A Real-Time Capsule-Based Design Model to Realize AUV Controllers

Ngo Van Hien
Hanoi University of Science and Technology, No. 1, Dai Co Viet, Hai Ba Trung, Hanoi, Viet Nam
Received: October 10, 2017; Accepted: November 03, 2017

Abstract

This paper brings out a real-time capsule model of Autonomous Underwater Vehicles (AUVs) controllers, which is based on the real-time Unified Modeling Language (UML) with a Domain-Specific Language (DSL) of Modeling and Analysis of Real-Time and Embedded Systems (MARTE) in order to intensively carry out the whole of development lifecyle for the AUV's control system. The main study is stepwise carried out as follows: the AUV dynamics together with control structure are firstly adapted for developing entirely an AUV controller. The use-case model combined with an implementable functional block diagram and the Extended Kalman Filter (EKF) algorithm are then specialized to closely gather the requirements analysis of control. The specializations of real-time UML/MARTE's features combined with the capsule evolution of timing concurrency are next realized to precisely design structures and behaviors for the controller. The detailed design model is then converted into the implementation model by using open-source platforms in order to quickly simulate and realize this controller. Finally, a trajectory-tracking controller, which permits a miniature unmanned submarine possessing a torpedo shape to autonomously reaches and follows a horizontal planar reference trajectory, was completely deployed and tested.

Keywords: AUV control, Capsule-based design, EKF, real-time UML/MARTE

1. Introduction

Autonomous Underwater Vehicles (AUVs) are increasingly used by civil and military operators for performing the complex underwater missions. This is due to the basic features of safety and efficiency when compared to manned vehicles. AUV does not require human operators and subject to the conditions and the dangers inherent in the underwater environment. With such outstanding features, the type of AUV has been used successfully and effectively in the maritime industry for both the civil and military purposes [1, 2].

Within the autonomy architecture of AUVs are three main systems. These are: the guidance system, which is responsible for generating the trajectory for the vehicle to follow; the navigation system, which produces an estimation of the current state of the vehicle; and the control system, which calculates and applies the appropriate forces to manoeuvre the vehicle [3]. All three of these systems have their own individual tasks to complete, yet must also work cooperatively in order to reliably allow an AUV to complete its objectives. Hence, the AUV controller must take into account models with discrete events and continuous behaviors that can be considered as a Hybrid Dynamic Systems (HDS) [4].

In addition, the customization and reusability are factors to be associated with the production of a new application in order to reduce its costs, resources and time development. According to the Object Management Group (OMG) [5], UML appeared to us to be essential for its visual object-oriented design support, which has been largely spread and appreciated in the software industry. However, UML is not well adapted to visualize, interconnection types between control objects or sub-systems for modeling industrial control systems. Furthermore, the System Modeling Language (SysML) [6], which is a UML profile for systems engineering, has been standardized by OMG. SysML supports the specification, analysis, design, verification and validation of a broad range of complex systems. But both of UML and SysML lack the constructs for modeling time and duration constraints of the developed system. Hence, the real-time UML/MARTE version [7-9] is chosen to model in detail the analysis and design artifacts for real-time and embedded control systems, e.g. the AUV controller. This version also includes the 'capsules, ports, protocols, connectors' concepts that can be adapted by specializing a set of control capsules in precise behaviors and structures of the AUV controller.

The paper aims to implement a control model integrated the AUV dynamics for control into the real-time object paradigms, which can permit us to intensively realize and deploy the AUV controller, and also allow the designed and implemented control

[*] Corresponding author: Tel.: (+84) 904.255.855
Email: hien.ngovan@hust.edu.vn

elements to be closely customizable and re-usable in the realization of new applications for various AUV types. In the current model, the AUV dynamics and control structure are also adapted in detail for the AUV controller that are then combined with the models as follows: The Object-Oriented (OO) Analysis (OOA), OO Design (OOD) and OO Implementation (OOImpl) models; this control system permits an AUV to track a reference trajectory. Here, the OOA includes the use-case model specialized closely with an implementable function block diagram to precisely capture the requirement analysis for an AUV controller; the OOD model is built on the identified OOA model by specifying the real-time UML/MARTE to entirely design the real-time control capsules with their timing concurrency of evolutions in detail. The detailed OOD elements is then converted into OOImpl models by using open-source platforms such as *Arduino* [10] in order to quickly simulate, realize and deploy the AUV controller. Finally, a planar trajectory-tracking controller of a miniature unmanned submarine was developed and taken on trial trip.

