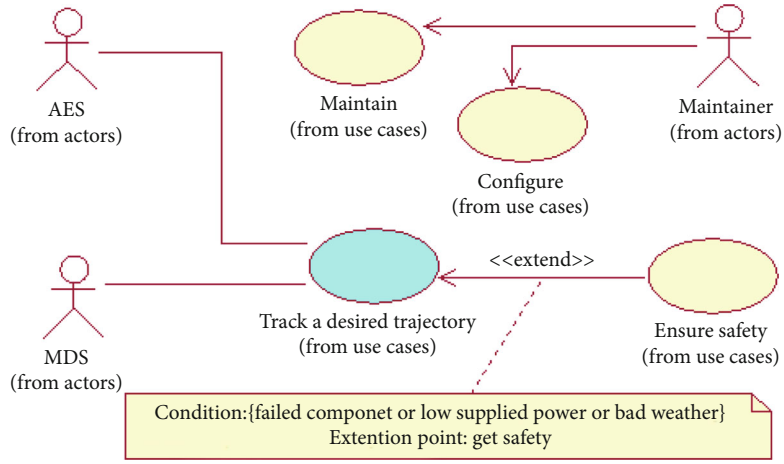

Figure 3: Use case diagram for capturing the main control requirements of the Q-UAV.

the Q-UAV to fulfill its tasks, even in the presence of unknown environmental disturbances. Figure 1 shows a general block definition diagram in SysML, which captures how these subsystems interact.

As previously stated, control systems and their actuators can be considered models including discrete events and continuous behaviors that can be named HDSs. Furthermore, controlled systems do not always have the same behavior because they are associated with validity hypotheses that need to be checked at any moment, as well as security forces to envisage events and behaviors different from nominal behaviors. In this paper, we are also interested in analyzing and designing an industrial HDS (IHDS). This IHDS contains two parts, which are the HDS controller and controlled HDS. These parts mutually exchange periodic signals and episodic events. The episodic event is either external or internal.

Figure 2 shows the block diagram of an IHDS. Here, E_o and E_i are, respectively, output and input events; S_o and S_i are, respectively, output and input signals; ΔT is a sampling period of the evolution model for control; and Actor$_1$, Actor$_2$, and Actor$_m$ are descriptions of a coherent set of roles that users (i.e., persons or involved external systems) play when they interact with the developed IHDS.

As shown by the above Q-UAV dynamic and general control architecture together with the described characteristics of the IHDS, controllers of Q-UAVs are then considered as IHDSs whose dynamic behaviors can be modeled by HA. These control systems have *continuous/discrete* parts and they have interactions, such as those with motional components, e.g., *horizontal transferring*, *VTOL*, and *rotation*, external interacting events from the guidance and navigation system, and environmental disturbances. In this study, we are interested in developing a trajectory-tracking controller of Q-UAVs, so this hybrid dynamic model can be used to form control algorithms combined with a specific guidance law such as the Line-Of-Sight (LOS) guidance implemented in [67–70].

4. Model-Driven Development for a Q-UAV Controller

As previously stated, the CIM, PIM, and PSM components of MDA/MBSE combined with the real-time UML/SysML are

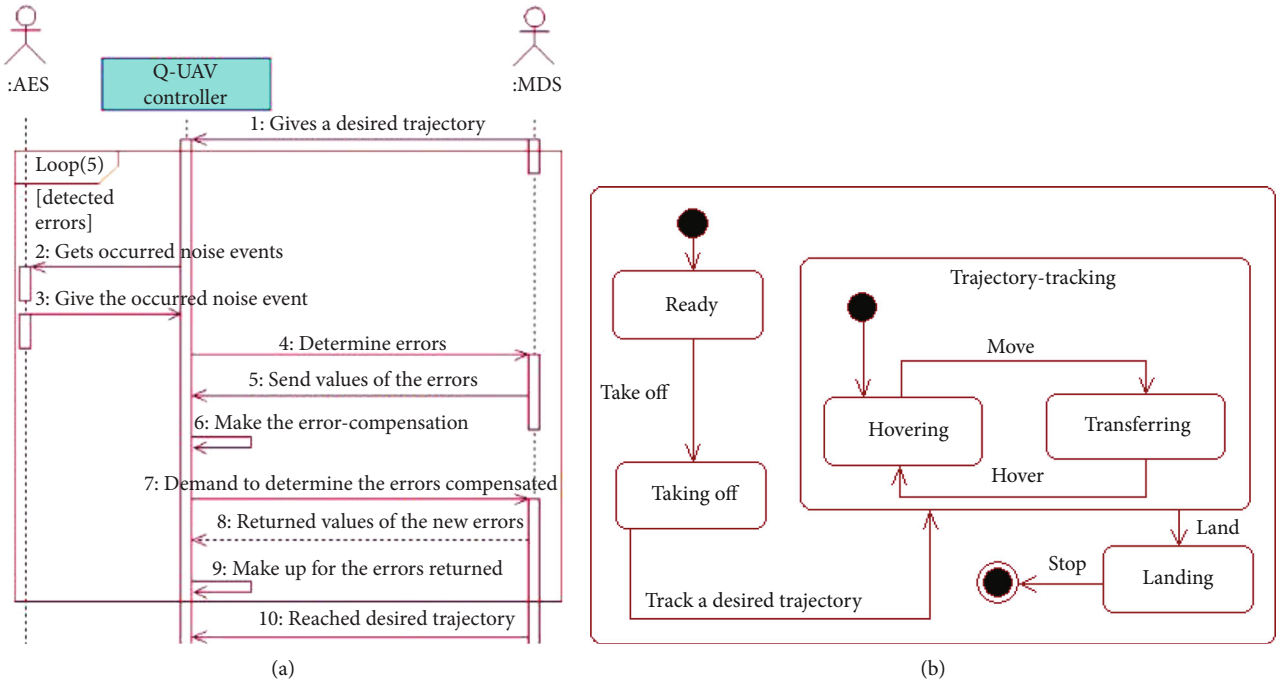

Figure 4: A desired trajectory-tracking scenario (a) and local state machine for performing the "Track a desired trajectory" use case (b).

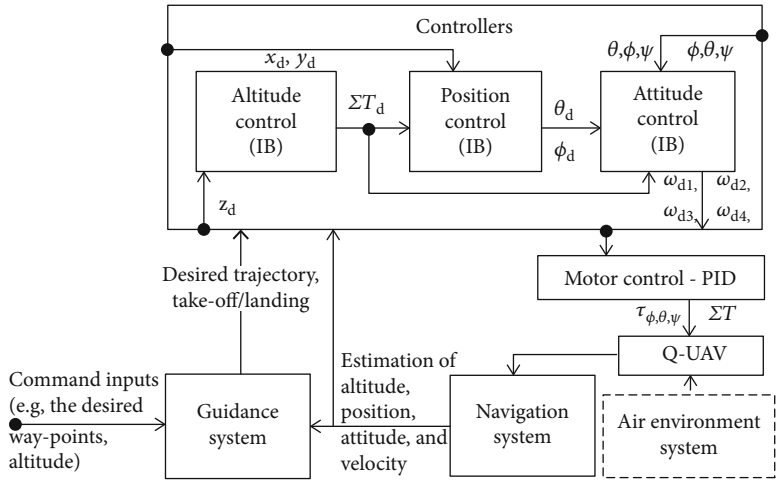

Figure 5: A functional block diagram for implementing the continuous evolutions of the Q-UAV controller.

specialized for analyzing and designing control components of complex control systems, e.g., the Q-UAV controller, in detail. Following the above-described Q-UAV dynamics and general control architecture for the real-time UML/SysML specifications, we specialize the MDA/MBSE approach in detail for deploying the Q-UAV controller that consists of the following three models: CIM, PIM, and PSM. This separates the operation specifications of the system from the details of the way that the system uses the capabilities of its platform. All of the artefacts and activities in CIM, PIM, and PSM for a Q-UAV controller will be detailed in the next subsections.

4.1. CIM for a Q-UAV Controller. In the CIM, object collaborations with the real-time UML/SysML are based on the use case model, together with the specifications of continuous behaviors including the implemented functional block diagram, the IB, and UKF algorithms, and HA in order to closely depict the control requirements for a Q-UAV controller.

