An Object-Oriented Analysis and Design Model to Implement Controllers for Quadrotor UAVs by Specializing MDA's Features with Hybrid Automata and Real-Time UML

DIEM P.G., HIEN N.V., KHANH N.P.
Department of Aeronautical and Space Engineering, School of Transportation Engineering
Hanoi University of Science and Technology
N$^{\underline{o}}$1, Dai Co Viet, Hai Ba Trung, Hanoi
VIETNAM
{diem.phamgia, hien.ngovan, khanh.nguyenphu}@hust.edu.vn http://www.hust.edu.vn

Abstract: - This paper presents a new approach which is based on the specialization of the Model-Driven Architecture (MDA) with the Real-Time Unified Modeling Language (RT UML) and hybrid automata to effectively analyze, design and implement controllers for quadrotor UAVs (Unmanned Aerial Vehicles). It also allows the designed elements to be customizable and re-usable in the development of new control applications of different quadrotor UAVs. The paper shows out step-by-step the quadrotor UAV dynamic model-to-be used, and the specialization of MDA's features such as the Computation Independent Model (CIM) with use-cases and hybrid automata, the Platform Independent Model (PIM) carried out by using RT UML, and its Platform Specific Model (PSM) implemented by sub-system paradigms and object-oriented mechanisms to entirely perform the development lifecycle of quadrotor UAV controllers. The object transformation rules are also introduced and applied to convert the detailed control design model of PIM into the implementation model of PSM using open-source platforms in order to quickly simulate and realize the control performance and operational functionalities of system. Based on this approach, a trajectory-tracking controller of a mini quadrotor UAV is completely developed and successfully taken on trial flights.

Key-Words: - Quadrotor UAV control, Autonomous flying robots, Object-oriented analysis and design, Hybrid automata, Real-Time UML, MDA.

1 Introduction

The last decade has seen many successfully developed platforms of micro UAVs, especially the quadrotor UAV. Quadrotor UAVs are capable of Vertical Take Off and Landing (VTOL), hovering and horizontal flight, so they are ideal platforms for various civilian and military operations such as intelligence, surveillance, and reconnaissance, target acquisition, traffic monitoring, resource exploration, power line monitoring, forest fire warning, as well as search and rescue [4], [7], [17]. They can be more easily handled in turbulences such as wind and are easier to design using a compact airframe. The research on autonomous control for quadrotor UAVs is very active now. A key characteristic of quadrotor UAVs is that the 6 Degrees of Freedom (DoF) dynamic model in the airframe is controlled by tuning the rotational speed of four motors placed symmetrically around that center at a radial distance.

There are actually many quadrotor UAV control applications, which have used soft computing techniques combined with different control methods to optimally solve the control of quadrotor UAV dynamics. For example, Xun Gong et al. [28] presented a Backstepping Sliding Mode attitude control algorithm incorporating an adaptive estimator for the attitude stability control design of a quadrotor UAV with respect to uncertain parameters and external disturbances. Yu Yali et al. [31] introduced the whole control system of a quadrotor aerial robot, which was divided into three interconnected parts such as the attitude subsystem, vertical subsystem and position subsystem; then nonlinear control strategy of them was applied such as State-Dependent Riccati Equation (SDRE) and Backstepping. An implementation of autonomous visual tracking and landing for a low-cost quadrotor was shown in [30] that adopted computer vision algorithms with classical Proportional-Integral-Derivative (PID) controller. Different controllers based on *Lyapunov* theory, PID, Linear Quadratic (LQ), Backstepping and Sliding-Mode techniques were developed to the control design of a miniature quadrotor UAV, and were compared for attitude control that could be found in [25].

However, we find that these guidance and control models are based on structural procedures and implementations. Thus, they could be difficult to customize and re-use the designed control elements for implementing controllers of different quadrotor UAVs into various software and hardware platforms in order to suitably realize them. Furthermore, modeling, identification, stability analysis, stabilization and control of general industrial systems are challenging and fascinating tasks in modern systems and control theory as well as in academic and industrial applications [23]. In fact, the immersion in an industrial control context makes that the designers and programmers must take into account costs and existing standards for analyzing, designing and implementing effectively these systems. The customization and reutilization are factors to be associated with the production of a new application in order to reduce its costs, resources and time development.

In addition, the Object Management Group (OMG) has standardized the Model-Driven Architecture (MDA) [18], which has been started with the well-known and long established idea to separate the specification of system operations from the details of the way that system uses the capabilities of its platform. MDA provides an approach 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.

Starting from the above considerations, we have developed an object-oriented model, which is mainly based on the specialization of MDA's features with RT UML [5], [14], [20] and Hybrid Automata (HA)) [3], [8] in order to effectively perform an executable process for analyzing, designing, implementing and realizing systematically controllers of the most standard quadrotor UAVs platforms. This paper is depicted by the following main sections:

- Section 2 presents the overview of quadrotor UAV dynamic model and general control structure, which permit us to gather the requirements analysis of a quadrotor UAV controller;
- Section 3 indicates the MDA specialization to obtain an executable object-oriented process model to develop controllers of quadrotor UAVs;
- Section 4 brings out the development models and their model transformations of this process in detail; they include the Computation Independent Model (CIM), the Platform Independent Model (PIM), the Platform Specific Model (PSM), and the object transformation rules to entirely perform the development lifecycle of quadrotor UAV controllers.

Finally, this approach is applied to develop a trajectory-tracking control system, which permits a mini quadrotor UAV to reach and follow a geometric reference path in the *Cartesian* space starting from a given initial configuration.

2 Quadrotor UAV Dynamic Model and Control Structure

2.1 Quadrotor UAV configuration

Present quadrotor 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 $\{\omega_1, \omega_2, \omega_3, \omega_4\}$.

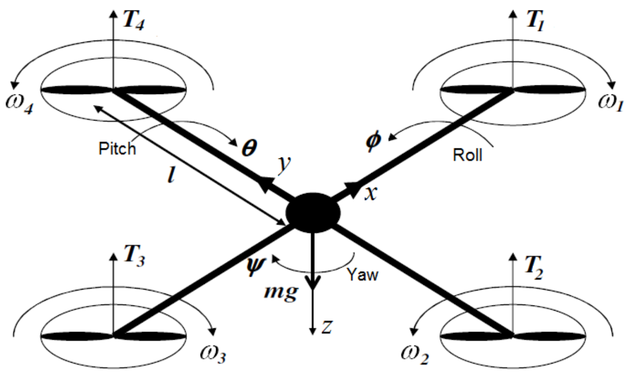

Fig. 1. Quadrotor UAV model.

The following explains the flight principles of a quadrotor UAV. The motors in Fig. 1 are numbered as No.1 at the front motor and then clockwise to No.4. Lift is obtained from the total force by all motors. When the rotors net thrust equals the aircraft's gravity force the quadrotor UAV hovers in the air. Further increasing the net thrust accelerates the aircraft up in the air. The moment around x-axis is generated by the rotational speed difference between No.2 and No.4 motors, so as the attitude angle around x-axis of the airframe changes, the thrust is converted into the component force on y-direction. Using the same principle, by using the rotational speed difference between No.1 and No.3 motors, it is possible to control the x-direction of the airframe. Moreover, the No.1 and No.3 rotors are rotating clockwise while the No.2 and No.4 rotors are rotating in the opposite direction (counter-clockwise). The rotation around z-axis of the airframe is controlled by counterbalancing the moment.

