applied sciences

Article

A MBSE Application to Controllers of Autonomous Underwater Vehicles Based on Model-Driven Architecture Concepts

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

1 School of Transportation Engineering, Hanoi University of Science and Technology, Hanoi 10000, Vietnam
2 College of Systems Engineering and Science, Shibaura Institute of Technology, Tokyo 135-8548, Japan
3 School of Mechanical Engineering, Hanoi University of Science and Technology, Hanoi 10000, Vietnam
* Correspondence: hien.ngovan@hust.edu.vn (N.V.H.); he.ngovan@hust.edu.vn (N.V.H.); thuan.truongvan@hust.edu.vn (V.-T.T.); tambn@shibaura-it.ac.jp (N.-T.B.)

Received: 30 September 2020; Accepted: 21 November 2020; Published: 23 November 2020

Abstract: In this paper, a hybrid realization model is proposed for the controllers of autonomous underwater vehicles (AUVs). This model is based on the model-based systems engineering (MBSE) methodology, in combination with the model-driven architecture (MDA), the real-time unified modeling language (UML)/systems modeling language (SysML), the extended/unscented Kalman filter (EKF/UKF) algorithms, and hybrid automata, and it can be reused for designing controllers of various AUV types. The dynamic model and control structure of AUVs were combined with the specialization of MDA concepts as follows. The computation-independent model (CIM) was specified by the use-case model combined with the EKF/UKF algorithms and hybrid automata to intensively gather the control requirements. Then, the platform-independent model (PIM) was specialized using the real-time UML/SysML to design the capsule collaboration of control and its connections. The detailed PIM was subsequently converted into the platform-specific model (PSM) using open-source platforms to promptly realize the AUV controller. On the basis of the proposed hybrid model, a planar trajectory-tracking controller, which allows a miniature torpedo-shaped AUV to autonomously track the desired planar trajectory, was implemented and evaluated, and shown to have good feasibility.

Keywords: autonomous underwater vehicle (AUV); AUV control; extended/unscented Kalman filter (EKF/UKF); model-based systems engineering (MBSE); model-driven architecture (MDA); real-time UML/SysML; hybrid automata

1. Introduction

Underwater vehicles have been extensively developed for many military applications in recent decades. In particular, autonomous underwater vehicles (AUVs) are of interest with respect to developing civil applications for enhancing economic effectiveness, e.g., ocean exploration, environmental monitoring, mapping, and disaster and tsunami warnings [1–8].

Controller design for AUVs has been a challenge because controllers are closely linked to AUV dynamics in complex underwater environments [9–11]. The AUV controller can consist of discrete models, continuous models, and their interaction in a hybrid dynamic system (HDS), as modeled by hybrid automata (HA) [12–15]. Traditional control methods have often been used for implementing complex systems to make them more effective for their controllers [16–18]. They have also been used for building AUV controllers. Some traditional control techniques applied for AUV applications are described below.

Lyapunov stability [19–22] was demonstrated to be very reactive. However, the stability of the desired waypoint was not suitable enough to track horizontal planar trajectories. The proportional–integral–derivative (PID) regulator [23–26] proved to be well suited for use in AUVs when tracking horizontal planar trajectories. It was possible to successfully perform the first autonomous trip using this method. Nevertheless, the PID controller was implemented to control the AUV in the absence of large disturbances. The linear quadratic (LQ) [27,28] controller presented average stabilization results. Backstepping methods [29,30] were shown to be able to control the Euler roll, pitch, and yaw (RPY) angles in conditions of high environmental noise. The sliding-mode controller (SMC) [31–33] did not give good results when applied alone, as it seemed to be short of adaptation for the dynamics of AUVs. Hence, in some studies investigating backstepping [34,35], neural networks [36–39], the computed torque method [40,41], and digital filters such as extended/unscented Kalman filters (EKF/UKF), the SMC has been improved by using control techniques to improve its performance for AUVs.

The above assessment led to us choosing a combination of PID and backstepping to perform a continuous model evolution of the AUV controller, called the integral backstepping (IB) technique.

Reusability must also be considered in the development of new AUV applications with respect to their lifecycle in an effort to reduce their cost and resources. The Object Management Group (OMG) [42] standardized the unified modeling language (UML), which is as an industry standard used to visualize, specify, construct, and document the artefacts of a software-intensive system. The system modeling language (SysML) [43] was standardized by the OMG for systems engineering. SysML is a UML profile that can provide simple but powerful constructs for modeling a wide range of systems engineering problems. However, the drawback of UML and SysML is that they lack the ability to model the evolution of internal continuous behavior for developed systems.

On the other hand, the model-based systems engineering (MBSE) approach was formalized by INCOSE [44,45] to robustly model whole artefacts in the development lifecycle of unintelligible systems. Examples of systems engineering methods [46] were identified in a survey of MBSE methodologies [47], including *Harmony* for systems engineering (*Harmony*-SE) [48,49], the object-oriented systems engineering method (OOSEM) [50,51], the rational unified process for systems engineering (RUP-SE) [52], the state analysis method [53], and the object process methodology (OPM) [54,55]. The model-driven architecture (MDA) [56,57] was standardized by the OMG for separating the specification of system operations from the details of how a system uses the capabilities of its platform. The three main goals of MDA are portability, interoperability, and reusability through an architectural separation of concerns. Here, portability allows the same solution to be realized on new or multiple platforms, while interoperability creates systems that can easily integrate and communicate with other systems and use a variety of resource applications, and reusability builds solutions that can be reused in many different applications in different contexts [56]. Sebastián et al. [58] investigated MDA applications by conducting a systematic mapping of MDA literature in software engineering between 2008 and 2018. Actually, the principle of MDA can be used within the unified architecture framework (UAF) [59] to strengthen the interoperability of a system. In many commercial applications, real-time SysML/UML has been combined with the above model-based methods for systems engineering [57,60–67]. Hence, the MBSE approach and the features of MDA can be used in combination with real-time UML [68–71] and SysML (for example, real-time UML/SysML) to describe, in detail, the artefacts of the developed system.

On the basis of the above-assessed points, this work focuses on the construction of a hybrid control model based on the MBSE methodology, in combination with the MDA concept, real-time UML/SysML, and HA, permitting us to intensively realize an AUV controller. The control artefacts designed can be customized and reused for deployment on various AUV platforms. In this study, the dynamic models of AUV for control were combined with the specialization of MDA features, composed of the platform-specific model (PSM), platform-independent model (PIM), and computation-independent model (CIM). Lastly, a planar trajectory-tracking controller for a miniature torpedo-shaped autonomous underwater vehicle running on a free surface was deployed and evaluated through simulation experiments.

The three main contributions of this research are as follows:

(1) The MBSE methodology, together with MDA components, was adapted for usability in the lifecycle development of AUV controllers.
(2) The designed control capsules are customizable and reusable for many kinds of AUVs.
(3) A planar trajectory-tracking controller of a miniature AUV running on the free surface was developed and evaluated through simulation experiments.

This manuscript is structured as follows. Section 2 presents the adapted dynamics and control structure of AUVs, while Section 3 proposes the details of MBSE-driven development aimed at intensively realizing AUV controllers, consisting of the CIM, PIM, and PSM components. A case study on application of the specialized model is discussed in Section 4, followed by the paper's conclusions and future prospects.