2. AUV dynamics and control structure

2.1. Overview of AUV dynamics for control

According to SNAME [11], the six motion components of an underwater vehicle are defined as *surge*, *sway*, *heave*, *roll*, *pitch*, and *yaw* which are shown in Table 1.

Table 1. SNAME notations for underwater vehicles

Degree of freedom	Motions	Force and moment	Linear and angular velocity	Position and Euler angles
1	Surge	X	u	x
2	Sway	Y	v	y
3	Heave	Z	w	z
4	Roll	K	p	ϕ
5	Pitch	M	q	θ
6	Yaw	N	r	ψ

From the large field of guidance, navigation and control of underwater vehicles, the 6 DoF dynamic model of AUVs in body frame [3] can be written in equation (1).

$$\begin{cases} \dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau}(\boldsymbol{\nu}, \mathbf{u}) \end{cases} \quad (1)$$

Where: $\boldsymbol{\eta}=[\boldsymbol{\eta}_1^T, \boldsymbol{\eta}_2^T]^T$ includes the position $\boldsymbol{\eta}_1=[x, y, z]^T$ (NED: *North, East and Down*) and the orientation $\boldsymbol{\eta}_2=[\phi, \theta, \psi]^T$ (*Euler* RPY: *Roll*, *Pitch* and *Yaw* angles); $\boldsymbol{\nu} = [\boldsymbol{\nu}_1^T, \boldsymbol{\nu}_2^T]^T$ is composed the linear $\boldsymbol{\nu}_1=[u, v, w]^T$ and the angular $\boldsymbol{\nu}_2=[p, q, r]^T$ velocities; $\boldsymbol{M}=\boldsymbol{M}_{RB}+\boldsymbol{M}_A$ is a mass matrix, which denotes the 6×6 system inertia matrix containing $\boldsymbol{M}_{RB}$ - the generalized constant inertia matrix, and $\boldsymbol{M}_A$ - the added mass inertia matrix; $\boldsymbol{C}(\boldsymbol{\nu})=\boldsymbol{C}_{RB}(\boldsymbol{\nu})+\boldsymbol{C}_A(\boldsymbol{\nu})$ is the 6×6 *Coriolis* and centripetal forces matrix including added mass; Linear and nonlinear hydrodynamic damping are contained within the 6×6 matrix $\boldsymbol{D}(\boldsymbol{\nu})=\boldsymbol{D}+\boldsymbol{D}_n(\boldsymbol{\nu})$, $\boldsymbol{D}$ contains the linear damping terms, and $\boldsymbol{D}_n(\boldsymbol{\nu})$ contains the nonlinear damping terms; $\boldsymbol{g}(\boldsymbol{\eta})$ is the 6×1 vector of gravitational and buoyancy effects; $\boldsymbol{\tau}(\boldsymbol{\nu}, \mathbf{u})$ is the vector of resultant force and moment acting on the underwater vehicle, and $\mathbf{u}$ is the control inputs, e.g. the rotational speed of the motors related to the generated thrusts, the driving angles sent to the needed servo-motor for sail planes and rudder.

A discrete state-space representation is required for modeling the AUV controller in order to use a recursive digital motion estimation filter, e.g. the Extended *Kalman* Filter (EKF); the developed system can be then described by a set of equations (2).

$$\begin{cases} \mathbf{x_k} = \mathbf{f_{k-1}}(\mathbf{x_{k-1}}, \mathbf{u_{k-1}}) + \mathbf{w_{k-1}} \\ \mathbf{y_k} = \mathbf{h_k}(\mathbf{x_k}) + \mathbf{v_k} \end{cases} \quad (2)$$

Here, $\mathbf{x} = \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\nu} \end{bmatrix}$ is a 12-dimensional state vector for describing the motion of AUV, while $\mathbf{x_k}$ is the vector of state variables at the kth instant of $\mathbf{x}$; $\mathbf{u_k}$ and $\mathbf{y_k}$ are respectively the inputs and outputs of the system; $\mathbf{w_k}$ and $\mathbf{v_k}$ are the additive process and measurement noise; the first equation in (2) is called the system's evolution equation, while the second one is called the measurement equation.

2.2. Control structure for an AUV

As previously stated, main sub-systems, which can be participated in the physical control architecture of AUVs are the guidance system, navigation system, and control system. Fig. 1 shows out a functional block diagram, which captures how these sub-systems interact.