2.2 Modelling Quadrotor UAV Dynamics for Control

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 [4], [7], [15], [17], [25], [29], the 6 DoF dynamic model of quadrotor UAVs in body coordinate frame can be written in the following form:

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

Here: $I_{xx,\ yy,\ zz}$ are inertia moments; ϕ, θ, ψ are respectively *Roll*, *Pitch*, *Yaw* 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, 2, 3, 4); Ω_r is the overall residual propeller angular speed; A_c is 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 vertical distance and horizontal distance: propeller center to Center Of Gravity (CoG); x, y, z define the position in body coordinate frame.

To develop controllers of quadrotor UAVs, it is advisable to simplify the model in order to comply with the real-time constraints of the embedded control loop. In our model, we proposed that hub forces and rolling moments are neglected and thrust and drag coefficients are supposed constant. The system can be rewritten in state-space form $\dot{X} = f(X, U)$ with U inputs vector and X state vector chosen as follows:

$$U = [u_1, u_2, u_3, u_4]^T \quad (2)$$

Here: u_i is the control input of motor; i = 1, 2, 3, 4 (motor number).

$$X = \left[\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}, x, \dot{x}, y, \dot{y}, z, \dot{z}\right]^T \quad (3)$$

The detailed dynamics for control of quadrotor UAVs could be seen in [4], [7], [25].

2.3 Control Structure of Quadrotor UAVs

Various components that make up the autonomy architecture of quadrotor UAVs are the guidance system, navigation system, and control system. Fig. 2 shows out a functional block diagram, which captures how these sub-systems interact. Here, the guidance system is responsible for producing the desired trajectory for the vehicle to follow; the navigation system addresses the task of determining the current state of the quadrotor UAV. The controllers are responsible for providing the corrective signals and events to enable the quadrotor UAV to follow a desired trajectory. This is achieved by receiving the desired state of the quadrotor UAV from the guidance system, and its current state combining the altitude, position, attitude and velocity from the navigation system.

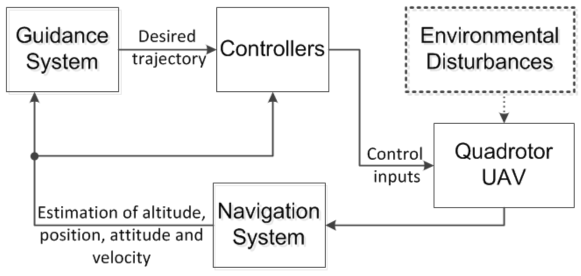

Fig. 2. Block diagram of guidance, navigation and control for quadrotor UAVs.

Environmental disturbances such as winds are extremely complex and highly dynamic, and make the control of a quadrotor UAV highly challenging tasks. These disturbances appear and must be taken into account for a quadrotor UAV in order to traverse such environments.

In addition, control systems of actual machines or actuators generally take account of models with discrete events and continuous behaviors that are called Hybrid Dynamic Systems (HDS) [3], [8], [10]. These behaviors are distributed on different operating modes, which are associated with processes related to the interactivity with users such as the designer, supervisor, maintainer etc. Furthermore, controlled systems do not always have the same behavior because they are associated with validity hypotheses to check at any moment; the security requirement forces to envisage events, and behaviors different from nominal behaviors. The behaviors of such systems are thus complex; their behaviors can be modeled by HA [8], [10] for

modeling completely requirements in the development lifecycle of these systems.
From the described above quadrotor UAV dynamic model (1) or (2) and (3) together with its general control structure and characteristics of HDS, we find that controllers of quadrotor UAVs are HDS whose dynamic behaviors can be modeled by HA. These control system 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 system, and environmental disturbances. In our approach, we are interested in developing the trajectory-tracking controller of quadrotor 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 presented in [2], [27], [29].

3 MDA Process to Develop Controllers of Quadrotor UAVs

3.1 Overview of MDA specification

MDA is an approach to system development, which increases the power of models in that work. 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 uses a variety of resource applications; the reusability builds solutions that can be reused in many applications in different contexts [18].
MDA contains three models (Fig. 3) to separate the specification of the operation of a system from the details of the way that system uses the capabilities of its platform.

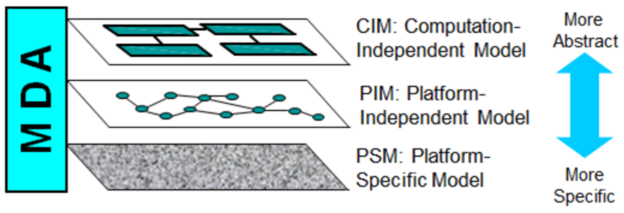
MDA
CIM: Computation-Independent Model
PIM: Platform-Independent Model
PSM: Platform-Specific Model
More Abstract
More Specific

Fig. 3. Models in MDA.

The CIM is referred to as a domain model; it presents the system at the highest level of abstraction. The goal of the CIM is to model the problem entirely in business terms and without getting into the solution or how it might be implemented.
The PIM is used by software architects and designers to describe the software solution at a high level, independent of the solution's deployment platform. This high-level definition of the solution can then be translated into multiple platform-specific models.
The PSM specifies a combination between the details found in the PIM with the details representing how a solution can be implemented on a platform.
Furthermore, MDA also supports for model transformations. The model transformation is the process of converting one model to another model of the same system. The input to the transformation is the marked PIM and the mapping. The result is the PSM and the record of transformation.

3.2 Executable MDA Process for Developing Controllers of Quadrotor UAVs

Starting from MDA specifications and characteristics of the quadrotor UAV dynamic model (1), (2) and (3), we define here an executable process (Fig. 4), which permits us to quickly analyze, design, implement controllers of quadrotor UAVs modeled by HA and to re-use them for new control applications of different quadrotor UAVs.

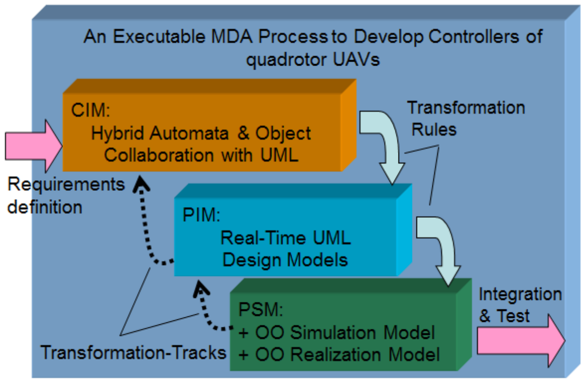
An Executable MDA Process to Develop Controllers of quadrotor UAVs
CIM: Hybrid Automata & Object Collaboration with UML
Transformation Rules
Requirements definition
PIM: Real-Time UML Design Models
PSM: + OO Simulation Model + OO Realization Model
Integration & Test
Transformation-Tracks

Fig. 4. Executable MDA process for developing quadrotor UAV controllers.

This process includes the following main activities and artifacts:
- Object collaborations with UML [19], which are based on the use case model, interaction diagrams and state machines, are used to present the structural and behavioral analysis of a quadrotor UAV. In CIM, HA are used to describe mathematical behaviors [8], i.e. the dynamic model of quadrotor UAV including *Situations*, *Continuous*

State Variables, *Events*, *Transitions*, *Global Continuous Behavior* and *Invariants* of its HA for this system.