The main use case model of a Q-UAV controller is defined as shown in Figure 3. It encompasses an example of trajectory-tracking scenarios and the local state machine of the "*Track a desired trajectory*" use case, which are, respectively, shown in Figures 4(a) and 4(b)). In Figure 4(a), the "*loop(5)*" fragment is a value typically seen in the practice of LOS guidance, as shown in [71],

where

(i) MDS is the *Measurement-cum-Display System*, consisting of the guidance and navigation systems,

Function *UKF algorithm*
Step *UKF predict*
Data: $\hat{\mathbf{x}}_{k-1|k-1}, P_{k-1|k-1}, \mathbf{f}_{k-1}(.)$
Result: $\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}$
$(\hat{\mathbf{x}}_{k|k-1}, \bar{P}_{k|k-1}) = UT(\hat{\mathbf{x}}_{k-1|k-1}, \bar{P}_{k-1|k-1}, \mathbf{f}_{k-1}(.))$;
$P_{k|k-1} = \bar{P}_{k|k-1} + Q_{k-1}$;
End

Step *UKF update*
Data: $\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}, \mathbf{h}_k(.)$
Result: $\hat{\mathbf{x}}_{k|k}, P_{k|k}$
$(\hat{\mathbf{y}}_{k|k-1}, \bar{S}_k, P_k^{xy}) = UT(\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}, \mathbf{h}_{k-1}(.))$;
$S_k = R_k + \bar{S}_k$;
$L_k = P_k^{xy} S_k^{-1}$;
$\mathbf{e}_k = \mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1}$;
$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + L_k \mathbf{e}_k$;
$P_{k|k} = P_{k|k-1} - L_k S_k L_k^T$;
end

Algorithm 1: Standard navigation filter based on the Unscented *Kalman* Filter (UKF) for a Q-UAV controller.

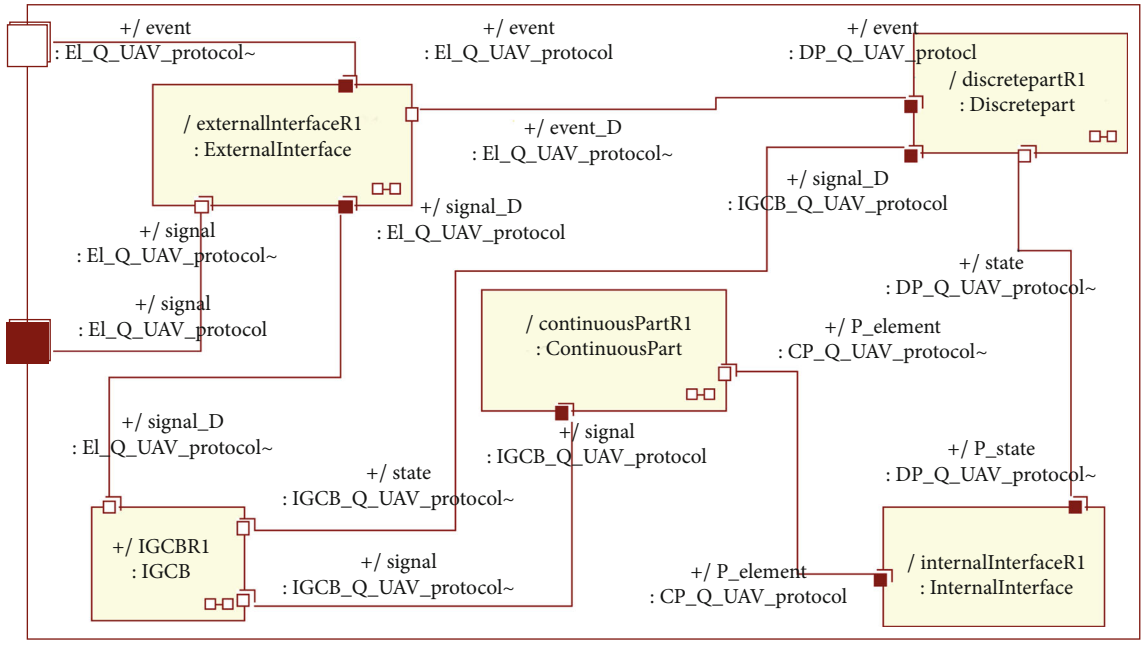

Figure 6: Collaboration diagram of the real-time capsule pattern for the Q-UAV controller.

because both of them essentially act as a signal supplier for the controllers of Q-UAV

(ii) AES is the *Air Environment System* including disturbances generated by the weather

(iii) The "*Track a desired trajectory*" use case is performed for tracking the desired trajectory for the vehicle to follow

(iv) The "*Ensure safety*" use case is used to maintain system safety when one of its components fails or its supplied power is low or there is bad weather

(v) The "*Configure*" use case permits users or maintainers to configure and update control parameters for starting up the controllers

(vi) The "*Maintain*" use case is performed for maintaining the whole system which includes activities, e.g., error identification and correction of the whole physical Q-UAV or periodical maintenance

In the use case model, industrial maneuvering constraints, e.g., the maximum tilted angle, velocity, altitude, and other safety modes of the Q-UAV, need to be set up. In this study, a functional block diagram must be built to model the continuous dynamics of the Q-UAV controller, because the real-time UML/SysML lacks the constructs for depicting internal continuous models for each state situated in 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 Figure 5,

where

(i) *Desired trajectory* and *take-off/landing* are events that are used for providing, respectively, the desired