2. AUV Dynamics and Control Architecture

2.1. AUV Dynamic Model for Controlling

The six motions of the AUV are defined as sway, surge, roll, heave, yaw, and pitch by the Society of Naval Architects and Marine Engineers (SNAME [72] (Table 1).

Table 1. Society of Naval Architects and Marine Engineers (SNAME) notations for underwater vehicles (data from [64]).

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

According to the guidance, navigation, and control of underwater vehicles [9,73–77], the kinematic model in the inertial frame and the dynamic model in the main frame of AUVs can be written as Equations (1) and (2), respectively.

$$\dot{\boldsymbol{\eta}}=\boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu}, \tag{1}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}}+\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu}+\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}+\boldsymbol{g}(\boldsymbol{\eta})=\boldsymbol{\tau}(\boldsymbol{v},\ \boldsymbol{u}), \tag{2}$$

where $\boldsymbol{\eta}$ = $[\boldsymbol{\eta_1}^{\mathrm{T}},\boldsymbol{\eta_2}^{\mathrm{T}}]^{\mathrm{T}}$ consists of the position $\boldsymbol{\eta_1}$ = $[x,y,z]^{\mathrm{T}}$ and the orientation $\boldsymbol{\eta_2}$ = $[\phi,\theta,\psi]^{\mathrm{T}}$ of the vehicle expressed in the inertial frame, while $\boldsymbol{\nu}$ = $[\boldsymbol{v_1}^{T},\boldsymbol{v_2}^{\mathrm{T}}]^{\mathrm{T}}$ includes the linear $\boldsymbol{v_1}$= $[u,v,w]^{\mathrm{T}}$ and the angular $\boldsymbol{v_2}$ = $[p,q,r]^{\mathrm{T}}$ velocities of the vehicle expressed in the body frame. The model matrices $\boldsymbol{M}$, $\boldsymbol{C(\nu)}$, and $\boldsymbol{D(\nu)}$ denote inertia, Coriolis, and damping, respectively, while $\boldsymbol{g(\eta)}$ is a vector of gravity and buoyancy forces. On the right-hand side of Equation (2), $\boldsymbol{\tau(v,u)}$ is the vector of resultant forces and moments acting on the AUV, and $\mathbf{u}$ represents the control inputs.

A model of state-space discreteness can be used to model the control evolution of an AUV that is used to estimate the states of an AUV by using the EKF or UKF [78–82] methodologies; the motion of the control system can be described as shown in Equation (3).

$$\begin{cases} \mathbf{x_k} = \mathbf{f_{k-1}}(\mathbf{x_{k-1}},\ \mathbf{u_{k-1}}) +\ \mathbf{w_{k-1}} \\ \mathbf{y_k} = \mathbf{h_k}(\mathbf{x_k}) +\ \mathbf{v_k} \end{cases} \tag{3}$$

where $\mathbf{x} = \begin{bmatrix} \eta \\ \nu \end{bmatrix}$, $\mathbf{x_k}$ is the state variable vector at the k-th instant of $\mathbf{x}$, $\mathbf{u_k}$ and $\mathbf{y_k}$ are the system's inputs and outputs, respectively, and $\mathbf{h_k}$, $\mathbf{w_k}$, and $\mathbf{v_k}$ are the measurement function, additive process, and measurement noise, respectively.

2.2. General Control Architecture for an AUV

The physical architecture of an AUV consists of the following subsystems: the guidance subsystem; the navigation subsystem; and the control subsystem. These subsystems have their own tasks, yet they must also cooperate to permit the vehicle to complete its mission. Figure 1 shows a block definition diagram in SysML that depicts the interactions of the subsystems.

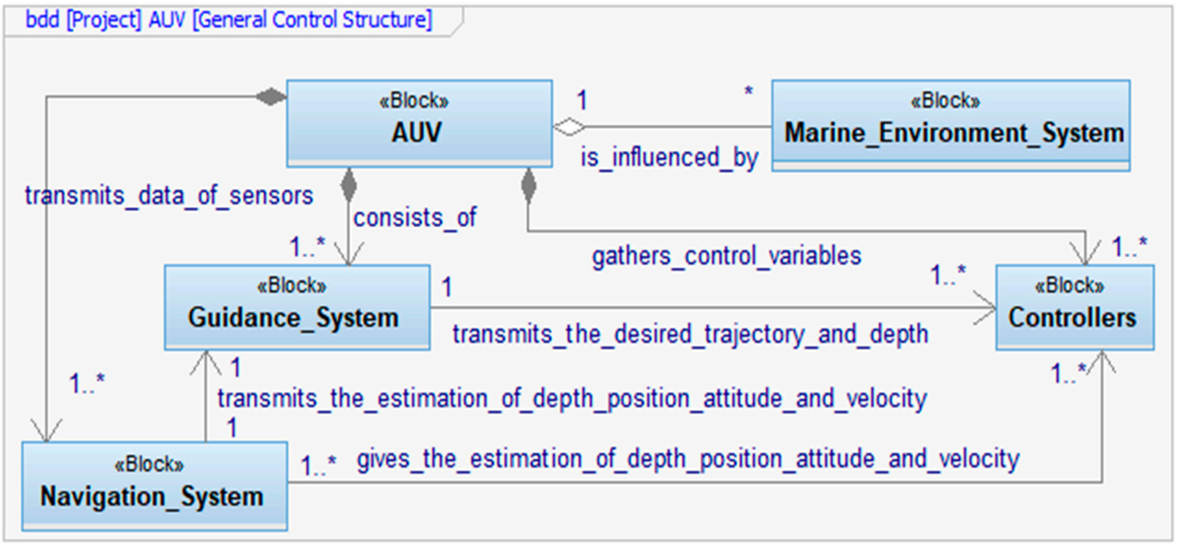

bdd [Project] AUV [General Control Structure]
«Block» AUV
«Block» Marine_Environment_System
1
*
is_influenced_by
transmits_data_of_sensors
consists_of
1..*
gathers_control_variables
1..*
«Block» Guidance_System
1
1..*
«Block» Controllers
transmits_the_desired_trajectory_and_depth
1
1..*
1..*
transmits_the_estimation_of_depth_position_attitude_and_velocity
1
«Block» Navigation_System
1..*
gives_the_estimation_of_depth_position_attitude_and_velocity

Figure 1. Autonomy architecture block definition diagram for an autonomous underwater vehicle (AUV).

According to the above AUV dynamic and control architecture, and the definition of an HDS described in [13–15], AUV controllers can, thus, be considered HDSs whose dynamic behaviors can be modeled by HA [12] and implemented by line-of-sight (LOS) navigability, as described in [83–87].

3. MBSE-Driven Development for an AUV Controller

3.1. CIM for an AUV Controller

On the basis of the dynamics and control frame of AUVs described in Section 2, the main use case model of AUV controllers is shown in Figure 2. Figure 3a,b describes a case study of path-tracking scenarios, where the state machine of the "Track a desired trajectory" use case is shown using the sequence and state diagrams of real-time UML/SysML conventions.

The system/human actors and use cases of the AUV controller are defined as follows:

- MDS is the measurement display system actor, which includes the guidance subsystem and navigation subsystem.
- MES is the marine environment system actor, which represents the marine environmental noises.
- Maintainer is a human actor who has authority to check the physical AUV components and configure system parameters AUV for running AUV tasks.
- "Track a desired trajectory" is a use case study for tracking the target of a predefined path.
- "Ensure safety" is a use case for ensuring system safety.
- "Configure control parameters of the AUV" is a use case for configuring and updating system parameters.