- Real-Time UML (RT UML) models indicate the PIM, which permit us to cover the design phase of the developed system. These models are described by using the '*capsules*, *ports*, and *protocols*' concept that we adapted by specializing a set of capsules in precise behaviors.

- Object-Oriented (OO) simulation models firstly are used to introduce the PSM of this system in order to verify and validate the identified control design model into the specific software platforms that permits us to theoretically evaluate the control performance and functionalities, and to easily identify control design elements of this system before we decide to realize and deploy it. Then OO realization models are developed in the PSM in order to carry out its implementation phase with specific platforms, which can support object-oriented programming languages such as C++, Java, Ada, etc., and upload the implemented control program to compatible microcontrollers.

There are transformation rules, which allow the identified CIM to be transformed into a PIM, and to convert the PIM into a PSM. It also contains transformation-tracks, which permit the models to track their transitions. This process will be gone into detail in next sections for describing entirely the development of a quadrotor UAV controller.

4 Implementing MDA Process to Develop Controllers of Quadrotor UAVs

4.1 CIM for a Quadrotor UAV Controller

4.1.1 Capturing the requirements

To capture the general requirements in the object-oriented paradigm, we present here a model, which consists of the abstract classes by using UML stereotypes and class diagram [19] in order to describe the main functional components for quadrotor UAVs (Fig. 5). In UML models, a *stereotype* is a model element that is an extensibility mechanism, which we can use to identify the purpose of the model element to which we apply it. For example, the <<*Guide*>> stereotype can be applied to the abstract *Guidance System*'s class in order to indicate that it is an instruction function for the quadrotor UAV.

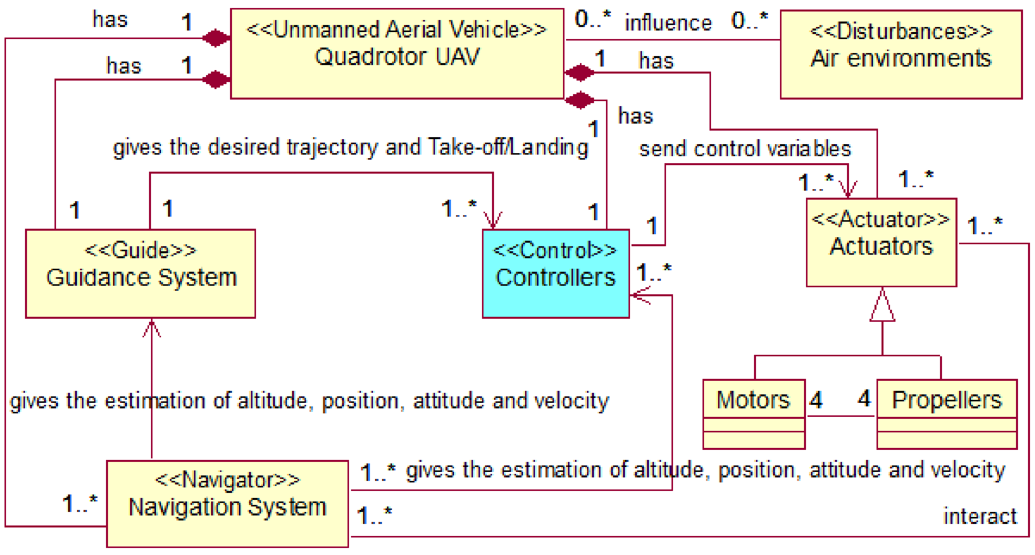

Fig. 5. A UML class diagram for presenting the main functional components for quadrotor UAVs.

The *Guidance System*'s class is responsible for giving the desired trajectory for a quadrotor UAV to follow. This responsibility is completed by taking the desired waypoints defining pre-mission with the possible inclusion of external environmental disturbances issued from the *Air Environment*'s class; then it generates a path for this quadrotor UAV to follow in order to reach each successive waypoint. The *Navigation System*'s class addresses the task of determining the current state (i. e., the combination of the current altitude, position, attitude and velocity) of the quadrotor UAV. It has to provide a best estimation of the current state of this system, regardless of what sensor information is available. The *Controllers*' class is responsible for providing the corrective signals and events to enable the quadrotor UAV to follow a desired trajectory. This is achieved by receiving the desired state from the *Guidance System*'s class, and the current state of the quadrotor UAV from the navigation system's class. This *Controllers*' class then calculates and applies the correcting forces, through use of actuators combining the four motors with their propellers on the quadrotor UAV, to minimize the difference between desired and current states. The *Actuators*' class is used to represent the forces applied to the quadrotor UAV.

4.1.2 Building the CIM for a quadrotor UAV controller

Main steps to construct the CIM for a quadrotor UAV controller are the followings:

i) Identifying complex behaviors of the quadrotor UAV being developed by using the use case model. The use case represents a set of functions or behaviors being provided by the developed system to actors with its stereotype relationships of *include*, *extend* and *generalization* [18], [19]. From the above dynamic model (2) (3), general control structure (Fig. 2) and main functional components (Fig. 5) for quadrotor UAVs, we present the main use case model of controllers as shown in Fig. 6. Here, MDS is the *Measurement and Display System* combined with the guidance and navigation system; AES is the *Air Environment System* including disturbances generated by the weather. In this step, it is necessary to provide industrial control constraints, e.g. the maximum tilted angle, velocity, altitude and other safe flying modes of the developed Quadrotor UAV in order to ensure the operational safety of this system.

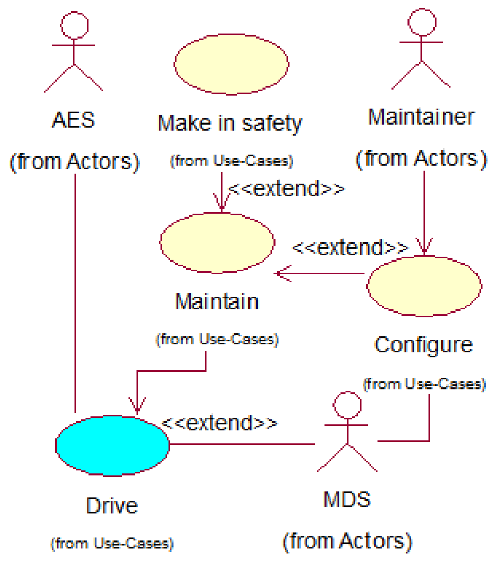

Fig. 6. Main use case model for a quadrotor UAV.

We find that the "*Drive*" use case is oriented towards control modes; its complex behaviors must be specified by using sequence diagrams and local state machine in the RT UML convention [5], [14], [20]. The local state machine of this use case will be used to build HA for control. An example of specified dynamic behaviors is shown in Fig. 7 and Fig. 8 for this use case. In our model, we use the above quadrotor UAV dynamic model (2), (3) and Line-Of-Sight (LOS) guidance [2], [27], [29] because we are interested in the trajectory-tracking control.