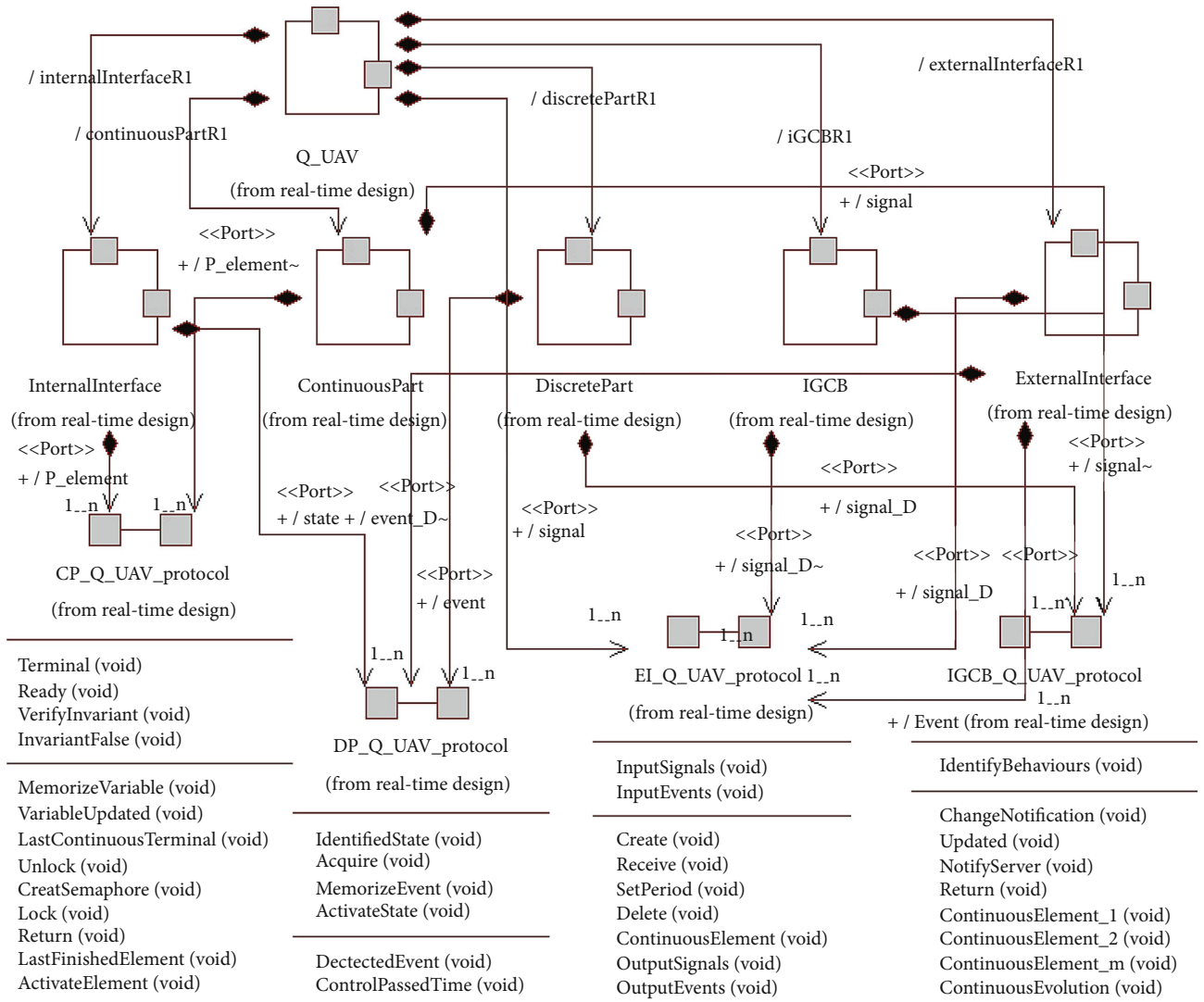

Figure 7: Class diagram of the real-time capsule pattern for the Q-UAV controller.

position and altitude to the blocks of position and altitude control

(ii) ω_{di}, $i = \bar{1,4}$ are desired rotational speeds, which are applied to the four motors

(iii) ΣT and $\tau_{\phi,\theta,\Psi}$ are the overall output forces and moments acting on the Q-UAV

As previously mentioned, the UKF algorithm is used to estimate the altitude, position, attitude, and velocity control. Following the Q-UAV dynamics in equation systems (1)–(5), the application details of the UKF algorithm for a Q-UAV controller are shown in Algorithm 1.

In Algorithm 1, $\hat{\cdot}$ denotes an estimate; P is the state covariance; Q and R are, respectively, the covariance matrices of process and measurement noise, assumed to be zero mean stationary white noise with zero cross-correlation; the state is recursively estimated starting from the assumed initial conditions as $\hat{x}_{0|0} = x_0$ and $P_{0|0} = 0_{12\times12}$; and the Unscented Transform (UT) is a deterministic sampling technique that allows us to compute the mean and the covariance matrix of a random variable. It undergoes a generic nonlinear transformation by propagating a minimum set of its samples and exploiting the knowledge of the mean and the covariance of the starting variable.

In addition, the IB expansion combined with the CLF for Q-UAV controllers is well used in many Q-UAV control applications, for instance [72, 73]. PID regulators can be applied to the functional block of *motor control* to reduce the inertial and delay time caused by the physical Q-UAV actuators in the whole system's evolution.

In the CIM, hybrid automata (HA) are specified to present the mathematical implementation model of the Q-UAV controller consisting of terms such as the *situations*, *continuous state variables*, *event*, *transition*, *global continuous behavior*, and *invariants*. An HA of the Q-UAV controller ($\mathbf{H_{Q-UAV}}$) is established by data as follows:

$$\mathbf{HQ-UAV} = \left(\mathbf{Q}, \mathbf{X}, \sum, \mathbf{A}, \mathbf{Inv}, \mathbf{F}, \mathbf{q_o}, \mathbf{x_{co}}\right). \quad (6)$$

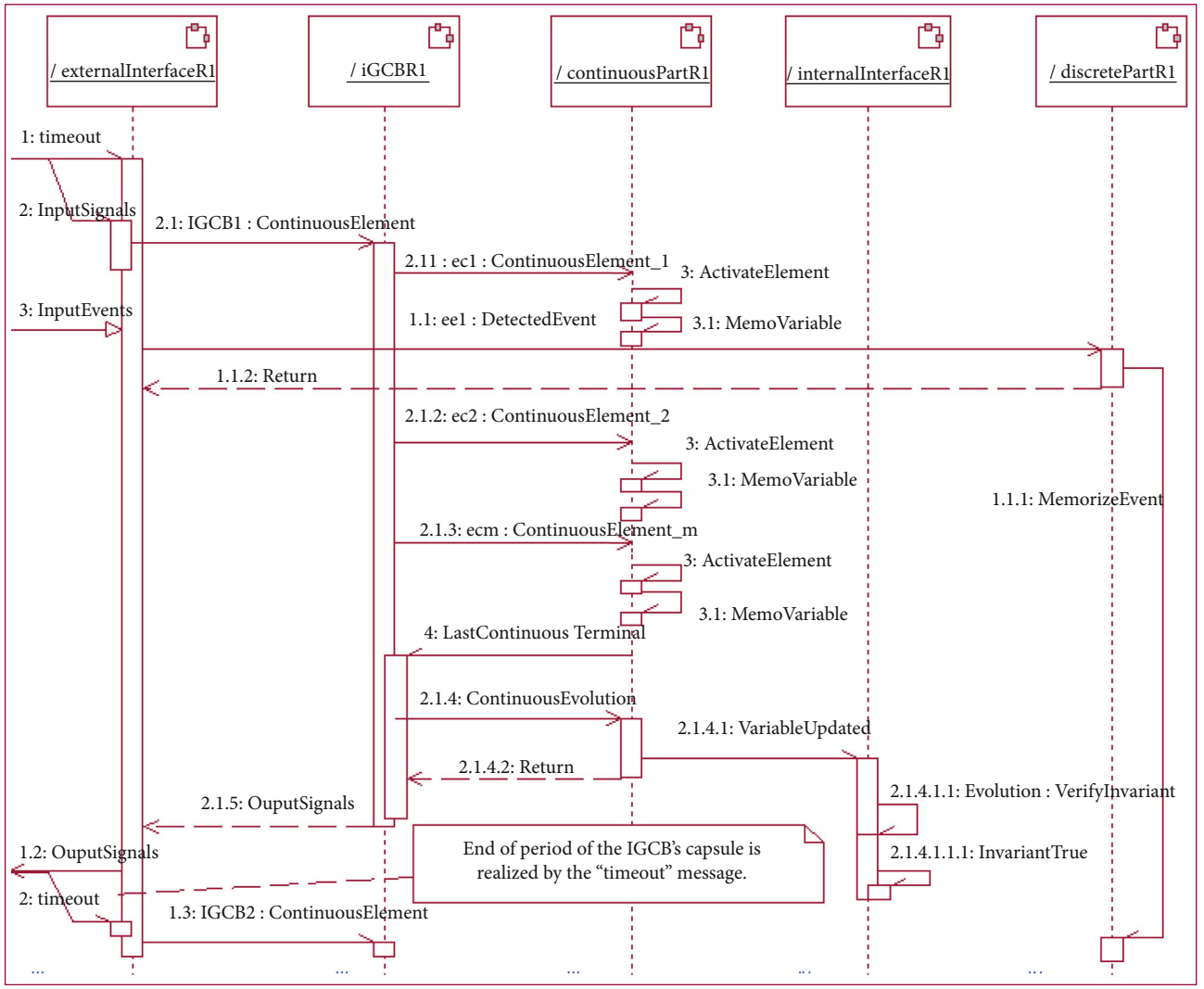

Figure 8: Sequence diagram of the real-time capsule pattern for the Q-UAV controller.

Here, $\mathbf{Q}$ is a set of operation modes called the situations of $\mathbf{H}_{\mathbf{Q-UAV}}$, e.g., the modes of *hovering*, *transferring*, *taking off*, and *landing*; $\mathbf{q_o} \in \mathbf{Q}$ is the initial situation; $\mathbf{X}$ is the continuous state-space of continuous elements of the Q-UAV, e.g., the control blocks of a motor, position, altitude; $\mathbf{x_{co}}$ is the initial value of this space; $\sum$ is a set of external interacting events for triggering movement from the current situation to the reached situation, which is issued from the guidance and navigation systems as well as environmental disturbances for the Q-UAV controller; $\mathbf{A}$ is a set of transitions between the situations that are linked via appropriate events $\sigma \in \sum$; **Inv** is called the *invariant* term of the situation and is used for verifying when the situation is $\mathbf{q}$; the continuous state must be then $\mathbf{x_c} \in \mathbf{inv}(\mathbf{q})$; and $\mathbf{F}$ is the global continuous model issued from the 6 DoF dynamic model in equations (1)–(5) and the implemented functional block diagram (Figure 5). The global evolution of the continuous state is performed when a new situation is reached in the operation modes of the Q-UAV. The details of HA specialization and its hypotheses for the evolution of a control system can be seen in previous reports [74, 75].