- "Maintain the physical components" is a use case for servicing the whole physical system.

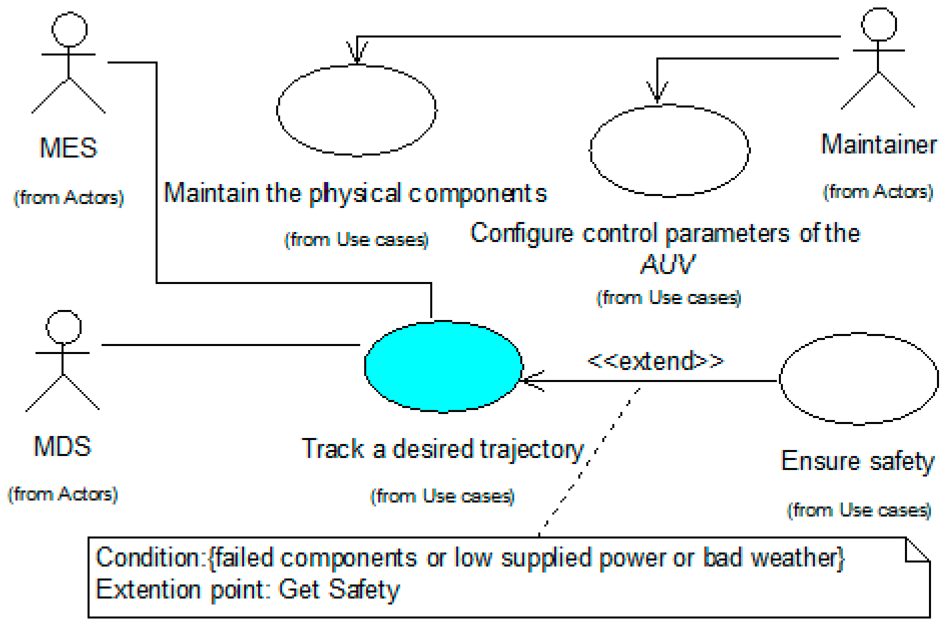
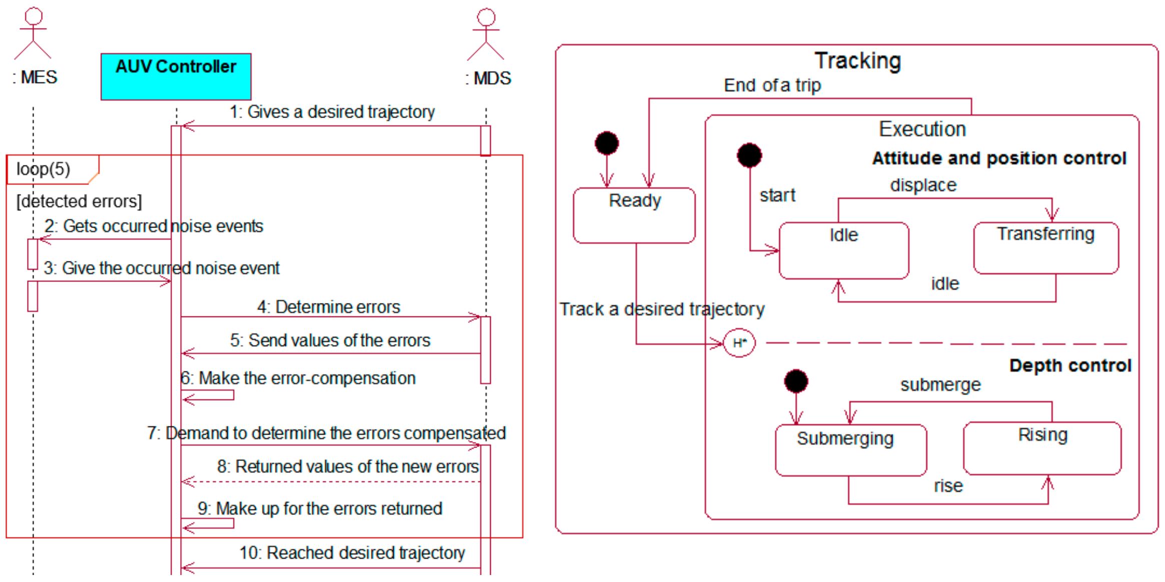

MES
(from Actors)
Maintain the physical components
(from Use cases)
Configure control parameters of the AUV
(from Use cases)
Maintainer
(from Actors)
MDS
(from Actors)
Track a desired trajectory
(from Use cases)
<<extend>>
Ensure safety
(from Use cases)
Condition:{failed components or low supplied power or bad weather}
Extention point: Get Safety

Figure 2. Use case model of the developed AUV.


: MES
AUV Controller
: MDS
1: Gives a desired trajectory
loop(5)
[detected errors]
2: Gets occurred noise events
3: Give the occurred noise event
4: Determine errors
5: Send values of the errors
6: Make the error-compensation
7: Demand to determine the errors compensated
8: Returned values of the new errors
9: Make up for the errors returned
10: Reached desired trajectory
Tracking
End of a trip
Execution
Attitude and position control
displace
start
Ready
Idle
Transferring
idle
Track a desired trajectory
H*
Depth control
submerge
Submerging
Rising
rise

(**a**) (**b**)

Figure 3. (**a**) Desired trajectory-tracking scenario, and (**b**) local state machine for performing the "*track a desired trajectory*" use case.

In this work, an implemented functional block diagram (Figure 4) is proposed for the kinematic and dynamic models of an AUV, described in Equations (1) and (2), to obtain the internal continuous evolutions for the controller, where Ω_{di}, $i = \overline{1,n}$ are desired rotational speeds, which are applied to the n actuators of the AUV, and ΣT and $\tau_{\phi,\theta,\Psi}$ are the overall output forces and moments acting on the actuators of the AUV.

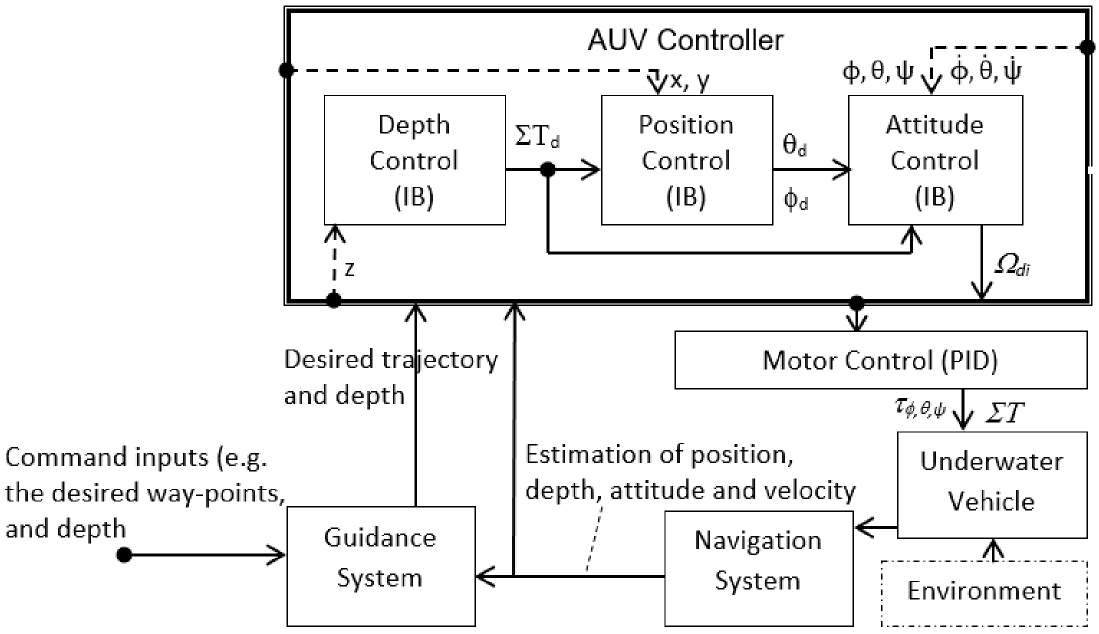


Figure 4. Functional block diagram for implementing the continuous evolution of an AUV controller.