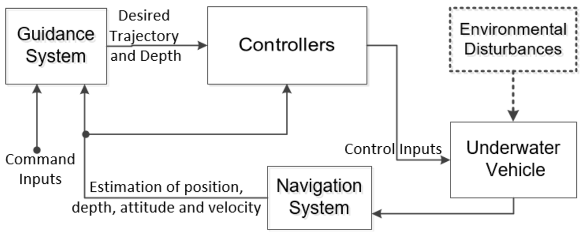
Guidance System
Desired Trajectory and Depth
Controllers
Environmental Disturbances
Command Inputs
Control Inputs
Underwater Vehicle
Estimation of position, depth, attitude and velocity
Navigation System

Fig. 1. Block diagram of guidance, navigation and control for an AUV.

Here, the *Guidance System* block is responsible for producing the desired trajectory for the vehicle to follow; The *Navigation System* block addresses the task of determining the current state of the AUV; The *Controllers* block is responsible for providing the corrective signals and events to enable the AUV to follow a desired path. This is achieved by receiving the desired state of AUV from the *Guidance System* block, and the current state of AUV from the *Navigation System* block. It then calculates and applies correcting forces and moments, through use of the various actuators on the AUV, to minimize the difference between desired and current states. This allows the AUV to track a desired trajectory even in the presence of unknown disturbances.

From the above described AUV dynamics together with its general control structure and characteristics of HDS [4], controllers of AUV are then HDS. These controllers have the continuous/discrete parts and their interactions such as the motions in *surge*, *sway*, *heave*, *roll*, *pitch*, and *yaw*, and external interacting events from the guidance and navigation system and environmental disturbances. In the current model, the paper is interested in developing the trajectory-tracking controller of AUVs, so this hybrid dynamic model can be used to find out the control algorithms combined with a specific guidance law such as the implemented Line-of-Sight (LOS) guidance described in [12].

3. Capsule-based development for AUV controllers

As the previous state, the real-time UML/MARTE version was chosen to model in detail the analysis and design artifacts for real-time and embedded control systems, e.g. the AUV controller. Starting from the above adapted AUV dynamics, control structures, real-time UML/MARTE features together with the author's object-unified approach for AUV controllers [13], we develop in detail a model-driven implementation for the AUV controller, which includes three models as follows: OOA, OOD and OOImpl model so separate the specification of the operation of the system from the details of the way that system uses the capabilities of its platform.

i) In OOA model, the use case model is specialized by the implementable functional block diagram to closely capture the requirements analysis for an AUV controller.

ii) OOD model is built up by specifying the real-time control capsules, ports, protocols and their timing concurrency of evolutions together with the EKF algorithm in order to model the precise behaviors and structures of AUV controller.

iii) OOD model is then converted into OOImpl model by object-oriented specific platforms such as *MatLab/Simulink*, *Arduino* etc. in order to completely simulate, realize and deploy the AUV controller.

3.1. OOA model for an AUV controller

Following the AUV dynamic model and control architecture described in Section 2 together with LOS guidance described in [12], we present here the main use case model (Fig. 2) of AUV controllers. Here, MDS represents the *Measurement and Display System* combined with the guidance and navigation system; MES represents the *Marine Environment System* including disturbances such as the wind, waves, ocean currents etc. In this model, it is necessary to provide industrial conditions, e.g. the maximum swing angles of rudder and sail planes, velocity, immersible depth and additional safe trip modes of the AUV in order to entirely make in the operational safety of system.

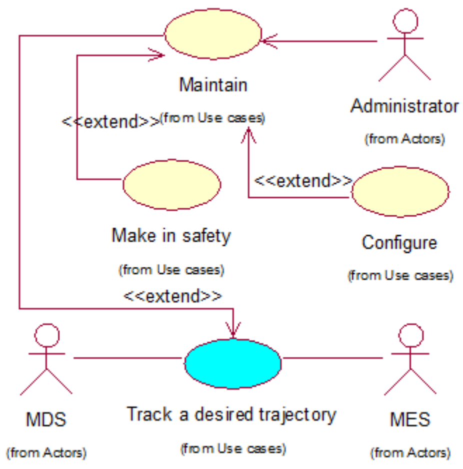

Fig. 2. Main use case model for the AUV controller.