4.2. PIM for a Q-UAV Controller. The above-defined CIM is transformed into a PIM that is based on the use case approach [5, 6], real-time UML profile [17–20], and *IBM Rational Rose RealTime* or *IBM Rational Software Architect RealTime* tools [76]. The goal of the PIM is to closely build up the real-time control capsules, ports, protocols, and their timing evolutions, which allow the precise behaviors and structures of the Q-UAV controller to be designed. From the above-defined CIM components, the five main control capsules have been specialized to participate in the HA evolution of the Q-UAV controller: the continuous part's capsule, the discrete part's capsule, the internal interface's capsule, the external interface's capsule, and the Instantaneous Global Continuous Behavior (IGCB's capsule). Figures 6–8 indicate the real-time capsule pattern for the Q-UAV controller by using the real-time UML's collaboration, class, and sequence diagrams.

Table 2: The main control capsules of PIM can be customized and reused in the new control application for various UAVs of the Vertical Take-Off and Landing (VTOL) type.

Designed control capsules	Specialization rules	
	Generic artifacts	*Specialized artifacts*
Discrete capsule	The discrete part's capsule is not changed in the overall design of the new controller of VTOL-type UAVs.	None
Continuous part	The ports and protocols of this capsule are not changed in the overall design of the new controller of VTOL type UAVs.	The continuous part's capsule is specialized by supplementing or cutting down continuous components ($\mathbf{x_c} \in \mathbf{X}$) that depend on the physical actuators installed on the new VTOL-type UAV. 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 is used to redefine the concrete Instantaneous Global Continuous Behaviors (IGCBs) ($\mathbf{f} \in \mathbf{F}$).
IGCB	The state machine, ports, and protocols of this capsule are not changed in the overall design for the new controller of the VTOL-type UAV	The specifications of the IGCB's capsule make up the new IGCB model and are formed by restructuring the new set of continuous elements according to the implemented functional block diagram.
Internal interface	The state machine and ports of this capsule are not changed in the overall design for the new controller of the VTOL-type UAV.	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. A new **Inv** term corresponds to new supplemented/cut down situations in/from the discrete part's capsule of application.
External interface	The state machine, ports, and protocols of this capsule are not changed in the overall design for the new controller of the VTOL-type UAV.	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).

The real-time capsule patterns shown in Figures 6–8 are not changed in the overall design for new controllers of VTOL-type UAVs.

In Figure 6, the *discrete part's capsule* consists of the situations **Q** and transitions **A** in the 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 $\mathbf{f} \in \mathbf{F}$, where **f** is derived from equations (1)–(4) and the implemented functional block diagram (Figure 5) can be implemented in **f** for estimating the values of the 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; and the *external interface's capsule* is an intermediary that receives or sends episodic events and periodic signals between the Q-UAV controller and their interacting systems, such as the AES and MDS.

In Figure 8, the messages exchanged between the main control capsules, which are defined by the protocols (Figure 7), are synchronous, and the interval between two adjacent timeout messages indicates the sampling period (ΔT) of the IGCB's capsule. The external interface's capsule receives periodic 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 this event, named "*ee1: DetectedEvent*", is given 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 *LastContinuousTerminal* message coming from the continuous part's capsule, 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 ($\mathbf{x_c} \in \mathbf{inv}(\mathbf{q})$) 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, which 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*."

Tools such as *HyTech*, *CheckMate*, *PHAver*, and *HSolver* [77] can be used to test a hybrid automaton to verify the generally dynamic behaviors of the Q-UAV controller. At this

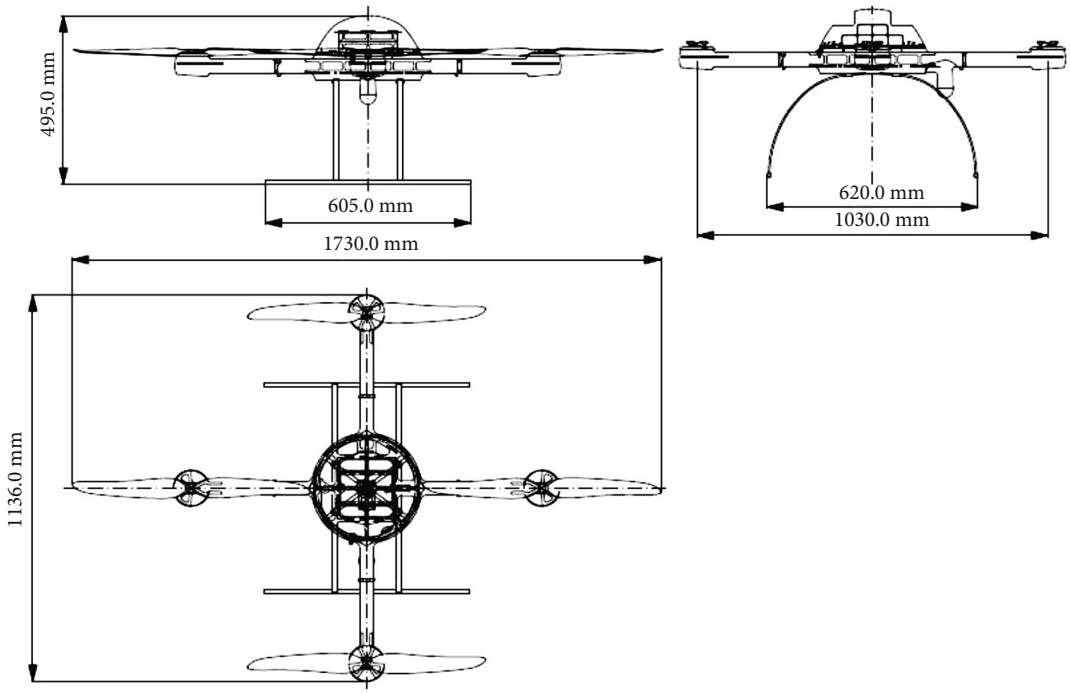

Figure 9: Computer-Aided Design (CAD) model of the Q-UAV used for the case study.

Table 3: Main operational parameters of the Q-UAV.

Parameter	Value
Weight	4.50 kg
Maximum payload	1.55 kg
Autonomous duration	25 minutes
Li-Po battery	22.2 V, 20000 mAh
Maximum take-off speed	5.0 m/s
Maximum horizontal transferring speed	7.5 m/s
Maximum altitude	500 m
Maximum radius of action	4900 m
Inertia moment on x-axis I_{xx}	51.34e-3 kg·m^2
Inertia moment on y-axis I_{yy}	51.34e-3 kg·m^2
Inertia moment on z-axis I_{zz}	11.18e-2 kg·m^2
Rotor inertia of motor J_r	36.24e-5 kg·m^2
Horizontal distance: propeller center to CoG l	0.515 m

moment, formal tools, such as *IBM Rational Test RealTime* [76], also exist to directly test a capsule collaboration. This tool permits us to validate a set of input sequences and defined objects in order to make sure the behaviors of the system are well modeled. Moreover, we can find software tools such as *IBM Rational Rose RealTime* and *IBM Rational Software Architect RealTime* [76], which can be used to support the formal reuse of designed control components. They permit us to undergo reverse engineering of applications to track the requirements of the developed systems. In particular, we can also use the *SystemVerilog* tool [58, 59] and a novel NLP framework [60] to completely validate the control requirements and design models of the developed system.

In addition, the reusability is very important when implementing controllers for different applications of UAVs of the VTOL type because it can reduce the development time and cost. 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 of the VTOL type, are summarized in Table 2.