As previously assessed, the IB expansion combined with the control Lyapunov function (CLF) can be used in many AUV control applications. This was also applied to the functional blocks of deep control, position control, and attitude control (Figure 4), which participate in the continuous evolutions. PID regulators were also used for the functional block of motor control. This study did not focus on the decomposition of these control techniques for an AUV because they were developed in many AUV applications [23–26,34,35,88–90].

In addition, the discrete state-space models in Equation (3), in combination with the EKF or UKF [78–82] implementations, allowed the estimation of the states of the developed AUV, as introduced in Section 4.

Furthermore, hybrid automata (HA), presented by Henzinger, Kopke, Puri, and Varaiya [12], provide a mathematical model for digital computer systems that interact with an analog environment in real time. In the CIM, HA are established as shown in Equation (4).

$$H_{AUV} = (Q, X, \Sigma, A, Inv, F, q_o, x_{co}), \tag{4}$$

where Q is a set of running cases of the AUV, $q_o \in Q$ is the starting situation, X is the continuous state-space of continuous elements, $x_{co} \in X$ is the initial value, Σ is a set of external events, A is a set of transitions between running cases corresponding to events $\sigma \in \Sigma$, Inv is an application tool, which is used to check $x_c \in inv(q)$, and F is the continuous global model issued from the kinematic and dynamic models in Equations (1) and (2).

3.2. PIM for an AUV Controller

The PIM's goal is to implement real-time capsule collaboration, which allows capturing, in detail, the design model for control. From the above-identified CIM, the five primary control capsules were specialized to implement the HA for an AUV controller: the discrete part's capsule, the continuous part's capsule, the external interface's capsule, the internal interface's capsule, and the instantaneous global continuous behavior (IGCB)'s capsule. Figures 5 and 6 indicate the real-time capsule collaboration for an AUV controller using real-time UML cooperation and class diagrams.

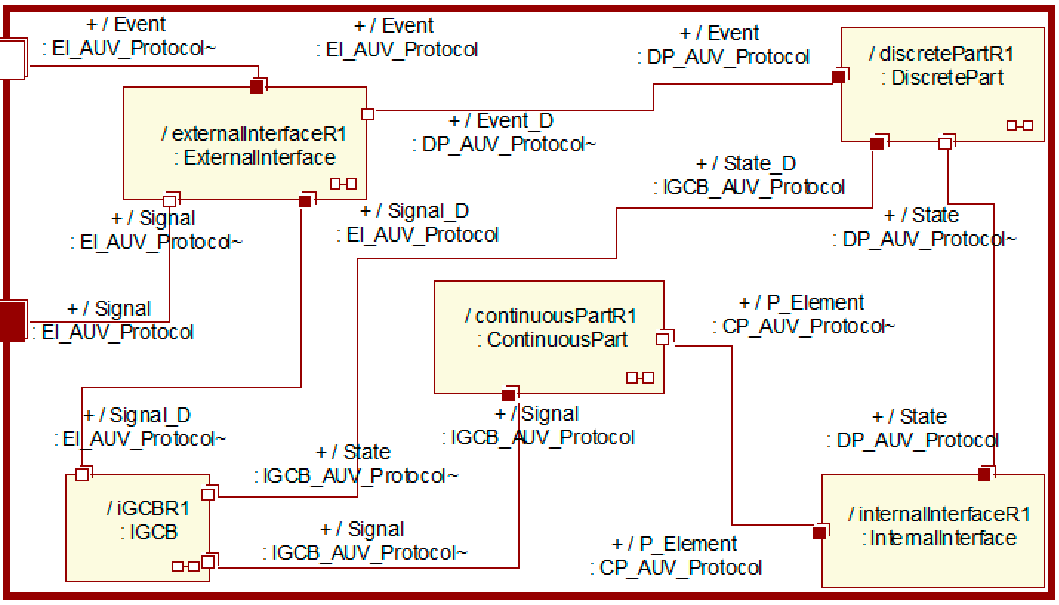
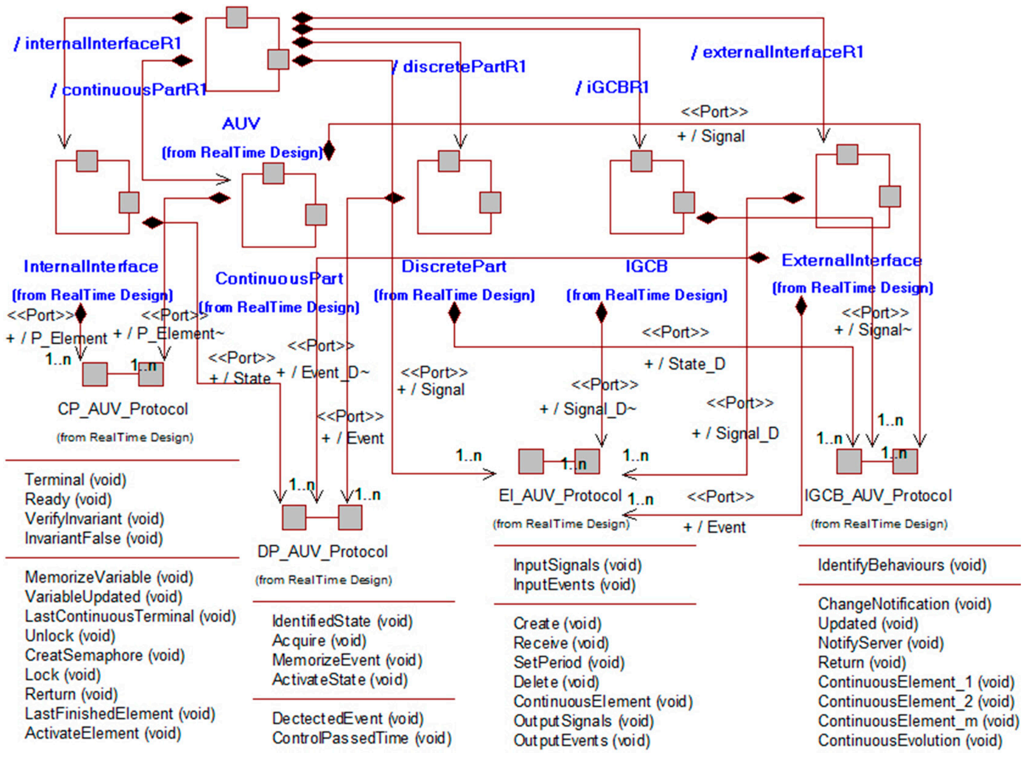


Figure 5. Collaboration diagram of real-time capsules for an AUV controller (data from [63]).

