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*.

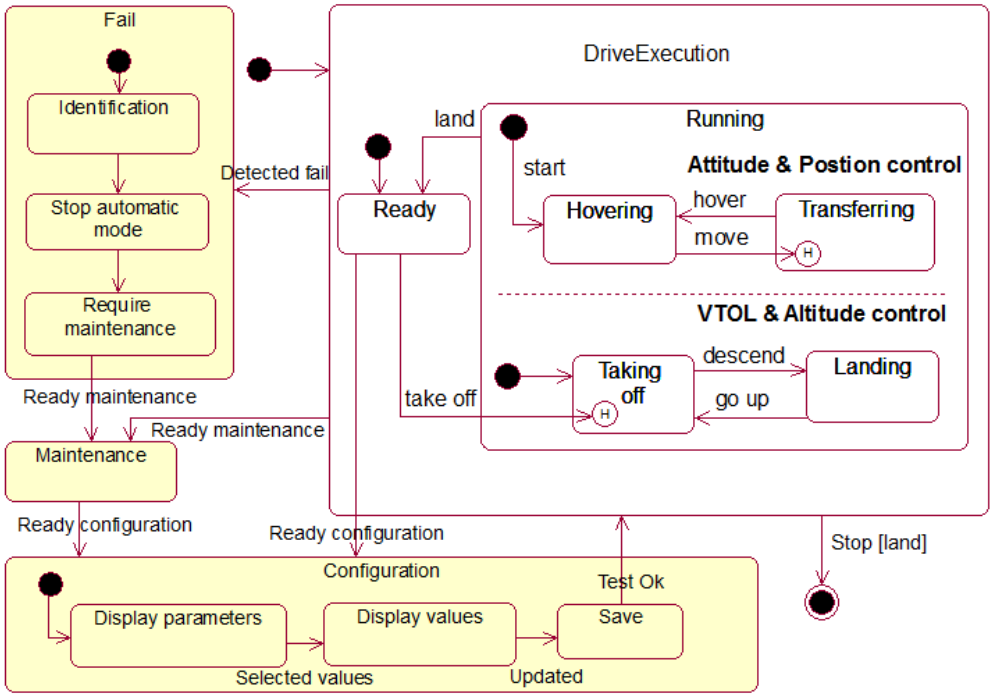

Fig. 10. Global state machine of the quadrotor UAV controller being developed.

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.

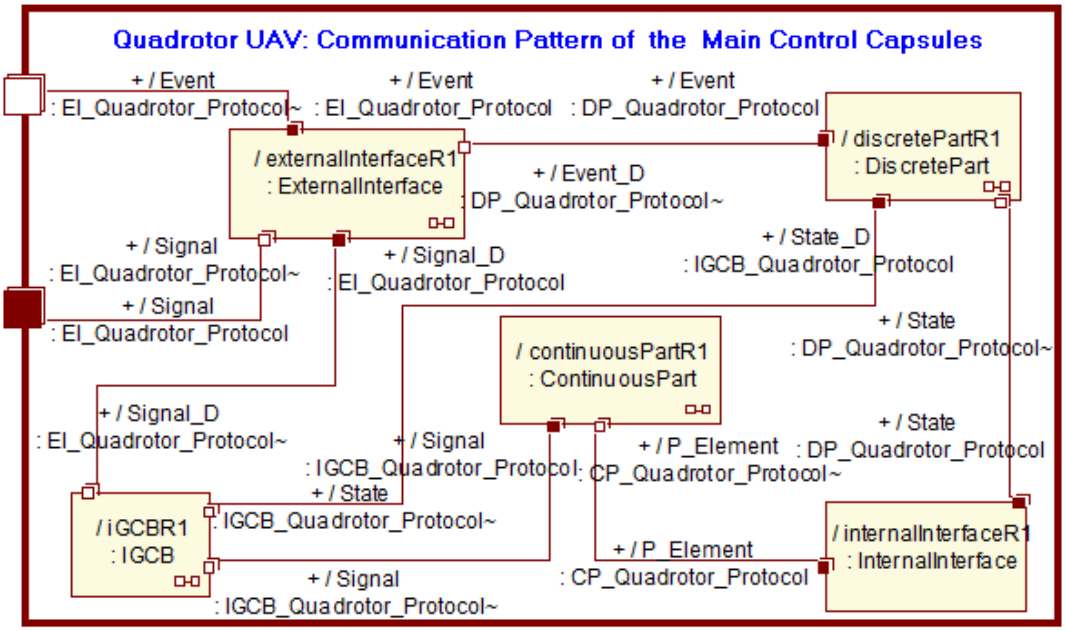

Fig. 11. Communication pattern of main control capsules for Quadrotor UAV controllers.