4.3. PSM for a Q-UAV Controller. To realize and deploy the Q-UAV controller, the PIM is first implemented in the PSM (i.e., the simulation model) that is transformed from the above-built PIM by using tools such as the *IBM Rational Rose RealTime* or *IBM Rational Software Architect RealTime* [76] or *Papyrus for Real-Time* (*Papyrus-RT*) [78] and *Modelica*/*OpenModelica* [21, 79]. Here, the *IBM Rational* provides a rich set of capabilities delivered as part of a collaborative and integrated range of products to support the real-time UML design of industrial software systems. *Papyrus-RT* is an industrial-grade, complete open-source modeling environment that is used for the development of complex, software-intensive, real-time, embedded, and cyber-physical systems. It acts as an implementation of the real-time UML together with editors, a code generator for C++, and a supporting runtime system. In addition, *OpenModelica* is an open-source modeling and simulation environment intended for industrial and academic usage. It is an object-oriented declarative multidomain modeling language that is used for complex dynamic systems. The *OpenModelica* environment allows most of the expression, algorithm, and function parts of *Modelica* to be executed interactively, as well as equation

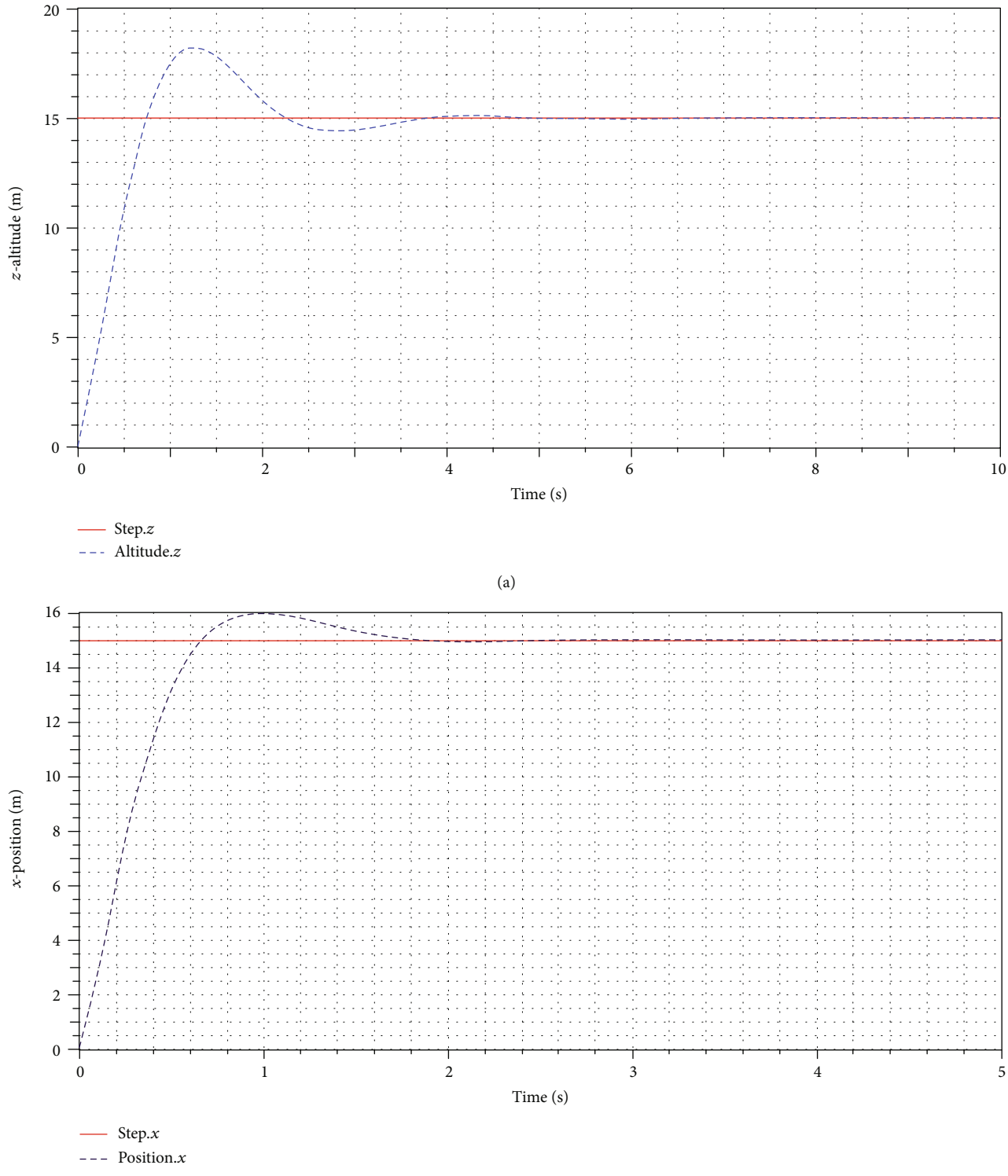

Figure 10: Transient control responses in the z-direction (a) and x-direction (b).

models and *Modelica* functions to be compiled into efficient C/C++ codes. The simulation results in *OpenModelica* allow us to theoretically assess the control performance and to choose the properties of control elements before they are deployed.

In this study, the PIM with optimized control elements in the simulation model was updated to obtain a new PIM for the deployment model of Q-UAV called PIM$^+$. Finally, this PIM$^+$ was transformed into the new PSM$^+$ (i.e., the realization model) by using various specific platforms based on

Figure 11: Installation and trial flights for the Q-UAV controller.

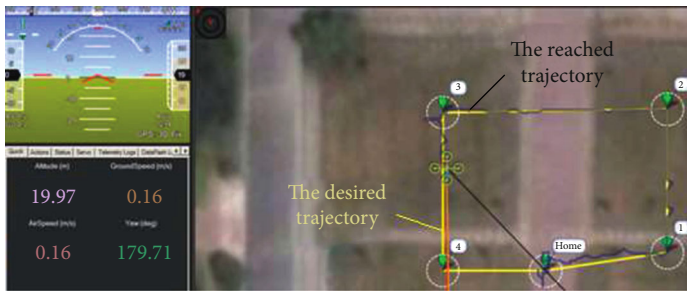

Figure 12: The Q-UAV reached and followed a desired polygon-shaped trajectory with heading angles of about 90° and 110°.

Figure 13: Trial VTOL for the Q-UAV.

the object-oriented Implementation Development Environment (IDE), e.g., the *Arduino*'s IDE [22], to construct a Q-UAV controller with compatible microcontrollers, e.g., the *ATMEGA32-U2* and *STM32 Cortex-M4* microcontrollers [22]. The above PIM-PSM transformations were performed through round-trip engineering (i.e., forward and reverse engineering) of the intermediate C++ codes, including about 80% of the generated codes and 20% of the hand-crafted codes issued from the models depicted in *IBM Rational Rose RealTime* or *IBM Rational Software Architect RealTime* or *Papyrus-RT*, *OpenModelica* tools, and the *Arduino*'s IDE. The transformation rules that were used to convert the PIM into the PSM or the PIM∗ into the PSM∗ and vice versa through the round-trip engineering of the intermediate C++ codes can be found in the authors' reports [50, 74, 75].

In addition, the above-defined HA can be automatically implemented by using the *State pattern* described in [80, 81]. The *State pattern* represents either of the following cases: (i) an object's behavior depends on its state, and the object must change its behavior at run-time depending on that state, and (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 contain this same conditional structure. The State pattern puts each branch of the conditional structure 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 codes of HA have been found in the author's report [74] to enhance the significant programming effectiveness of the control program of Q-UAV.

5. Application

5.1. Physical System Configurations. Based on the above-proposed hybrid model, a trajectory-tracking controller that permits a low-cost Q-UAV to reach and follow the predetermined trajectory was developed. Figure 9 shows the Computer-Aided Design (CAD) model of this Q-UAV used for the case study. The main operational parameters of the Q-UAV are summarized in Table 3.

5.2. Results of Control Implementation. All elements of the analysis and design models were created to allow the implementation of the trajectory-tracking controller of this Q-UAV. Here, we present some of the control simulation results that were obtained through the following simulation cases: the guidance system addressed a desired event of *taking-off* to the Q-UAV with an altitude of 15 m; the transient control response in the z-direction is illustrated in Figure 10(a); Figure 10(b) shows the transient control response in the x-direction when the Q-UAV received the desired event of *transferring* with a desired distance of 15 m in the x-direction from the current position. All of the obtained simulation results allow the control performance of the system to be assessed theoretically within the control criteria such as the admissible timing response, the transition, and static errors. The assessed simulation results permitted us to choose the designed control elements and their properties that were used to perform the deployment phase.