Here, the discrete part's capsule consists of situations ***Q*** and transitions ***A*** in HA of the AUV controller; the continuous part's capsule contains the continuous state-space ***X***; the IGCB's capsule implements concrete global continuous behaviors as ***f*∈*F***, where ***f*** is directly derived from Equation (3) and the implemented functional block diagram (Figure 4) can be implemented in ***f*** for the estimation of AUV states; the external interface's capsule is an intermediary, which receives/sends events/signals between the AUV controller and the MES/MDS; the internal interface's capsule permits the ***Inv*** tool to generate internal events in the HA evolution. The detailed specification of this capsule collaboration can be found in the author's previous report [63].



CP_AUV_Protocol	DP_AUV_Protocol	EI_AUV_Protocol	IGCB_AUV_Protocol
Terminal (void) Ready (void) VerifyInvariant (void) InvariantFalse (void)	IdentifiedState (void) Acquire (void) MemorizeEvent (void) ActivateState (void)	InputSignals (void) InputEvents (void)	IdentifyBehaviours (void)
MemorizeVariable (void) VariableUpdated (void) LastContinuousTerminal (void) Unlock (void) CreatSemaphore (void) Lock (void) Rerturn (void) LastFinishedElement (void) ActivateElement (void)	DectectedEvent (void) ControlPassedTime (void)	Create (void) Receive (void) SetPeriod (void) Delete (void) ContinuousElement (void) OutputSignals (void) OutputEvents (void)	ChangeNotification (void) Updated (void) NotifyServer (void) Return (void) ContinuousElement_1 (void) ContinuousElement_2 (void) ContinuousElement_m (void) ContinuousEvolution (void)

Figure 6. Class diagram of real-time capsules for an AUV controller (data from [63]).

Reusability is essential in the operator of controllers for different AUV applications because it reduces manufacturing time and equipment costs. Moreover, this can allow the capsule collaboration of a developed AUV to be customized and reused in a new control application for many types of AUVs, as shown in Table 2.

Table 2. The customizability and reusability of designed control capsules in new control applications for many types of AUVs. IGCB, instantaneous global continuous behavior.

Designed Control Capsules	**Specialization Rules**	
	Generic Artifacts the New AUV Controller	**Specialized Artifacts the New AUV Controller**
IGCB	The state machine, ports, and protocols of this capsule are not changed.	The specifications of the IGCB's capsule make up the new IGCB model and are formed by the new continuous components.
Continuous part	The ports and protocols of this capsule are not changed.	It is specialized by adding or removing down continuous elements.
Discrete capsule	This is not changed.	None.
External interface	The state machine, ports, and protocols of this capsule are not changed.	It is specialized by adding/removing inputs/outputs events issued from the outside.
Internal interface	The state machine and ports of this capsule are not changed.	It is specialized by adding/removing ***Inv*** in/from the new IGCB.

The real-time capsule collaborations shown in Figures 5 and 6 are not changed for new control applications of AUVs.

3.3. PSM for an AUV Controller

In the construction of the AUV controller, the above-designed PIM was converted into the PSM using IBM Rational Software Architect Real Time, IBM Rational Rose Real Time [91], or Papyrus for Real Time (Papyrus-RT) [92]. These tools are effectively used to develop complex real-time and embedded systems and software applications. They act as implementations of real-time UML/SysML for C++, Java, Ada, and runtime system supports.

Hence, the PIM could be converted into the PSM using different implementation development environments (IDEs) to ultimately realize a controller with suitable microcontrollers. The MDA's features also support model transformation. This transformation model could be rapidly applied through round-trip engineering. The transformation rules, which can be used to convert the PIM into PSM and vice versa through round-trip engineering of the intermediate codes of an object-oriented programming language, were presented in the authors' previous report [1].

Furthermore, the above-defined HA could be automatically implemented using the state pattern described in [93,94]. According to this pattern, the HA's structure implementation to display the meaningful programming usefulness of the control program of an AUV is shown in Figure 7. An example of HA implementation based on the state pattern was performed and compiled using Arduino's IDE [95] to fit into ATMEGA32-U2 and STM32 Cortex-M4 microcontrollers for an AUV controller, as shown in Appendix A.

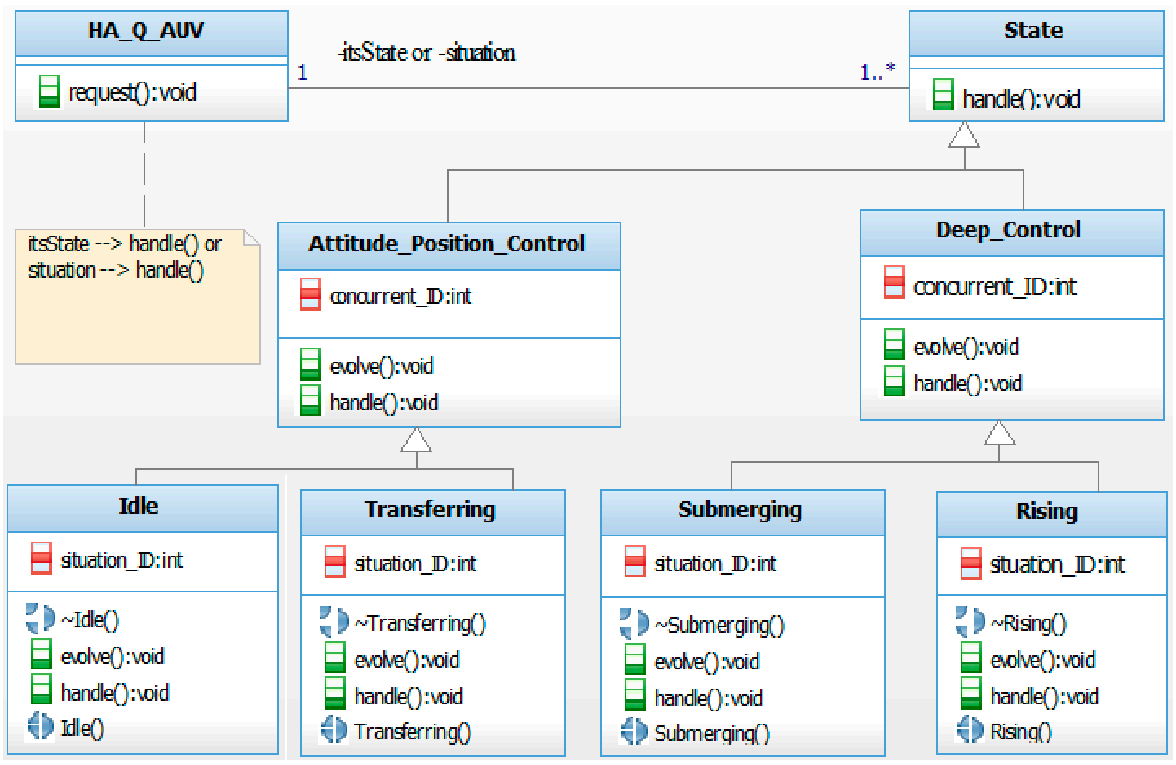


Figure 7. State pattern structure of hybrid automata (HA) for the AUV controller.