In addition, an implemented functional block diagram must be defined in order to model continuous behaviors of this system with events issued from outside; because the real-time UML/MARTE lacks the constructs for modeling internal continuous behaviors for each state on the state machine diagram. Starting from the considered dynamic model of AUV as well as the defined use case model, we propose here an implemented functional block diagram of the AUV controller as shown in Fig. 3. Here, *Desired trajectory* and *depth* actions respectively give the desired position (x_d, y_d) and depth (z_d) to the position and deep controller; ΣT_d is the desired overall thrust; the position controller receives the AUV'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* block; the attitude controller gives then the desired control signals to the actuator commands (e.g. Ω_{di} can be the desired motor speeds sent to the main motor controllers for the propellers or tunnel thrusters, and Ω_{di} can be also the desired driving steps sent to the needed servo-motor controllers for sail planes, rudder and displacement units, $i = \overline{1, n}$ for an AUV operating with n actuators, so **u** will be the control input of size n×1); the Proportional-Integral-Derivative (PID) regulators can be applied to the *Motor Control* block including the main motor controllers and servo-motor controllers in order to reduce the inertial and delay time caused by the physical AUV actuators in the whole system evolution; $\tau_{\phi,\theta,\psi}$ and ΣT are respectively the overall moment and force acting on the AUV.

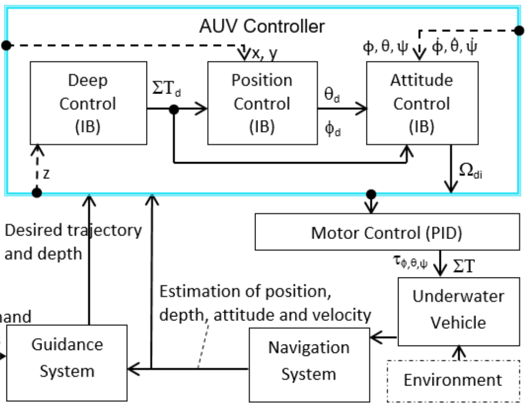

Fig. 3. Implemented functional block diagram for the AUV controller.

In the current model, the Integral Backstepping (IB) techniques implemented in [14] are hierarchically used for control the depth, position and attitude of the AUV. The state-space models (2) described in Sub-section 2.1 are used for the estimation and prediction of the position, depth, attitude, and velocity corresponding to the sensors installed on the AUV that are implemented in the *Navigation System* block that are based on the standard navigation filter is based on EKF [15].

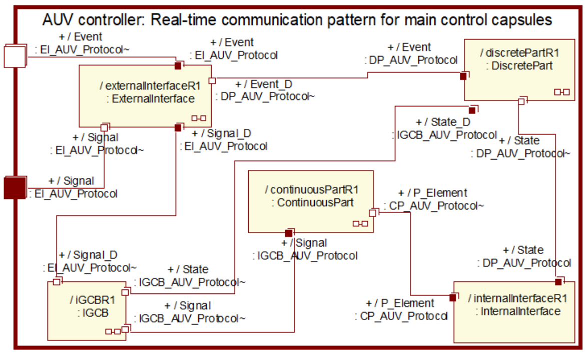
AUV controller: Real-time communication pattern for main control capsules
+ / Event : EI_AUV_Protocol~
+ / Event : EI_AUV_Protocol
+ / Event : DP_AUV_Protocol
/ discretePartR1 : DiscretePart
/ externalInterfaceR1 : ExternalInterface
+ / Event_D : DP_AUV_Protocol~
+ / State_D : IGCB_AUV_Protocol
+ / Signal : EI_AUV_Protocol~
+ / Signal_D : EI_AUV_Protocol
+ / State : DP_AUV_Protocol~
+ / Signal : EI_AUV_Protocol
/ continuousPartR1 : ContinuousPart
+ / P_Element : CP_AUV_Protocol~
+ / Signal : IGCB_AUV_Protocol
+ / State : DP_AUV_Protocol
+ / Signal_D : EI_AUV_Protocol~
+ / State : IGCB_AUV_Protocol~
/ iGCBR1 : IGCB
+ / Signal : IGCB_AUV_Protocol~
+ / P_Element : CP_AUV_Protocol
/ internalInterfaceR1 : InternalInterface

Fig. 4. Real-time communication pattern for the main control capsules for the AUV controller.

3.2. OOD for an AUV controller

In the OOD model, we have specialized the 5 main control capsules, which take part in the HA realization of the AUV: 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. 4 indicates the real-time communication pattern of main control capsules by using the real-time UML/MARTE's collaboration diagrams.