In this case study, the defined state-space models presented in equation systems (4) and (5) were implemented by the UKF algorithm (Algorithm 1) to predict the motion

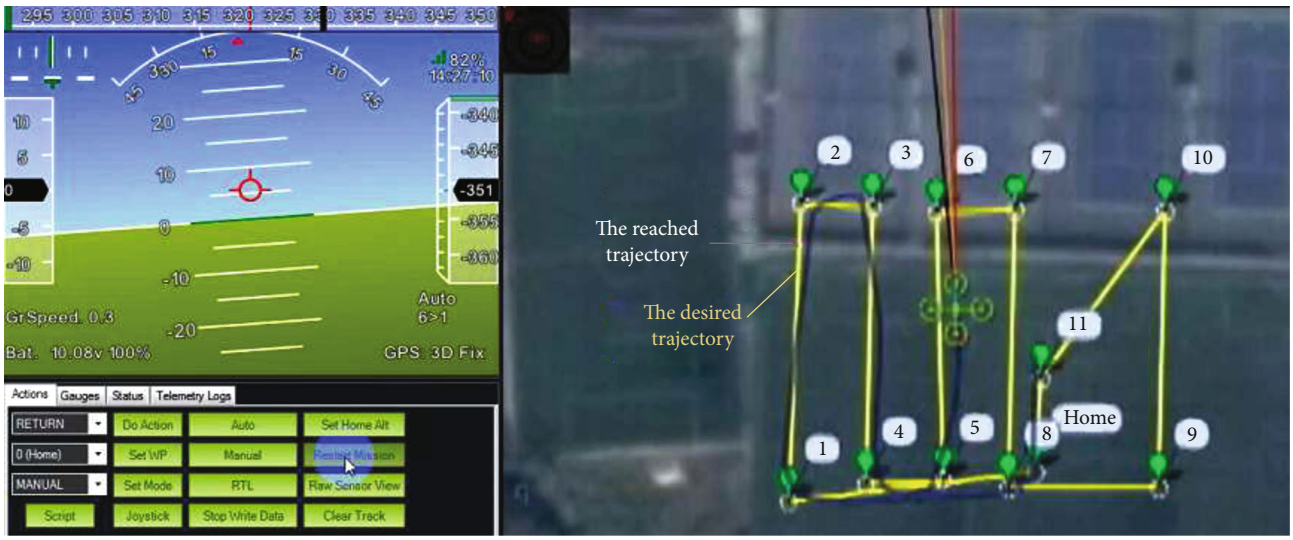

Figure 14: The Q-UAV reached and followed a desired zigzag-lined trajectory.

states corresponding to the sensors installed on this Q-UAV, such as the IMU *MPU6000* with a working frequency of 100 Hz [23] and the GPS *Ublox Neo 6M* with a working frequency of 10 Hz [24]. The *ATMEGA32-U2* and *STM32 Cortex-M4* microcontrollers [22] were used on the main board, and the control program was installed with *Arduino*'s IDE [22] based on C++ language. The Q-UAV installation for trial flights is shown in Figure 11.

The main test cases and obtained results for the Q-UAV are illustrated and were used to assess the control performance of this Q-UAV as follows:

(i) First test scenario: the Q-UAV autonomously reached and followed a desired polygon-shaped trajectory with heading angles of about 90° and 110° and an average velocity of 2.5 m/s (Figure 12). The maximum trajectory error was about 1.40 m against each Way-Point (WP) position at these heading angles.

(ii) Second test case: the Q-UAV was capable of Vertical Take-Off and Landing (VTOL) with a maximum altitude of 480 m (Figure 13); the landed position error was 0.30 m against the initial take-off position

(iii) Third test scenario: the Q-UAV autonomously reached and followed a desired zigzag-lined trajectory with various heading angles and an average velocity of 2.5 m/s (Figure 14). The maximum trajectory error was about 1.20 m against each WP position at these heading angles

Following the comparison between the above-obtained experimental data and the existing experimental results presented in [74], in which the EKF algorithm was used to implement the same physical Q-UAV model, the trajectory-tracking controller of this Q-UAV had an improved stabilized trajectory error, which decreased by about 0.2–0.4 m. From this case, we also found that the UKF can provide considerable improvement over the EKF in the control implementation for VTOL-type UAVs.

6. Conclusions and Future Work

This paper presented a hybrid control that can be used to conveniently realize controllers for Q-UAVs whose global behaviors mean that they can be considered industrial HDSs (IHDSs). This model is mainly based on the specialization of MDA/MBSE features combined with the real-time UML/SysML, UKF algorithm, and HA to closely implement and realize the control parts of the system. The Q-UAV dynamics and control architecture were firstly adapted for the control and combined with the specialization of MDA/MBSE features including the CIM, PIM, and PSM components. In the CIM, the use case model was specialized with continuous behaviors, the UKF algorithm, and HA to closely capture the requirements of the analysis for a Q-UAV controller. The PIM was built to obtain a detailed design model by specifying the control capsules, ports, and protocols enclosed with their timing evolutions in order to model the behaviors and structures of the Q-UAV controllers in detail. The designed PIM components were also used to create a real-time capsule pattern that can be customized and reused in new UAV applications, of which operating modes are capable of VTOL, hovering, and horizontal flight. The updated PIM with the optimized control elements of the simulation model (PIM^+) was then transformed into PSM^+ through the round-trip engineering of the intermediate C++ codes in order to form a Q-UAV controller with compatible microcontrollers. Based on this proposed hybrid control model, a trajectory-tracking controller of a low-cost quadrotor UAV was deployed and tested out on the *Arduino ATMEGA U2* and *STM32-Cortex-M4* microcontrollers.

Furthermore, the above-developed control application confirms that the MDA/MBSE approach combined with the real-time UML/SysML provides a framework for, and enables, tools to be provided to specify a system independently of the

platform that supports it, specify platforms, choose a particular platform for the system, and transform the system's specifications into one for a particular platform. However, the real-time UML/SysML versions lack the constructs required for modeling internal continuous behaviors for each state on the state machine diagram, so an implemented functional block diagram must then be defined in the CIM to model continuous behaviors for industrial HDSs such as the Q-UAV controller with events issued from outside.

We only applied the above-proposed control model in the low-cost Q-UAV controller and intend to implement it in new control applications for autonomous coordinated vehicles. Eventually, if we get positive feedback, we will investigate our application strategy to make it more effective. This will permit us to create controllers to effectively balance the search and target response in cooperative teams of a VTOL-type UAV with an unmanned ship and multiple autonomous underwater vehicles for ocean exploration. This will be specialized with design and verification models of safety-critical systems, as indicated by [58–60].

Abbreviations

CIM: Computation Independent Model
MDE: Model-Driven Engineering
CLF: Control *Lyapunov* Functions
NLP: Natural Language Processing
DoF: Degrees of Freedom
OMG: Object Management Group
GPS: Global Positioning System
PID: Proportional–Integral–Derivative
HA: Hybrid automata
PIM: Platform Independent Model
HDS: Hybrid Dynamic Systems
PSM: Platform Specific Model
IB: Integral Backstepping
Q-UAV: Quadrotor Unmanned Aerial Vehicles
IDE: Implementation Development Environment
SDLC: System Development Life Cycle
IGCB: Instantaneous Global Continuous Behavior
SMC: Sliding-Mode Control
IMU: Inertial Measurement Unit
SysML: System Modeling Language
LQ: Linear Quadratic
UKF: Unscented *Kalman* Filter
LOS: Line-Of-Sight
UML: Unified Modeling Language
MBSE: Model-Based Systems Engineering
UMLSV: UML profile for *SystemVerilog*
MDA: Model-Driven Architecture
WP: Way-Point.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Centennial SIT Action for the 100th anniversary of Shibaura Institute of Technology to enter the top ten Asian Institute of Technology.