4. Application

4.1. Physical Application Configurations

Following the above-proposed model, a planar trajectory-tracking controller, which allowed a low-cost AUV possessing a torpedo shape to reach and follow a predetermined trajectory on the free surface, was deployed. This case study represents one element of our long-term research project funded by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 107.03-2019.302. The torpedo-shaped AUV's main operating parameters are summarized in Table 3.

Table 3. The torpedo-shaped AUV's main operating parameters (data from [63]).

Parameters	Values
Size (L × H × W)	(1.50 × 0.20 × 0.20) m
Weight	11.50 kg
Autonomous duration	25 min
2× Li–Po battery	22.2 V, 20,000 mAh
Ultimate capacity	285 W
Maximum submersing/rising speed	0.70 m/s
Maximum horizontal moving speed	1.80 m/s
Maximum operation depth	1.20 m
Maximum radius of operation	400 m
Inertia moment on x-axis, I_{xx}	0.057 kg·m^2
Inertia moments on y-axis and z-axis, $I_{yy} = I_{zz}$	1.271 kg·m^2

4.2. Control Implementation and Test Results

According to the functional block diagram for performance describing the continuous evolution of the AUV controller, the environmental disturbance caused by a wave was only considered as sea state code 1 [96], i.e., slight ripples on the free surface.

The state-space models shown in Equation (3) were implemented to calculate the current states of the AUV using the installed sensors, e.g., the inertial measurement unit (IMU) MPU6000 [97] and the global positioning system (GPS) Ublox Neo 6M [98]. The state estimations in both cases were based on the EKF (Algorithm 1) and the UKF (Algorithm 2). In Algorithms 1 and 2, $\hat{\ }$ denotes an estimation, P is the state covariance, and Q and R represent the covariance matrices of the process and measurement noise, respectively. The state was estimated starting from the following initial conditions: $\hat{\mathbf{x}}_{0|0} = \mathbf{x}_0$ and $P_{0|0} = 0_{12\times12}$.

Algorithm 1. Navigation filter based on the extended Kalman filter (EKF).

```
Function EKF algorithm
Step EKF predict
Data: $\hat{\mathbf{x}}_{k-1|k-1}, P_{k-1|k-1}, \mathbf{f}_{k-1}(.)$
Result: $\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}$
$F_{k-1} = \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k-1|k-1}\mathbf{u}_{k-1}};$
        $\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}_{k-1}\left(\hat{\mathbf{x}}_{k-1|k-1}\right);$
        $P_{k|k-1} = F_{k-1}, P_{k-1|k-1}F^T_{k-1} + Q_{k-1};$
end
Step EKF update
Data: $\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}, \mathbf{h}_k(.)$
Result: $\hat{\mathbf{x}}_{k|k}, P_{k|k}$
$H_k = \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k|k-1}};$
        $S_k = R_k + H_k P_{k|k-1} H^T_k;$
        $L_k = P_{k|k-1} H^T_k S^{-1}_k;$
        $\mathbf{e}_k = \mathbf{y}_k - \mathbf{h}_k\left(\hat{\mathbf{x}}_{k|k-1}\right);$
        $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + L_k \mathbf{e}_k;$
        $P_{k|k} = P_{k|k-1} - L_k S_k L^T_{k-1};$
end
```

Algorithm 2. Navigation filter based on the unscented Kalman filter (UKF).

```
Function UKF algorithm
Step UKF predict
    Data: $\hat{\mathbf{x}}_{k-1|k-1}, P_{k-1|k-1}, \mathbf{f}_{k-1}(.)$
    Result: $\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}$
    $\left(\hat{\mathbf{x}}_{k|k-1}, \bar{P}_{k|k-1}\right) = UT\left(\hat{\mathbf{x}}_{k-1|k-1}, \bar{P}_{k-1|k-1}, \mathbf{f}_{k-1}(.)\right);$
    $P_{k|k-1} = \bar{P}_{k|k-1} + Q_{k-1};$
end
Step UKF update
Data: $\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}, \mathbf{h}_k(.)$
Result: $\hat{\mathbf{x}}_{k|k}, P_{k|k}$
        $\left(\hat{\mathbf{y}}_{k|k-1}, \bar{S}_k, P^{xy}_k\right) = UT\left(\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}, \mathbf{h}_{k-1}(.)\right);$
        $S_k = R_k + \bar{S}_k;$
        $L_k = P^{xy}_k S^{-1}_k;$
        $\mathbf{e}_k = \mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1};$
        $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + L_k \mathbf{e}_k;$
        $P_{k|k} = P_{k|k-1} - L_k S_k L^T_k;$
end
```

We used the *OpenModelica* tool [99], which is an open-source simulation environment, to perform the simulation of an AUV controller. *OpenModelica* is an object-oriented modeling environment of *Modelica* [100] and C/C++ for hybrid systems. A case study in which the MDS was assumed to address an event in the transferring state to the AUV controller with a desired course angle of 020° and average speed of 1.5 m/s is shown in Figure 8. Here, the average transient durations, which correspond to the cases using EKF and UKF, were 6.8 and 6.2 s for the AUV's stabilized course.

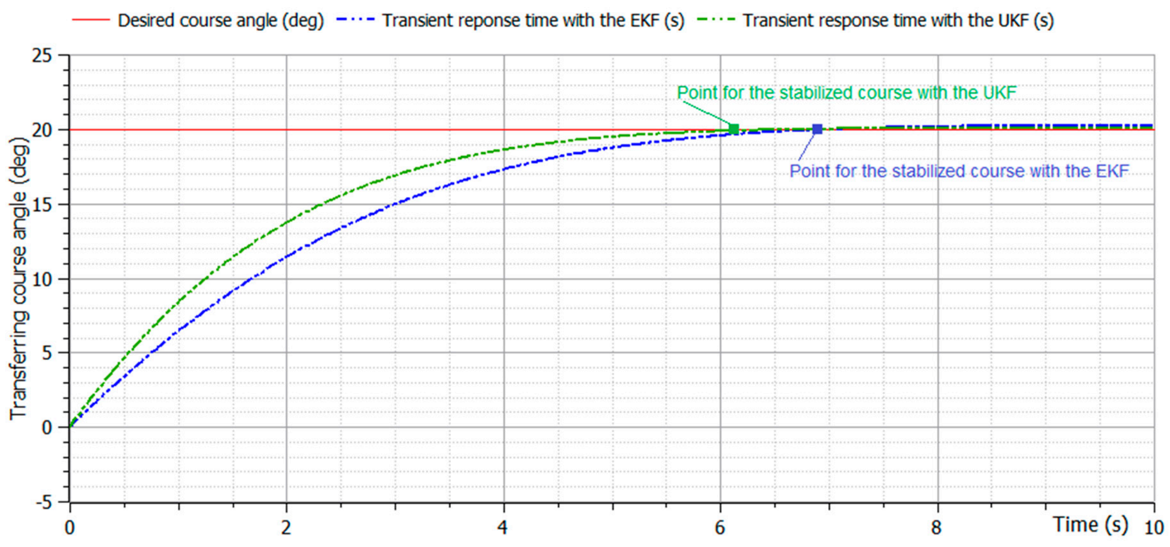


Figure 8. Average transient response time in a desired course of 020° from the current position for the cases using EKF and UKF.