Fig. 5 describes in detail the timing concurrency of evolutions for the above real-time communication pattern of main control capsules. Here, Ee_1, Ee_2, Ee_3… are the external events coming from the external interface's capsule; Ei_1, Ei_2, Ei_3… are internal events issued by the evolution of the internal interface's capsule; q_1, q_2, q_3… indicate the concrete situations (states) of the AUV controller; ec_1, ec_2, ec_3… represent the evolutions of continuous elements in the continuous part's capsule; and ΔT is a sampling period of the IGCB's capsule. The realization hypotheses of timing concurrency for capsule evolutions are applied as follows:

- If the end of the discrete part's evolution is located before or just at the sampling date of the IGCB's capsule, then the current IGCB model will change to the new IGCB model corresponding to this evolution;

- If the end of the discrete part's evolution is located after of appearing sampling date (ΔT), then the current IGCB model is not commutated;

- If an event appears during the evolution of the local state machine of AUV application, then this event is immediately memorized and solved later on;

- All of the external and internal events have the same process by the discrete part's capsule;

- During the sampling period of the IGCB's capsule, the continue part's capsule, internal interface's capsule and discrete part's capsule make their own evolutions to possibly commutate to a new IGCB model, the IGCB continuous model remains in its current mode for this period;

- So during the period of the IGCB's capsule, the current IGCB model can detect two or more appeared situations, then the IGCB's capsule synchronizes all these situations just at the end of this period with the null timing duration; the current IGCB model subsequently changes to a new IGCB model, which corresponds to the last appeared situation during this period.

The validation and verification of this OOD model and its traceability with the above defined OOA model have been corrected by using *IBM Rational Rose RealTime* or *IBM Rational Rhapsody* software [16]. *IBM Rational*'s leading role in defining the real-time UML is widely acknowledged, as is the pre-eminence of the *IBM Rational Rose RealTime* product in implementing UML to support the architecting of large-scale real-time and embedded software systems. It combines a rich modeling environment with a code-oriented tool set to create a comprehensive practitioner desktop for creating solutions in a variety of architectural styles, and targeted at specific runtime infrastructures.

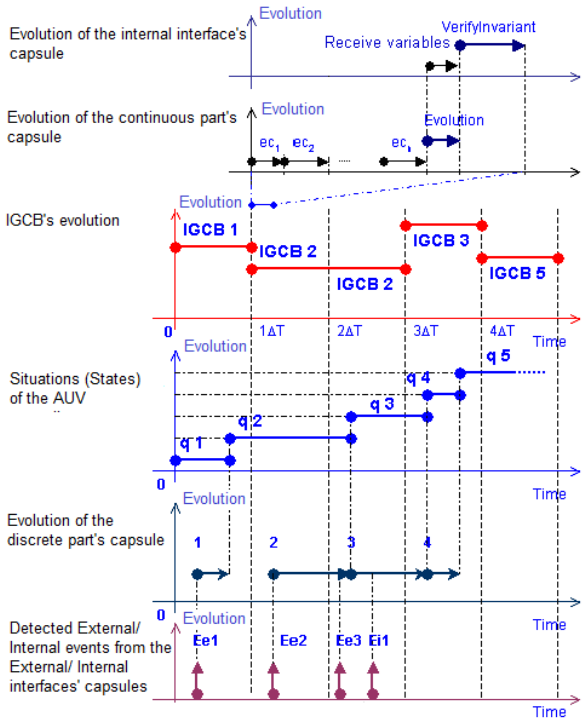
Evolution of the internal interface's capsule
Evolution
Receive variables
VerifyInvariant
Evolution of the continuous part's capsule
Evolution
ec_1 ec_2
ec_i
Evolution
IGCB's evolution
Evolution
IGCB 1
IGCB 2
IGCB 2
IGCB 3
IGCB 5
0
1ΔT
2ΔT
3ΔT
4ΔT
Time
Situations (States) of the AUV
Evolution
q 1
q 2
q 3
q 4
q 5
0
Time
Evolution of the discrete part's capsule
Evolution
1
2
3
4
0
Time
Detected External/ Internal events from the External/ Internal interfaces' capsules
Evolution
Ee1
Ee2
Ee3
Ei1
Time

Fig. 5. Timing concurrency of evolutions for main control capsules of the AUV controller.