References

[1] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, "A compilation of UAV applications for precision agriculture," *Computer Networks*, vol. 172, article 107148, 2020.

[2] R. Woellner and T. C. Wagner, "Saving species, time and money: application of unmanned aerial vehicles (UAVs) for monitoring of an endangered alpine river specialist in a small nature reserve," *Biological Conservation*, vol. 233, pp. 162–175, 2019.

[3] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha et al., "Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[4] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML - the Systems Modeling Language*, Elsevier, Waltham, MA, USA, 3rd edition, 2015.

[5] OMG, "Unified Modeling Language (UML® Version 2.5.1): a specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems," 2017, https://www.omg.org/spec/UML/2.5.1.

[6] OMG, "SysML specifications version 1.6," 2019, https://www.omg.org/spec/SysML/1.6.

[7] OMG, "Model Driven Architecture (MDA): guide revision 2.0 of MDA guide," 2014, https://www.omg.org/mda.

[8] OMG, "The architecture of choice for a changing world: success atories," 2020, http://www.omg.org/mda/products_success.htm.

[9] INCOSE, *Systems Engineering Vision 2025*, INCOSE, San Diego, CA, USA, 2014.

[10] INCOSE, "Object-oriented systems engineering method," 2019, https://www.incose.org/incose-member-resources/working-groups/transformational/object-oriented-se-method.

[11] L. Ding and H. Wu, "Dynamical modelling and robust control for an unmanned aerial robot using hexarotor with 2-DOF manipulator," *International Journal of Aerospace Engineering*, vol. 2019, Article ID 5483073, 12 pages, 2019.

[12] K. Lu, C. Liu, C. Li, and R. Chen, "Flight dynamics modeling and dynamic stability analysis of tilt-rotor aircraft," *International Journal of Aerospace Engineering*, vol. 2019, Article ID 5737212, 15 pages, 2019.

[13] X. Yao and Y. Yang, "Adaptive fault compensation and disturbance suppression design for nonlinear systems with an aircraft control application," *International Journal of Aerospace Engineering*, vol. 2020, Article ID 4531302, 16 pages, 2020.

[14] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?," *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 94–124, 1998.

[15] P. G. Diem, *An object-oriented design method for controllers of quadrotor unmanned aerial vehicles (UAV), [Ph.D. Thesis]*, Hanoi University of Science and Technology, Vietnam, 2017.

[16] N. V. Hien and T. Soriano, "Implementing hybrid automata for developing industrial control systems," in *Proceedings of 8th IEEE-ETFA*, pp. 129–137, Antibes - Juan les Pins, France, 2001.

[17] OMG, "UML profile for MARTE: UML for model-driven development of real time and embedded systems (RTES)," 2019, https://www.omg.org/spec/MARTE.

[18] B. P. Douglass, *Real-Time UML Workshop for Embedded Systems*, Elsevier, Oxford, UK, 2nd edition, 2014.

[19] B. Selic and S. Gerard, *Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE*, Elsevier, USA, 2014.

[20] B. Selic, "Using UML for modeling complex real-time systems," vol. 1474 of Lecture Notes in Computer Science, , Springer, 1998.

[21] OpenModelica, "OpenModelica Software, Version 1.14," January 2020, https://www.openmodelica.org.

[22] Arduino, "Open-source electronics prototyping platform for hardware and software," December 2019, https://www.arduino.cc.

[23] InvenSense, "Sensor system on chip," February 2020, https://invensense.tdk.com.

[24] u-blox, "Global leader in wireless communications and positioning semiconductors and modules for the industrial, automotive and consumer markets," December 2019, https://www.u-blox.com/en.

[25] N. Van Hien, N. Van He, and P. G. Diem, "A model-driven implementation to realize controllers for autonomous underwater vehicles," *Applied Ocean Research*, vol. 78, pp. 307–319, 2018.

[26] Z. Wang, J. Yu, S. Lin, J. Dong, and Z. Yu, "Distributed robust adaptive fault-tolerant mechanism for quadrotor UAV real-time wireless network systems with random delay and packet loss," *IEEE Access*, vol. 7, pp. 134055–134062, 2019.

[27] Z. Fu, J. Yu, G. Xie, Y. Chen, and Y. Mao, "A heuristic evolutionary algorithm of UAV path planning," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 2851964, 11 pages, 2018.

[28] C. Zhu and Z. Guo, "Design of head-pursuit guidance law based on backstepping sliding mode control," *International Journal of Aerospace Engineering*, vol. 2019, Article ID 8214042, 18 pages, 2019, 2019.

[29] Z. Chen and H. Jia, "Design of flight control system for a novel tilt-rotor UAV," *Complexity*, vol. 2020, Article ID 4757381, 14 pages, 2020.

[30] X. Liu and X. Liang, "Integrated guidance and control of interceptor missile based on asymmetric barrier Lyapunov function," *International Journal of Aerospace Engineering*, vol. 2019, Article ID 8531584, 17 pages, 2019.

[31] Q. Xu, Z. Wang, and Z. Zhen, "Information fusion estimation-based path following control of quadrotor UAVs subjected to Gaussian random disturbance," *ISA Transactions*, vol. 99, pp. 84–94, 2020.

[32] Y. Huang, W. Liu, B. Li, Y. Yang, and B. Xiao, "Finite-time formation tracking control with collision avoidance for quadrotor UAVs," *Journal of the Franklin Institute*, vol. 357, no. 7, pp. 4034–4058, 2020.

[33] J. Muliadi and B. Kusumoputro, "Neural network control system of UAV altitude dynamics and its comparison with the PID control system," *Journal of Advanced Transportation*, vol. 2018, Article ID 3823201, 18 pages, 2018.

[34] R. Miranda-Colorado and L. T. Aguilar, "Robust PID control of quadrotors with power reduction analysis," *ISA Transactions*, vol. 98, pp. 47–62, 2020.

[35] A. A. Najm and I. K. Ibraheem, "Nonlinear PID controller design for a 6-DOF UAV quadrotor system," *Engineering Science and Technology, an International Journal*, vol. 22, no. 4, pp. 1087–1097, 2019.

[36] L. Martins, C. Cardeira, and P. Oliveira, "Linear quadratic regulator for trajectory tracking of a quadrotor," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 176–181, 2019.

[37] H. I. Lee, D. W. Yoo, B. Y. Lee et al., "Parameter-robust linear quadratic Gaussian technique for multi-agent slung load transportation," *Aerospace Science and Technology*, vol. 71, pp. 119–127, 2017.

[38] P. Bader, S. Blanes, and E. Ponsoda, "Structure preserving integrators for solving (non-)linear quadratic optimal control problems with applications to describe the flight of a quadrotor," *Journal of Computational and Applied Mathematics*, vol. 262, pp. 223–233, 2014.

[39] Y. Liu, J. Ma, and H. Tu, "Robust command filtered adaptive backstepping control for a quadrotor aircraft," *Journal of Control Science and Engineering*, vol. 2018, Article ID 1854648, 9 pages, 2018.

[40] Z. Chen, K. Niu, and L. Li, "Research on adaptive trajectory tracking algorithm for a quadrotor based on backstepping and the sigma-pi neural network," *International Journal of Aerospace Engineering*, vol. 2019, Article ID 1510341, 9 pages, 2019.

[41] R. Wang and J. Liu, "Trajectory tracking control of a 6-DOF quadrotor UAV with input saturation via backstepping," *Journal of the Franklin Institute*, vol. 355, no. 7, pp. 3288–3309, 2018.

[42] M. Labbadi and M. Cherkaoui, "Robust adaptive backstepping fast terminal sliding mode controller for uncertain quadrotor UAV," *Aerospace Science and Technology*, vol. 93, article 105306, 2019.

[43] M. Labbadi and M. Cherkaoui, "Robust integral terminal sliding mode control for quadrotor UAV with external disturbances," *International Journal of Aerospace Engineering*, vol. 2019, Article ID 2016416, 10 pages, 2019.

[44] T. Kusznir and J. Smoczek, "Sliding Mode-Based Control of a UAV Quadrotor for Suppressing the Cable- Suspended Payload Vibration," *Journal of Control Science and Engineering*, vol. 2020, Article ID 5058039, 12 pages, 2020.