The ATMEGA32-U2 and STM32 Cortex-M4 microcontrollers [95] were installed on the mainboard. The AUV installation for trial trips is shown in Figure 9. The test scenarios were based on different desired courses, for various desired shape-based paths and average velocities. Some of the main planar course-tracking test results are shown in Table 4. Figure 10a,b and Figure 11a,b respectively show that the AUV reached and followed the desired rectangle- and triangle-shaped trajectories.

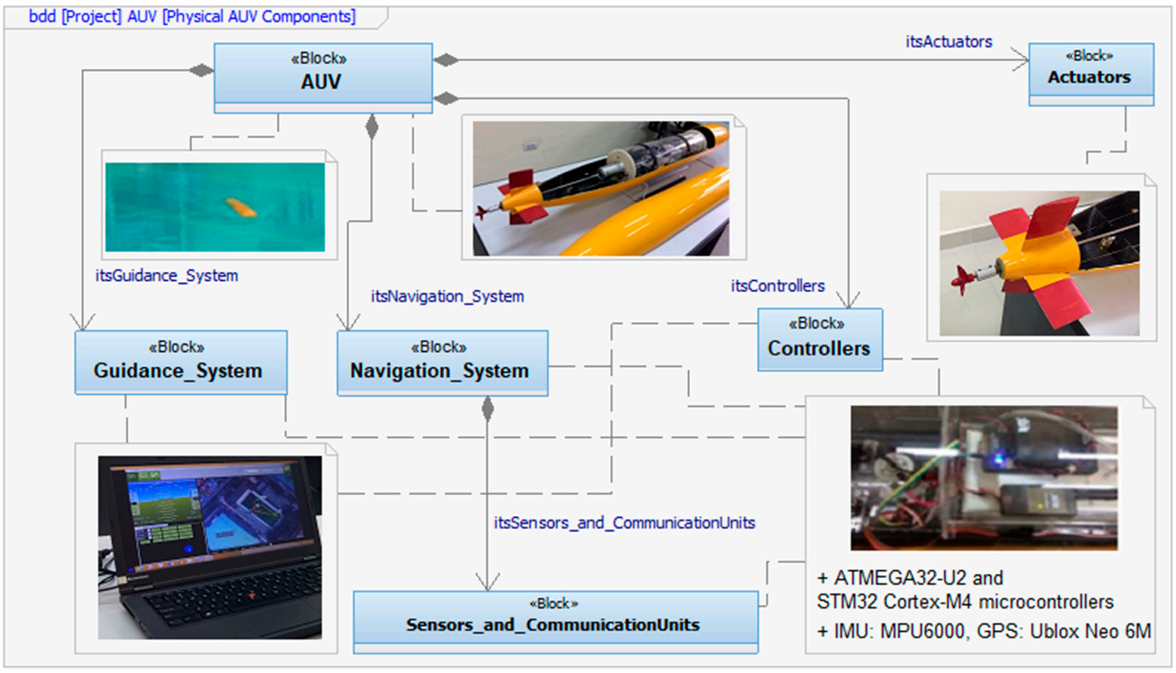

bdd [Project] AUV [Physical AUV Components]
«Block» AUV
itsActuators
«Block» Actuators
itsGuidance_System
itsNavigation_System
itsControllers
«Block» Guidance_System
«Block» Navigation_System
«Block» Controllers
itsSensors_and_CommunicationUnits
«Block» Sensors_and_CommunicationUnits
+ ATMEGA32-U2 and STM32 Cortex-M4 microcontrollers
+ IMU: MPU6000, GPS: Ublox Neo 6M

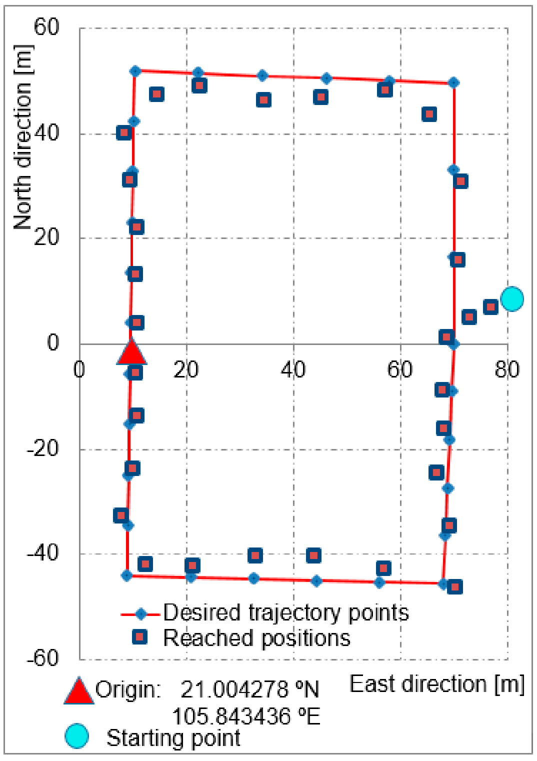
Figure 9. AUV installation for trial trips.

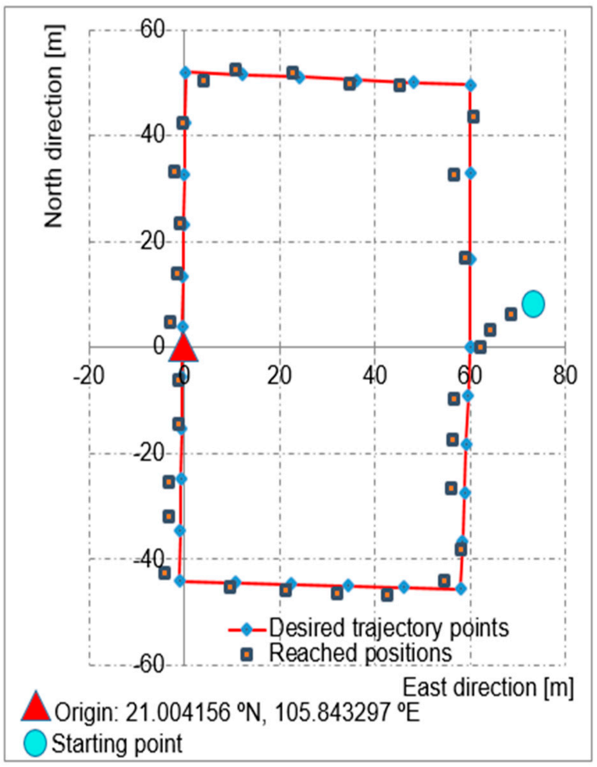
Table 4. Testing results for the course-tracking of the torpedo-shaped AUV.

No.	Desired Course Angle (°)	Average Velocity (m/s)	Stabilized Interval (s), (with the EKF)	Stabilized Interval (s), (with the UKF)
1	010	0.5	7.1	6.4
2	010	1.5	5.7	5.2
3	020	0.5	7.6	7.1
4 *	020	1.5	6.9	6.2
5	030	0.5	9.3	8.8
6	030	1.5	8.3	7.9

* This test scenario corresponds to the simulation case shown in Figure 8.