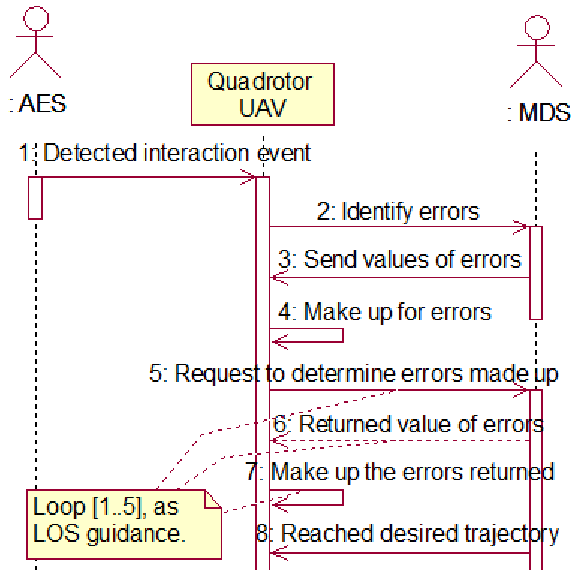
: AES
Quadrotor UAV
: MDS
1: Detected interaction event
2: Identify errors
3: Send values of errors
4: Make up for errors
5: Request to determine errors made up
6: Returned value of errors
7: Make up the errors returned
Loop [1..5], as LOS guidance.
8: Reached desired trajectory

Fig. 7. A scenario of trajectory-tracking control of the "*Drive*" use case.

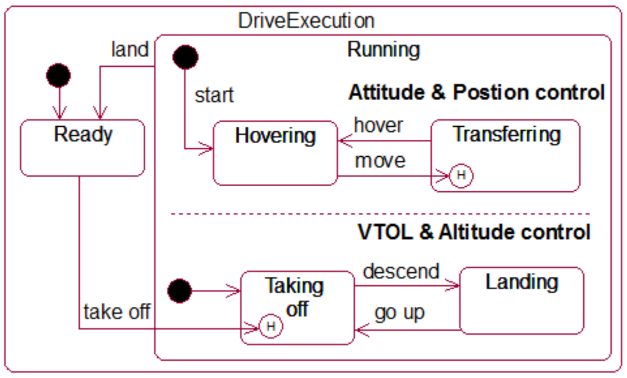

Fig. 8. Local state machine for the "*Drive*" use case.

ii) Defining an implemented functional block diagram, which permits us to model transformational activities of the quadrotor UAV with events coming from outside; because UML lacks the constructs for modeling internal continuous behaviors for each state on the state machine diagram. Starting from the considered dynamic model of quadrotor UAVs, the general control structure, the industrial control constraints, and the identified use case model with LOS guidance, we propose an implemented functional diagram for the quadrotor UAV controller as shown in Fig. 9.

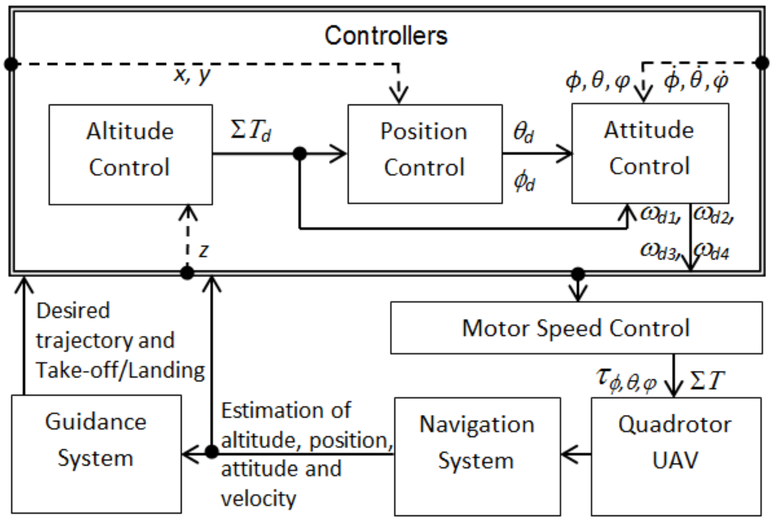

Fig. 9. An implemented functional block diagram for the quadrotor UAV controller

Here, *Desired trajectory* and *Take-off/Landing* actions respectively give the desired position (x_d, y_d) altitude (z_d) to the position and altitude controller; ΣT_d is the desired overall thrust; the position controller receives the quadrotor UAV's position (x, y) and desired thrust, it outputs desired roll (ϕ_d) and pitch (θ_d) while desired yaw (Ψ_d) comes directly from the guidance system; the attitude controller gives then the desired motor speeds (ω_{d1}, ω_{d2}, ω_{d3}, ω_{d4}) to the motor controllers; $\tau_{\phi,\theta,\Psi}$ and ΣT are respectively the overall torque and thrust acting on the quadrotor UAV. In our current model, the Integral Backstepping (IB) technique [2], [25], [28], [31] combined with *Kalman* filtering algorithm [22] are used for attitude, altitude and position control, and PID regulators are applied to the block of motor speed control.

iii) Building a global state machine in order to entirely bring out the global dynamic behavior of the quadrotor UAV being developed from all local identified state machines. The specific rules which permit us to discover this global state machine can be found in [9]. The detailed global state machine, which corresponds to the above use case model and implemented functional diagram, is shown in Fig. 10.

iv) Specifying the HA to carry out the control evolution of the quadrotor UAV controller. This evolution is modelled and implemented by the specialization the HA's formalism and its realization hypotheses introduced in [8], [9]. In addition, HA has only one global continuous behavior at time given, contains the *invariant* notation to verify hypotheses on the continuous state, is derived from an automaton to also model the dynamic behavior of general interactive software systems, and can be verified with proof tools such as *HyTech*, *CheckMate*, *HSolver* [3] and *OpenModelica* [21]. So, it is suitable to use HA to model and implement the control evolution of a quadrotor UAV controller.

A HA of the Quadrotor UAV controller is defined by data of

$$H_{UAV} = (Q, X, \Sigma, A, Inv, F, q_o, x_o) \qquad (4)$$

Where:

- Q is a set of states describing flying modes of H_{UAV}, e.g. the motion in *horizontal translations*, *hovering*, *VTOL*, *rotations*, *roll*, *pitch*, and *yaw*, which are combined with the local state machine oriented towards control modes (Fig. 8) in permutations. Q can be called situations of the Quadrotor UAV controller; q_o is the initial situation.

- X presents the continuous state space of H_{UAV}, $X \subset \Re^n$, x_o is the initial value of this space, e.g. continuous components of the Quadrotor UAV controller.

- Σ is a finite set of events, e.g. external interacting events from the guidance and navigation system, and environmental disturbances.

- A is a set of transitions defined by (q, *Guard*, σ, *Jump*, q') and represented by an arc between situations, here: $q \in Q$, $q' \in Q$; *Guard* is a subset of the state space in which the continuous state must be, so that the transition can be crossed; *Jump* represents the continuous state transformation during the change of situation; it is expressed by a state value function, whose result is affected like initial value of the continuous state in the new situation; $\sigma \in \Sigma$ presents the event being associated in the transition; this association does not imply to give an input or output direction to the event.

- *Inv* is an application, which associates a subset of the state space in each situation; it is called the *invariant* of the situation, in which the continuous state must remain, when the situation is *q*, the continuous state must verify $x \in inv(q)$.

- *F* is defined by using the 6DoF dynamic model of Quadrotor UAV specified from (2), (3), and the implemented functional block diagram (Fig. 9); the evolution of continuous state is occurred when the situation is activated. *F* will be named the *continuous fluid*.