[45] T. Huang, D. Huang, Z. Wang, and A. Shah, "Robust tracking control of a quadrotor UAV based on adaptive sliding mode controller," *Complexity*, vol. 2019, Article ID 7931632, 15 pages, 2019.

[46] M. E. Antonio-Toledo, E. N. Sanchez, A. Y. Alanis, J. A. Flórez, and M. A. Perez-Cisneros, "Real-time integral backstepping with sliding mode control for a quadrotor UAV," *IFAC-PapersOnLine*, vol. 51, no. 13, pp. 549–554, 2018.

[47] M. Labbadi and M. Cherkaoui, "Robust adaptive nonsingular fast terminal sliding-mode tracking control for an uncertain quadrotor UAV subjected to disturbances," *ISA Transactions*, vol. 99, pp. 290–304, 2020.

[48] D. J. Almakhles, "Robust backstepping sliding mode control for a quadrotor trajectory tracking application," *IEEE Access*, vol. 8, pp. 5515–5525, 2020.

[49] S. Li, Y. Wang, J. Tan, and Y. Zheng, "Adaptive RBFNNs/integral sliding mode control for a quadrotor aircraft," *Neurocomputing*, vol. 216, pp. 126–134, 2016.

[50] T. Soriano, N. V. Hien, K. M. Tuan, and T. V. Anh, "An object-unified approach to develop controllers for autonomous underwater vehicles," *Mechatronics - The Science of Intelligent Machines*, vol. 35, pp. 54–70, 2016.

[51] S. K. V. Ragavan, M. Shanmugavel, V. Ganapathy, and B. Shirinzadeh, "Unified meta-modeling framework using bond graph grammars for conceptual modeling," *Robotics and Autonomous Systems*, vol. 72, pp. 114–130, 2015.

[52] F. Herrera, H. Posadas, P. Peñil et al., "The COMPLEX methodology for UML/MARTE modeling and design space exploration of embedded systems," *Journal of Systems Architecture*, vol. 60, no. 1, pp. 55–78, 2014.

[53] L. T. W. Agner, I. W. Soares, P. C. Stadzisz, and J. M. Simão, "A Brazilian survey on UML and model-driven practices for embedded software development," *Journal of Systems and Software*, vol. 86, no. 4, pp. 997–1005, 2013.

[54] M. Rashid, M. W. Anwar, and A. M. Khan, "Toward the tools selection in model based system engineering for embedded systems–A systematic literature review," *Journal of Systems and Software*, vol. 106, pp. 150–163, 2015.

[55] F. Ciccozzi, I. Malavolta, and B. Selic, "Execution of UML models: a systematic review of research and practice," *Software & Systems Modeling*, vol. 18, no. 3, pp. 2313–2360, 2019.

[56] L. O. Freire, L. M. Oliveira, R. T. S. Vale et al., "Development of an AUV control architecture based on systems engineering concepts," *Ocean Engineering*, vol. 151, pp. 157–169, 2018.

[57] M. Rashid, M. W. Anwar, F. Azam, and M. Kashif, "Model-based requirements and properties specifications trends for early design verification of embedded systems," in *Proceedings of 11th System of Systems Engineering Conference (SoSE)*, pp. 1–7, Kongsberg, Norway, 2016.

[58] M. W. Anwar, M. Rashid, F. Azam, and M. Kashif, "Model-based design verification for embedded systems through SVOCL: an OCL extension for SystemVerilog," *Design Automation for Embedded Systems*, vol. 21, no. 1, pp. 1–36, 2017.

[59] M. W. Anwar, M. Rashid, F. Azam, M. Kashif, and B. Butt, "A model-driven framework for design and verification of embedded systems through SystemVerilog," *Design Automation for Embedded Systems*, vol. 23, no. 3-4, pp. 179–223, 2019.

[60] M. W. Anwar, I. Ahsan, F. Azam, H. W. Butt, and M. Rashid, "A natural language processing (NLP) framework for embedded systems to automatically extract verification aspects from textual design requirements," in *Proceedings of 12th International Conference on Computer and Automation Engineering*, pp. 7–12, Sydney, Australia, 2020.

[61] R. C. Nelson, *Flight Stability and Automatic Control*, McGraw Hill, Singapore, 2nd edition, 1998.

[62] K. P. Valavanis and G. J. Vachtsevanos, Eds., *Handbook of Unmanned Aerial Vehicles*, Springer, Dordrecht, Heidelberg, New York, London, 2015.

[63] B. B. Kocer, T. Tjahjowidodo, and G. G. L. Seet, "Model predictive UAV-tool interaction control enhanced by external forces," *Mechatronics*, vol. 58, pp. 47–57, 2019.

[64] J. Huang, H. Zhang, G. Tang, and W. Bao, "Profile-tracking-based adaptive guidance law against maneuvering targets," *International Journal of Aerospace Engineering*, vol. 2019, Article ID 8056342, 17 pages, 2019.

[65] E. A. Wan and R. V. D. Merwe, "The unscented Kalman filter," in *Kalman Filtering and Neural Networks*, S. Haykin, Ed., pp. 221–280, Wiley, New York, USA, 2001.

[66] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation- Theory Algorithms and Software*, Wiley & Sons, USA, 2001.

[67] A. M. Lekkas and T. I. Fossen, "Integral LOS path following for curved paths based on a monotone cubic Hermite spline parametrization," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 6, pp. 2287–2301, 2014.

[68] H. Choset, K. M. Lynch, S. Hutchinson et al., *Principles of Robot Motion: Theory, Algorithms, and Implementation*, the MIT Press, Cambridge, MA, USA; London, UK, 2005.

[69] J. H. Liu, J. Y. Shan, and Q. Liu, "Optimal pulsed guidance law with terminal impact angle constraint," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 231, pp. 1993–2005, 2016.

[70] Z. Zheng and Y. Zou, "Adaptive integral LOS path following for an unmanned airship with uncertainties based on robust RBFNN backstepping," *ISA Transactions*, vol. 65, pp. 210–219, 2016.

[71] L. Béla and M. Lorinc, *Nonlinear Control of Vehicles and Robots*, Springer, UK, 2011.

[72] M. Bouchoucha, S. Seghour, H. Osmani, and M. Bouri, "Integralbackstepping for tracking of a System," *Electronics and Electrical Engineering*, vol. 116, no. 10, pp. 75–80, 2011.

[73] J. J. Xiong and E. H. Zheng, "Position and attitude tracking control for a quadrotor UAV," *ISA Transactions*, vol. 53, no. 3, pp. 725–731, 2014.

[74] N. P. Hung, N. V. Hien, P. G. Diem et al., "Research on, design and manufacture a micro-unmanned aerial flying autonomously at desired trajectories," final report of research project, code: KC03.TN03/11-15, Hanoi University of Science and Technology, Hanoi, Vietnam, 2013.

[75] N. V. Hien, L. T. Thai, V. T. Truong, P. G. Diem et al., "Research, design and manufacture control systems with the integration of object-oriented technology (MDA & Real-Time UML) and navigation units (INS/GPS) for autonomous underwater vehicles," final report of research project, funded by the state, code: KC03.TN05/11-15; Hanoi University of Science and Technology, Hanoi, Vietnam, 2013.

[76] IBM, "IBM rational software and training kit," January 2020, https://my15.digitalexperience.ibm.com/b73a5759-c6a6-4033-ab6b-d9d4f9a6d65b/dxsites/151914d1-03d2-48fe-97d9-d21166848e65/academic/home.

[77] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Sangiovanni-Vincentelli, *Languages and Tools for Hybrid Systems Design*, now Publishers Inc., Boston, MA, USA, 2006.

[78] Papyrus, "Eclipse Papyrus for Real-Time ("Papyrus-RT")," February 2020, https://projects.eclipse.org/projects/polarsys.

[79] P. Fritzson, *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*, Wiley & Sons, USA, 2011.

[80] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Oxford, UK, 1995.

[81] B. P. Douglass, *Design Patterns for Embedded Systems in C - An Embedded Software Engineering Toolkit*, Elsevier, Oxford, UK, 1st edition, 2011.