- 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].

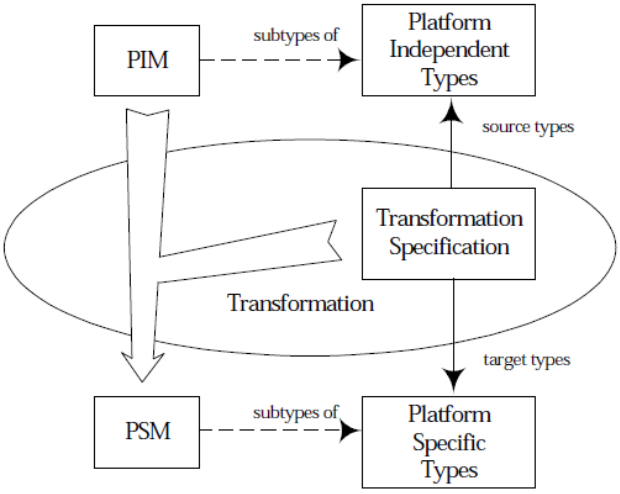

Fig 12. Model transformation by types.

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.

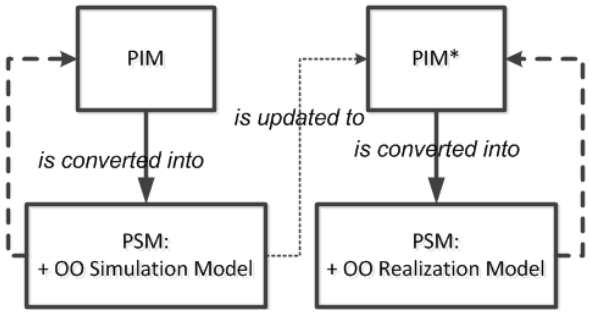

Fig. 13. A sketch of the model transformation for controllers of quadrotor UAVs.

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.

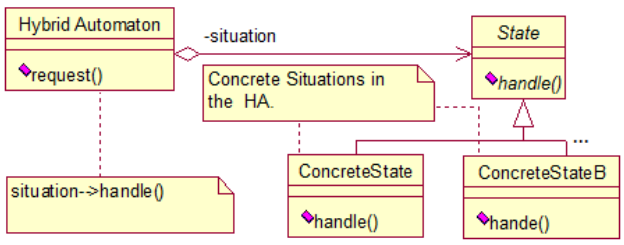

Fig. 14. Implementation structure for the HA of quadrotor UAVs.

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.

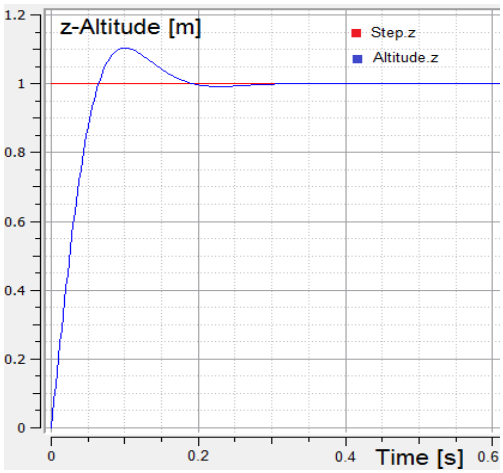

Fig. 15. Transient control response in z-direction

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.

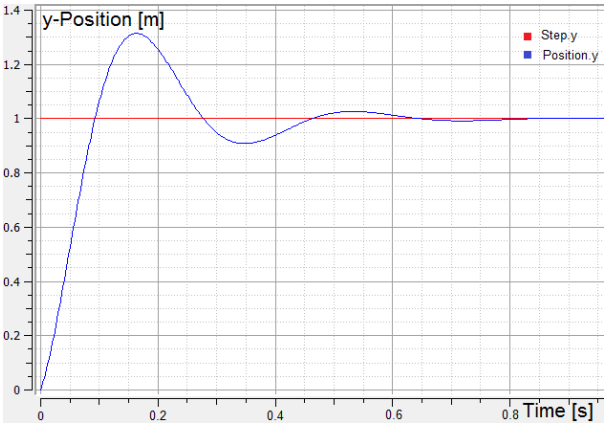

Fig. 16. Transient control response in y-direction

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/MECA-TRONICS.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/universit y/, 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.