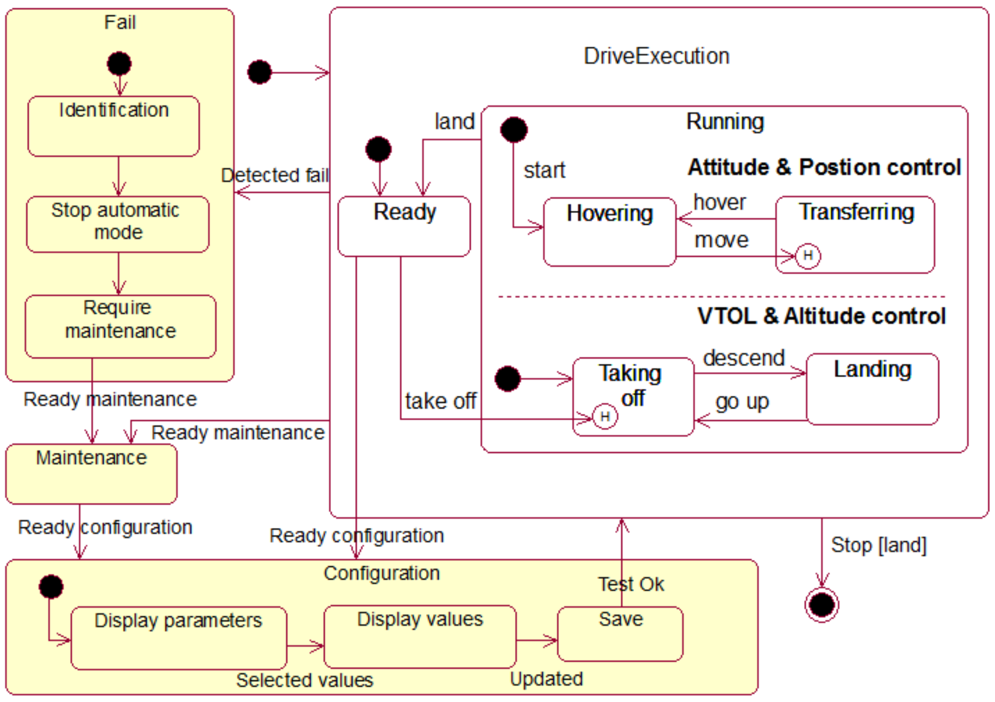

Fig. 10. Global state machine of the quadrotor UAV controller being developed.

4.2 PIM for a Quadrotor UAV Controller

4.2.1 Using RT UML

We find that the direct transformation of CIM to the implementation environment must be supplemented to carry out a quadrotor UAV controller and its re-use in the new application development phase. For example, the above identified CIM are not well adapted to visualize, model interconnection types between control objects or sub-systems. In the detailed design phase of this system, we transform the identified CIM into PIM, which is based on the use case approach [18], and uses the RT UML version [5], [14], [20].

RT UML has its own the graphical notation set to model structures and behaviors of real-time systems. A capsule stereotype is used to represent an active object. A capsule can communicate with other capsules through ports, which are boundary objects, and a protocol associated with the port. RT UML also defines a connector which connects ports to provide transmission facility for supporting a particular protocol. RT UML is more oriented towards the actual implementation and physical design. But RT UML lacks artifacts for modeling system requirement analysis [11]; that's why we launched the requirement modeling process in the above identified CIM for the quadrotor UAV controller in our approach.

Hence, we can use this CIM and the timing modeling convention of RT UML to completely depict the structures and behaviors of complex control systems such as the quadrotor UAV controller.

4.2.2 Constructing the PIM for a quadrotor UAV controller

From the approach introduced in [8], [26], we developed the 5 main control capsules of PIM, which take part in the HA realization of the quadrotor UAV being developed: the continuous part's capsule, discrete part's capsule, internal interface's capsule, external interface's capsule and Instantaneous Global Continuous Behavior (IGCB's capsule). Fig. 11 shows out the communication pattern of these control capsules by using the RT UML's collaboration diagram.

- The discrete part's capsule contains a set of situations *Q* and of transitions *A* of HA of the Quadrotor UAV being developed. This capsule also contains a state machine to make its own evolution with other capsules such as the internal interface's capsule and the IGCB's capsule and to treat the default internal event.

- The continuous part's capsule is related to transformational activities of an Quadrotor UAV controller. The sequential evolution of continuous elements is carried out by specifying the synchronization pattern described in [5] with two sub-capsules called *RendezVous* and *Semaphore*. The continuous part's capsule also has a state machine to make its own evolution with other capsules such as the IGCB's capsule and the internal interface's capsule.

- The IGCB's capsule contains concrete continuous fluids of the developed control system at time given just as *F* in its HA. Each fluid corresponds to a situation in this HA. The IGCB's capsule has a state machine to make its own evolution with other capsules such as the discrete part's capsule and the external interface's capsule. In this evolution, the IGCB's capsule exchanges periodic signals with other capsules such as the discrete part's capsule, continuous part's capsule and external interface's capsule.

- The internal interface's capsule can generate internal events of a control system so that the discrete part's capsule can make its own evolution by these events. It has a state machine to make its own evolution with other capsules such as the continuous part's capsule and the discrete part's capsule.

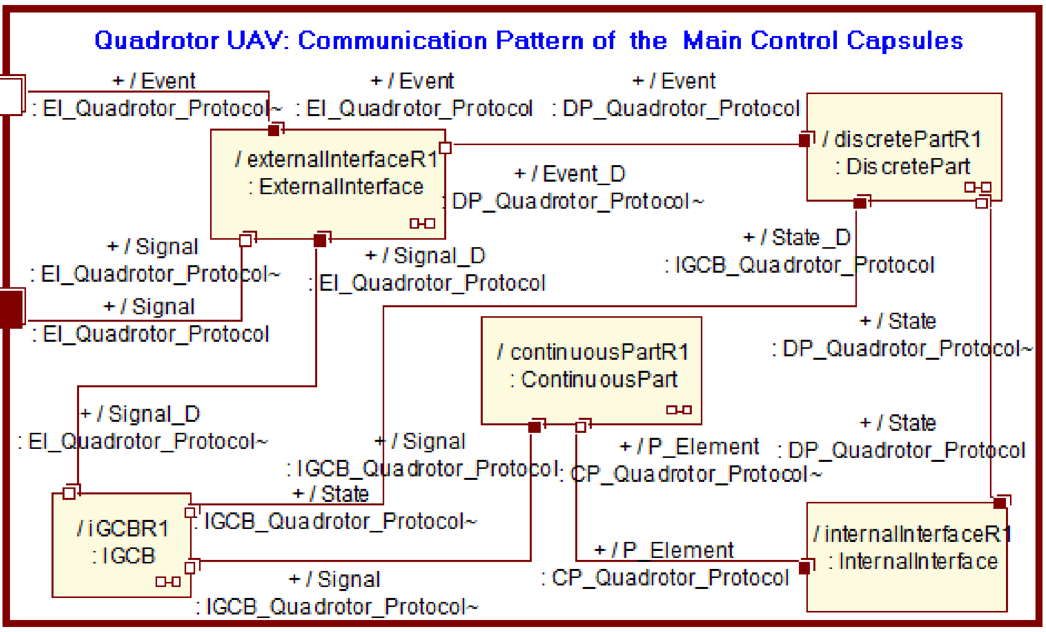

Fig. 11. Communication pattern of main control capsules for Quadrotor UAV controllers.