3.3. OOImpl for an AUV controller

To carry out the AUV controller, the OOD model is firstly implemented to the simulation model that is transformed from the above built OOD model by using tools such as *IBM Rational Rhapsody* [16] and *MatLab/Simulink* or *OpenModelica* [17]. The OOD model with the optimized control elements of simulation model is then adapted to obtain the new updated OOD model for realization models of AUV that will be called OOD*. Finally, this OOD* model is converted into new OOImpl models by using different specific platforms, which are based on the object-oriented Implementation Development Environment (IDE), e.g. the *Arduino*'s platforms [10] in order to completely realize the AUV controller with compatible microcontrollers. To implement the realization model for an AUV controller, we have to update the design model with the control elements modified in the acceptable simulation model, e.g. the control law and its parameters, continuous elements, etc. Then, we convert this updated design model into different IDEs, which support object-oriented programming languages such as C++, Java, Ada, etc. in order to completely realize it in industrial platforms, e.g. the microcontrollers. This model conversion can be carried out by using object-oriented modeling software tools, which support the round-trip engineering such as *IBM Rational Rhapsody* [16]. 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.

4. Application

Following the above proposed model, we developed a planar trajectory-tracking controller of a Miniature Unmanned Submarine (MUS) possessing a torpedo shape, which autonomously reaches and follows a geometric reference path starting from a given initial configuration. In our case study, the propeller, sail planes, rudder and displacement unit are used to provide the translational forces and rotational moments that drive MUS. The main characteristics of MUS are resumed in Table 2.

We have used then *Arduino* platform [10] to quickly deploy the realization model for 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 the pressure and magnetometer sensors, Inertial Measurement Unit (IMU), GPS, e.g. *MPU6000* with working frequency 10Hz [18], *Ublox Neo 6M* with working frequency 100Hz [19], etc. and can affect its surroundings by controlled actuators. *Arduino ATMEGA32-U2* and *STM32 Cortex-M4* microcontrollers [10] have been used on the board, and can be programmed by using the *Arduino* programming language based on the object-oriented embedded programming C++.

We have performed trial trips to test the realization model of this application. Fig. 6 shows out the installation of the whole of MUS components to prepare test cases for the planar trajectory-tracking controller of MUS. The test scenarios are based on the use-case model, various desired shape-based reference paths of MUS. Fig. 7 illustrate the horizontal planar trajectory-tracking controller that permits the MUS autonomously reaches and follows the rectangle-shaped reference paths.

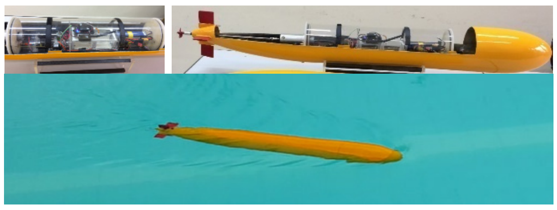

Fig. 6. Setup and test for the trajectory-tracking controller of the MUS.

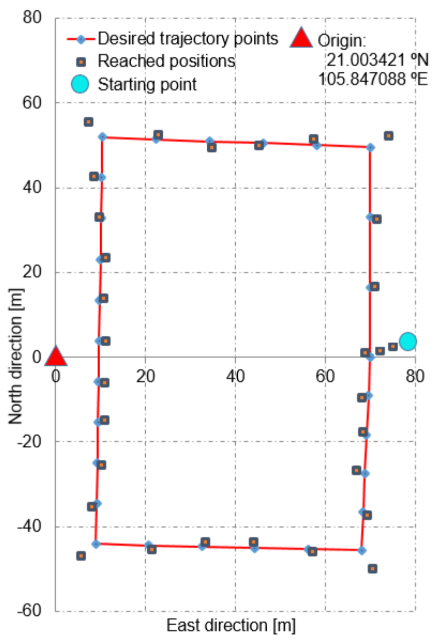

Fig. 7. MUS reaches and follows the rectangle-shaped reference path.

Based on the comparison between the experimental data of trial tests and the obtained simulation results, the planar trajectory-tracking controller of this MUS was satisfied with performance requirements, e.g. the admissible control duration, transition and static errors. In this application, the standard control method of IB and the EKF algorithm were used for the position and attitude control and the PID regulators were applied to the block of motor controllers that permit us to implement the functional block diagram (Fig. 3) for building up the ICCB's capsule of OOD model.