- The external interface's capsule is an intermediary, which receives or sends episodic events and periodic signals between the developed system and their interacted systems. The external interface's capsule has a state machine to make its own evolution with other capsules such as the discrete part's capsule and the IGCB's capsule.

In addition, the re-use is very important for developing the industrial control system; because it makes it possible to reduce the time and development cost. We find different re-use view in the development phase of this system as follows:

- The re-use view based on the virtual mechanism of objects, classes, or class hierarch;

- The re-use view based on design components. For example, the generic state machine of main control capsule, industrial operational constraints can be specialized to develop different control applications of Quadrotor UAVs.

The specialization, which makes it possible to re-use elements of the capsule collaboration of a general industrial control system, can be seen in [9], [10], [12]. The validation and verification of this collaboration and its traceability with the above identified use case model have been corrected by using the software tool of *IBM Rational Rose RealTime* [13].

4.3 PIM for a Quadrotor UAV Controller

4.3.1 Model transformation

MDA's features supports also for model transformation. The input to the transformation is the marked PIM and the mapping; then the result will be the PSM and the record of transformation. Transformations can use different mixtures of manual and automatic transformation. Fig. 12 shows out the general model transformation by types. A model is prepared using platform independent types specified in a model. The types may be part of software framework. The elements in the PIM are subtypes of the platform independent types. A particular platform is chosen. A specification of a transformation for this platform is available or is prepared. This transformation specification is in terms of a mapping between the platform independent types and the platform dependent types. The elements in the PSM are subtypes of the platform specific types [18].

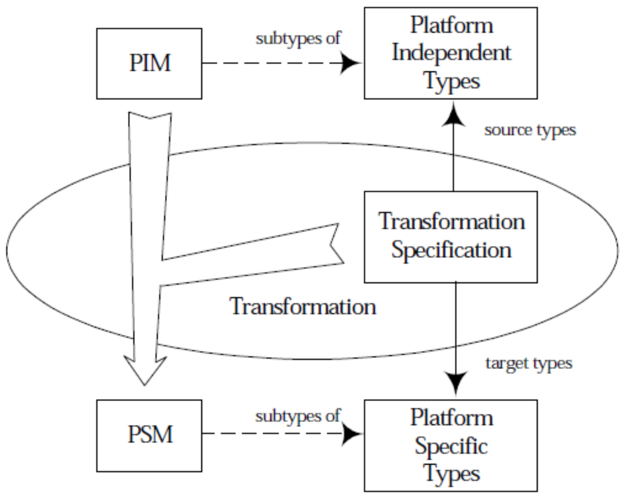

Fig 12. Model transformation by types.

To carry out control systems such as the quadrotor UAV, the PSM is firstly implemented to the simulation model transformed from the above identified PIM. It is important to perform simulation models instead of carrying out experiments on real systems because of expensive and dangerous experiments, investigated systems doing not yet exist, incompatible time scale of the dynamics of the system with the experimenter, inaccessible variables, etc. [24]. The simulation results also permits us to evaluate theoretically the control performance and functionalities, and to easily optimize control design elements of this system before we decide to realize and deploy it. Then, the PIM with the modifying control elements optimized in the PSM of simulation model is adapted to obtain the new updated PIM for realization models of quadrotor UAV that is called PIM*. Finally, this PIM* is converted into new PSMs by using different specific platforms, which are based on the object-oriented Implementation Development Environment (IDE) in order to realize completely the quadrotor UAV controller with compatible microcontrollers. Fig. 13 brings out a sketch of this model transformation.

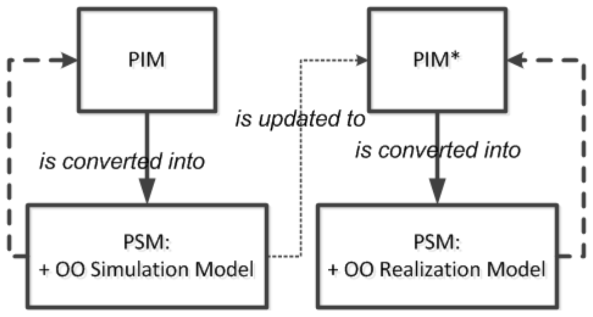

Fig. 13. A sketch of the model transformation for controllers of quadrotor UAVs.

4.3.2 Implementing the PSM for quadrotor UAV simulation model

The '*sub-system*' paradigms, which are supported by software tools such as *LabView-VI*, *MatLab-Simulink*, *OpenModelica*, etc. are used to perform the control simulation model of quadrotor UAVs; because they are easily adapted from the object-oriented design elements of PIM. In this study, we use *OpenModelica* [21] software tool to simulate the control performance of quadrotor UAVs, because it is tightly based on object-oriented mechanisms and properties of *Modelica* language such as the abstraction, encapsulation, modularity and heritance [24].

In addition, *Modelica* is primarily used to quickly solve the continuous and discrete time dynamics of complex systems based on solving differential and algebraic equations. So we applied the following rules to convert the defined elements of PIM into PSM with OpenModelica models in order to completely simulate the controllers of quadrotor UAVs:

- Each capsule is implemented by a class or a block model;

- Each sub-capsule is carried out by a component class or block model; the super-capsule corresponds to the composite class or block model;

- Messages are implemented by the "*functions*" of classes or block models;

- Interfaces are realized by the set of inputs and outputs of a block model;

- Passive classes such as continuous elements or Instantaneous Global Continuous Behaviors (IGCB) are mapped to the "*expressions*" terms;

- State machines of the main capsules are implemented by state graphs.

4.3.3 Performing the PSM for quadrotor UAV realization model

In the PSM with realization models, we have to firstly update the PIM with the modifying control elements optimized in the previous PSM of simulation model, for example, the PID law and its parameters in our case study. Then to carry out quadrotor UAV with microcontrollers, we convert this updated PIM into PSMs by using different specific platforms, which support object-oriented programming languages such as C++, Java, Ada, etc. in order to completely realize its design model. This conversion of updated PIM into PSMs can be carried out by using object-oriented modeling software tools, which support the round-trip engineering such as *IBM Rational Rose RealTime*, *Telelogic Rhapsody* [13]. That makes us to entirely obtain a generated skeleton control implementation model, which consists of the main capsules, sub-capsules, ports, protocols and connectors in their defined interactions.

In addition, the HA of quadrotor UAVs can be automatically implemented in the object-oriented convention by using the "*state pattern*" described in [6]. This pattern allows an object to alter its behavior when its internal state changes; the object will appear to change its class. Fig. 14 shows out the implementation structure of this pattern, which is specified to carry out the HA of quadrotor UAVs.

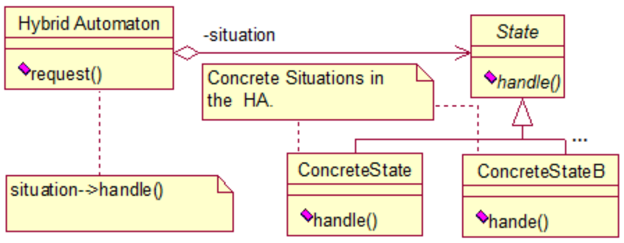

Fig. 14. Implementation structure for the HA of quadrotor UAVs.

Furthermore, the implementation pattern of traffic packet based intrusion detection [16] can be employed to increase detection performances for the *invariant* (*Inv*) in the HA of quadrotor UAVs.