5. Conclusion

This paper presented a specific model-driven implementation to intensively develop controllers for AUVs. This model is mainly based on the specializations of real-time UML/MARTE to intensively analyze, design, implement and realize the control parts of the system. The paper includes the following main points: The AUV dynamics and control architecture are adapted for control that are then combined with the specialization of real-time object features including the OOA, OOD and OOImpl components. In the OOA model, the use case model is specialized with the implementable functional block diagram for an AUV controller. The OOD model is built for obtaining the detailed design model by specifying the real-time control capsules, ports, protocols enclosed with their timing concurrency evolutions of capsules in order to model and construct in detail the behaviors and structures of AUV controllers. The OOD model with the optimized control elements is then adapted to obtain the new updated OOD model for the realization model. This updated OOD model is converted into new OOImpl models by using different object-oriented specific platforms in order to completely realize the AUV controller with compatible microcontrollers. Based on this model, a planar trajectory-tracking controller of a low-cost miniature unmanned submarine was deployed and tested out *Arduino ATMEGA U2* and *STM32-Cortex-M4* microcontrollers.

In the next time, we will investigate in strategy by equipping depth control and navigation sensors and using industrial microcontrollers in order to completely make up controllers for balancing search and target response in cooperative team of various AUVs.

References

[1]. Alam K, Ray T, Anavatti S. A brief taxonomy of autonomous underwater vehicle design literature. Ocean Engineering, Elsevier, ISSN 0029-8018,. 2014;88:627-30.

[2]. Wynn R, Huvenne V, Bas T, Murton B, Connelly D, Bett B, et al. Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience. Marine Geology - International Journal of Marine Geology, Geochemistry and Geophysics, Elsevier, ISSN 0025-3227,. 2014;352:451-68.

[3]. Fossen T. Handbook of Marine Craft Hydrodynamics and Motion Control. United Kingdom: John Wiley & Sons; 2011.

[4]. Carloni L, Passerone R, Pinto A, Sangiovanni V. Languages and Tools for Hybrid Systems Design. Boston: Now Publishers Inc; 2006.

[5]. OMG. Documents Associated With Unified Modeling Language™ (UML® Version 2.5): OMG formal/15-03-01. http://www.omg.org/spec/UML/2.5/: OMG; 2015.

[6]. OMG. SysML Specifications Version 1.4: OMG formal/2015-06-03. http://www.omg.org/spec/SysML/1.4/: OMG; 2015.

[7]. OMG. UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. OMG Formal Version. http://www.omg.org/spec/MARTE/: OMG; 2011.

[8]. Douglass B. Real-Time UML Workshop for Embedded Systems. 2nd edtion. Oxford, UK: Elsevier; 2014.

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

[10]. Arduino. Open-source electronics prototyping platform for hardware and software. Arduino; 2016.

[11]. SNAME. Nomenclature for Treating the Motion of a Submerged Body through a Fluid, Technical and Research Bulletin No. 1-5. New York 18, N. Y., USA: SNAME (the Society of Naval Architects and Marine Engineers); 1950.

[12]. Lekkas A, Fossen T. Integral LOS Path Following for Curved Paths Based on a Monotone Cubic Hermite Spline Parametrization. IEEE Transactions on Control Systems Technology, ISSN 1063-6536,. 2014;22:2287 - 301.

[13]. Soriano T, Hien N, Tuan K, Anh T. An object-unified approach to develop controllers for autonomous underwater vehicles. Mechatronics: The Science of Intelligent Machines, Elsevier, ISSN 0957-4158,. 2016;35:54-70.

[14]. Li W, Wu W, Wang J, Wu M. A novel backtracking navigation scheme for Autonomous Underwater Vehicles. Measurement, Elsevier, ISSN 0263-2241,. 2014;47:496–504.

[15]. Allotta B, Caitib A, Costanzi R, Fanelli F, Fenucci D, Meli E, et al. A new AUV navigation system exploiting unscented Kalman filter. Ocean Engineering, Elsevier, ISSN 0029-8018,. 2016;113:121–32.

[16]. IBM. IBM Rational Online Documentation and Training Kit. IBM; 2016.

[17]. OpenModelica. OpenModelica. OpenModelica software, version 1.11.0. OpenModelica; 2016.

[18]. InvenSense. Sensor System on Chip. 2014.

[19]. u-blox. Gobal leader in wireless communications and positioning semiconductors and modules for the industrial, automotive and consumer markets. u-blox; 2014.