5. An Application

Following the above described approach, we completely developed a trajectory-tracking controller of an autonomous mini quadrotor UAV, which must reach and follow a geometric reference path in the *Cartesian* space starting from a given initial configuration. Some of its characteristics are resumed in Table 1.

Table 1. Characteristics of the developed quadrotor UAV

Parameter	Value
Distance from propeller center to CoG	550 mm
Weight	8000 grams
Payload	4500 grams
Autonomy	20 minutes
Power Li-Po battery	22.2 V, 20000 mAh
Maximum motor speed	10000 rpm
Maximum Take-off speed	3 m/s
Maximum horizontal translation speed	5 m/s
Maximum altitude	500 m
Maximum radius of action	4900 m

We present here some of control simulation results performed by *OpenModelica* software tool that supposed this quadrotor UAV receiving a driving event of *Taking-off* with a desired altitude of 1m of the guidance system; the transient control response in z-direction is shown as Fig. 15.

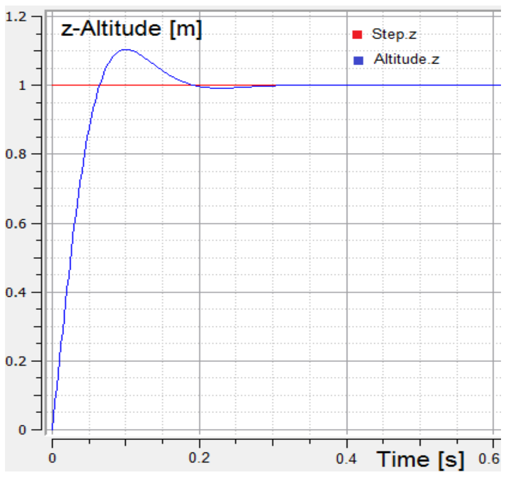

Fig. 15. Transient control response in z-direction

Fig 16 brings out the transient control response in y-direction, when the quadrotor UAV received a driving event of *Transferring* with a desired distance of 1m in y-direction from the current position.

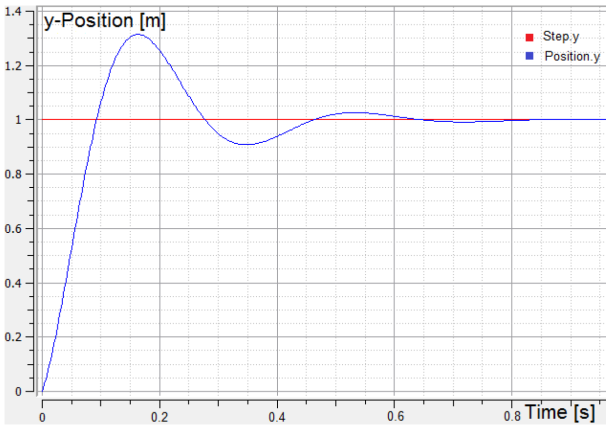

Fig. 16. Transient control response in y-direction

All of obtained simulation results permit us to theoretically evaluate the control performance of this system within the control criteria such as the admissible timing response, transition and static errors. From that point, we can decide to choose the designed control elements in the realization phase of this system.
We have used then *Arduino* platform [1] to quickly deploy the realization model of the controller. Because *Arduino* is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software; it intended for designers and programmers interested in creating interactive objects or environments. *Arduino* can sense the environment by receiving input from a variety of sensors such as pressure, magnetometer, Inertial Measurement Unit (IMU), Global Positioning System (GPS), etc., and can affect its surroundings by controlled actuators. *Arduino Mega 2560* microcontroller [1] has been used on the board, and can be programmed by using the Arduino programming language based on C++ and the object-oriented embedded programming C++. *Arduino* projects can be stand-alone or they can communicate with software running on a computer. *Arduino* IDE has an easy way to include libraries in our generated skeleton control implementation model. There not only are the *header* files included in this sketch, but the implementation files are also compiled behind the scenes. Hence, we can develop a more complex quadrotor UAV project that will use a specialized library of our own. That library will itself build on other libraries. In our realization model, behaviors of each continuous element or IGCB will be implemented as such library.
All of artifacts of the analysis, design and implementation model have been created by using the above presented approach to completely implement the trajectory-tracking controller for this mini quadrotor UAV. We have also performed trial flights to test the realization model of this application (Fig. 17). The scenarios of these tests are based on the use case model and global state machine. Results of trial flight tests are satisfied with the predetermined trajectory and control performance within control criteria such as the admissible control duration, transition and static errors. The detailed experimental scenarios are currently performed to improve the performances and features of this application in the aeronautic laboratory.

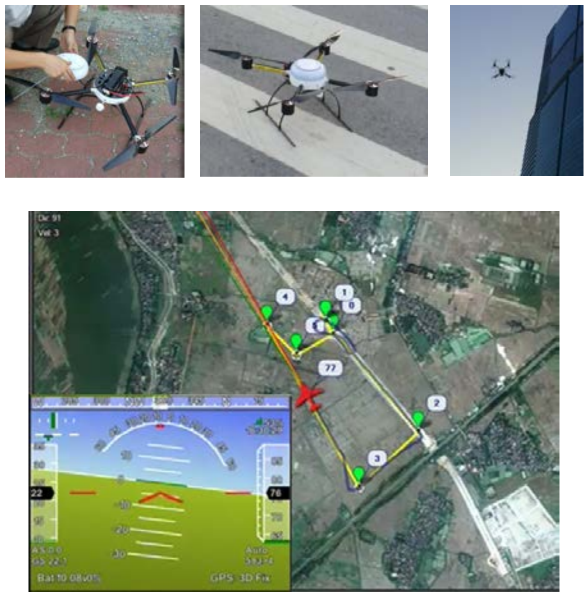

Fig. 17. Set-up and test the trajectory-tracking controller for the autonomous mini quadrotor UAV.

6 Discussion and Closure
In this paper, we have introduced an object-oriented approach to develop controllers of quadrotor UAVs. This approach is based on the specialization of MDA's features with RT UML, HA and microcontrollers in order to quickly analyze, design, implement and realize the control parts of system. 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 quadrotor UAV controller. The quadrotor UAV dynamic model and control structure are adapted to gather the requirement

analysis for controllers of quadrotor UAVs. To model industrial control systems such as the quadrotor UAV controller, we have used HA because there is only one global continuous behavior at time given in a hybrid automaton; there is the invariant notation to verify hypotheses on the continuous state; and the hybrid automaton is derived from an automaton, which models also the dynamic behaviors of general interactive software systems. So we consider that behaviors of the quadrotor UAV controller can be modeled by HA. The MDA's features are specified to obtain a general MDA process model including the CIM, PIM and PSM to entirely develop this system. The CIM of a quadrotor UAV controller is defined to carry out its object-oriented analysis phase by specializing use case model and hybrid automata. The PIM is specified for obtaining the detailed design model by specifying RT UML notations in the precise behaviors and structures of the quadrotor UAV controller. To realize quadrotor UAV controller, the PSM is firstly implemented to simulation model, which is transformed from the identified PIM by applying the determined model transformation rules. The obtained simulation results permits us to theoretically evaluate the system control performance and functionalities, and to easily optimize control design elements of this system before we decide to realize and deploy it. Then, the PIM with the modifying control elements optimized in the PSM of simulation model is adapted to obtain the new updated PIM for the realization model. This updated PIM is converted into new PSMs by using different object-oriented specific platforms in order to completely realize the quadrotor UAV controller with compatible microcontrollers. Based on this approach, a trajectory-tracking controller of an autonomous mini quadrotor UAV has been completely developed with the simulation model of *OpenModelica*, and *Arduino Mega2560* microcontroller for the realization model. The detailed experimental scenarios are currently performed to improve the performances and features of this application in the aeronautic laboratory. This application can be extended with the increase in the altitude, radius of action, velocity and duration of autonomy time by using compatible physical components such as the engineering material, power resource, vision-based navigation components, etc. But the activities of our process model described in this paper do not change of in spite of this extended quadrotor UAV control application.

The re-use is very important to develop controllers for different quadrotor UAVs in our approach. Reusable views in the development phase of this system are based on virtual mechanisms of objects or classes, and design capsule components of CIM and PIM. For example, the global state machine, industrial operational constraints, communication patterns and structures of main control capsules can be customizable and reusable to carry out different quadrotor UAV control applications. Furthermore, using the approach described in this paper, development engineers will be more capable of managing the system complexity through the visual modeling of artifacts and their transformations of this process. In particular, they can handle the defined design elements in the PIM to quickly deploy the quadrotor UAV controller to different object-oriented specific platforms to which they want to suitably realize it.

In the next time, we will develop this approach combined with various control formalisms and architectures in order to perfectly analyze, design, implement and realize controllers for balancing search and target response in cooperative autonomous quadrotor UAV teams.

References:

[1] Arduino, *Open-source electronics prototyping platform for hardware and software*, http://www.arduino.cc/, 2012.

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

[3] Carloni, L. P., Passerone, R., Pinto, A., Sangiovanni, V. A., *Languages and Tools for Hybrid Systems Design*, now Publishers Inc., Boston, 2006.

[4] Carrillo L.R.G., Lozano R., López A.E.D., Pégard C., *Quad rotorcraft Control: Vision-Based Hovering and Navigation*, Springer-Verlag London, 2013.

[5] Douglass, B.P., *Real Time UML: Advances in the UML for Real-Time Systems*, third edition, Addison-Wesley, 2004.

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

[7] Guillaume J. J. D., *Fault-tolerant Flight Control and Guidance Systems: Practical Methods for Small Unmanned Aerial Vehicles*, Springer, 2009.

[8] Hien N.V., Soriano, T., Implementing hybrid automata for developing industrial control systems, *Proc. of 8th IEEE-ETFA*, ISBN 0-7803-7241-7, doi:10.1109/ETFA.2001.997679, Vol. 2, 2001, pp. 129-137.

[9] Hien N.V., Vinh H. T., Soriano T., Using Model-Driven Architecture to Develop Industrial Control Systems, *Proc. of 4th IEEE-RIVF*, ISSN 1621-0875, 2006, pp. 75-80.

[10] Hien N.V. et al., *A Method of Model-Driven Architecture to Develop Industrial Hybrid Dynamic Systems*, Final report of research project, code: B2010-01-354, Hanoi University of Science and Technology, 2011.

[11] Hien N. V., Soriano T., A Model Transformation Process to Realize Controllers of Ship Autopilot Systems by the Specialized MDA's Features with UML/SysML, *Proc. of IEEE Conference on MECATRONICS-REM*, ISBN 978-1-4673-4771-6, doi:10.1109/MECATRONICS.2012.6450983, 2012, pp. 20-26.

[12] Hung N.P., Diem P.G., Khanh N.P. et al., *Research, 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, 2012.

[13] IBM - *IBM Rational Online Documentation, Training Kit, Software Delivery Platforms*, https://www.ibm.com/developerworks/university/, 2010.

[14] Lavagno, L., Martin, G., Selic, B. (Eds.), *UML for Real: Design of Embedded Real-Time Systems*, Kluwer Academic Publishers, 2003.

[15] Lin H.J., Tsay T.S., Modeling Identification and Simulation of Bank to Turn Unmanned Aerial Vehicle, *WSEAS Transactions on Systems*, ISSN 1109-2777, Issue 4, Volume 10, 2011, pp. 91-103.

[16] Neri F., Traffic packet based intrusion detection: decision trees and generic based learning evaluation, *WSEAS Transactions on Computers*, ISSN 1109-2750, WSEAS Press (Wisconsin, USA), Issue 9, Volume 4, 2005, pp. 1017-1024.

[17] Nonami K., Kendoul F., Suzuki S., Wang W., Nakazawa D., *Autonomous Flying Robots - Unmanned Aerial Vehicles and Micro Aerial Vehicles*, Springer, 2010.

[18] OMG, *Specifications of MDA, ver. 1.01*, http://www.omg.org/mda/, 2003.

[19] OMG, *Unified Modeling Language, ver. 2.1.1*, http://www.omg.org/spec/UML/, 2007.

[20] OMG, *UML Profile for MARTE: Modeling and Analysis of Real-time Embedded Systems, ver. 1.1*, http://www.omg.org/spec/MARTE/, 2011.

[21] OpenModelica, *OpenModelica Simulation software*, v1.9, http://www.openmodelica.org, 2013.

[22] Pakzad M.A., Kalman Filter Design for Time Delay Systems, *WSEAS Transactions on Systems*, E-ISSN 2224-2678, Issue 10, Volume 11, 2012, pp. 551-560.

[23] Pekar L., Neri F., An Introduction to the Special Issue on Time Delay Systems: Modelling, Identification, Stability, Control and Applications, *WSEAS Transactions on Systems*, E-ISSN 2224-2678, Issue 10, Volume 11, 2012, pp. 539-540.

[24] Peter F., *Introduction to Modeling and Simulation of Technical and Physical with Modelica*, John Wiley & Sons, 2011.

[25] Samir B., *Design and control of quadrotors with application to autonomous flying*, PhD Thesis, École Polytechnique Fédérale de Lausanne, France, 2007.

[26] Soriano T., Sghaier A., Hien N.V., Mechatronics Design from an Object-Oriented Point of View, *WSEAS Transactions on Communications*, ISSN 1109-2742, Issue 1, Volume 3, 2004, pp. 282-287.

[27] Tsay T.S., Intelligent Guidance and Control Laws for an Autonomous Underwater Vehicle, *WSEAS Transactions on Systems*, ISSN 1109-2777, Issue 5, Volume 9, 2010, pp. 463-475.

[28] Xun G., Zhicheng H., et al., Backstepping Sliding Mode Attitude Control of Quad-rotor with Adaptive Algorithm, *2012 2nd International Conference on Materials, Mechatronics and Automation, Lecture Notes in Information Technology*, Vol.15, ISBN 978-1-61275-015-6, ISSN: 2070-1918, ©2012 IERI, pp. 410-415.

[29] Yanushevsky R., *Guidance of Unmanned Aerial Vehicles*, CRC Press, Taylor & Francis Group, 2011.

[30] Yingcai B., Haibin D., Implementation of autonomous visual tracking and landing for a low-cost quadrotor, *Optik - International Journal for Light and Electron Optics*, ISSN: 0030-4026, Volume 124, Issue 18, 2013, pp. 3296–3300.

[31] Yu. Y., Sun F., Wang Y., Controller Design of Quadrotor Aerial Robot, *Elsevier, Physics Procedia 33*, 2012, pp. 1254-1260.