On the basis of comparison with the test results obtained in the literature [1,63], this current AUV controller was superior in terms of the stabilized interval and trajectory error, which decreased by about 0.7 s and 0.90 m, respectively. The UKF enabled more accurate estimations. Although operations in UKF such as the unscented transform (UT), i.e., the *UT* function in Algorithm 2, may appear more complex than those for the EKF, the assessments of actual computational complexity and optimizations for the application of various Kalman filter extensions were studied intensively by Zhang, et al. [101] and Raitoharju and Piché [102].





(**a**) (**b**)

Figure 10. The AUV reached and followed a desired rectangle-shaped trajectory: (**a**) with the extended Kalman filter (EKF) algorithm; (**b**) with the unscented Kalman filter (UKF) algorithm.

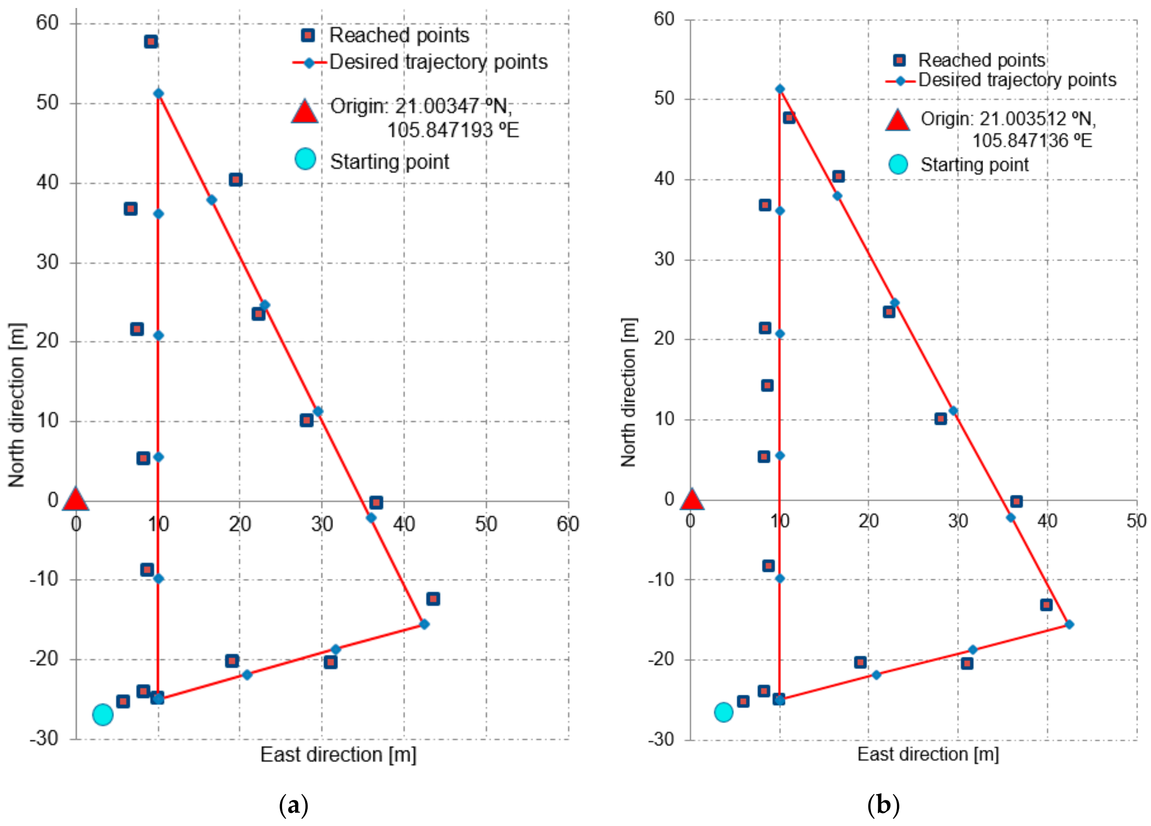


Figure 11. The AUV reached and followed a desired triangle-shaped trajectory: (**a**) with the EKF algorithm; (**b**) with the UKF algorithm.

An assessment of the above-described AUV application using MBSE methodology combined with MDA components is described in Table 5.

Table 5. Assessment of the torpedo-shaped AUV control application of model-based systems engineering (MBSE) methodology combined with the model-driven architecture (MDA). CIM, computation-independent model; PIM, platform-independent model; PSM, platform-specific model; IDE, implementation development environments; OMG, Object Management Group; XML, extensible markup language; MOF, Meta Object Facility; UML, unified modeling language; SysML, systems modeling language.

Proposed Models	Advantages	Disadvantages
CIM	This model focuses on a global model of top level, which can combine discrete models and continuous models.	An implemented functional block diagram must be supplemented in the CIM to depict internal continuous behaviors for the control system developed.
PIM	The PIM–PSM separation and its model transformation allow the designed control elements to be customizable and reusable for various kinds of AUVs.	This can influence the performance effort of projects.
PSM	The control capsules can be transformed into various PSM IDEs (e.g., Java, Net, or Ada IDEs).Arduino microcontrollers are used to deploy the real-time and embedded control system using open-source solutions.	Within the OMG, the XML Metadata Interchange (XMI) specification [103] supports the exchange of model data when using an MOF-based language such as real-time UML/SysML. However, development engineers may need training to develop the required skills in different IDEs.

5. Conclusions and Future Work

This paper introduced an application of MBSE methodology to intensively deploy controllers for AUVs whose dynamics can be considered an HDS. This application model is based on the MBSE methodology, combined with MDA concepts, real-time UML/SysML, EKF/UKF algorithms, and HA to systematically realize the controller. The dynamic models and control structure of AUV were first used for control combined with MDA components such as the CIM, PIM, and PSM. In the CIM, the use case model was defined with continuous behaviors, EKF/UKF algorithms, and HA to closely control the requirements. The PIM was established to establish the design model by constructing a real-time capsule pattern. This pattern can be customized and reused in new AUV control applications (Table 2). The PIM designed was then converted into the PSM through round-trip engineering of the intermediate C++ codes to form an AUV controller with suitable microcontrollers. On the basis of the proposed model, a planar trajectory-tracking controller of a miniature torpedo-shaped AUV running on the free surface was implemented and evaluated using the ATMEGA U2 and STM32-Cortex-M4 microcontrollers. Lastly, the advantages and disadvantages of the MBSE/MDA approach were discussed with respect to this AUV control application (Table 5).

In this case study, the above-described MBSE methodology, combined with MDA concepts, was only applied to simple test scenarios for a miniature torpedo-shaped AUV running on the free surface. We are yet to fine-tune parameters with respect to the process and measurement noise and the evolutionary optimization for noise parameters for this application. Thus, these important further developments are scheduled for the future. Firstly, the EKF/UKF-based navigation filters will be simulated online within a complete AUV combined with depth control and a suitable environment. Then, the new controller will be implemented on the AUV and tested online through fast frequency disturbances. The performances of the different Kalman filter extensions in terms of accuracy will be carefully investigated in different scenarios. In further MBSE/MDA studies, we will also follow our application strategy to specify, in detail, the patterns of model transformations using the different MDA transformation types, and we will compare them to cases using OPM, such as those described in [54,55].

Author Contributions: Conceptualization, N.V.H. (Ngo Van Hien), N.V.H. (Ngo Van He), V.-T.T. and N.-T.B.; methodology, N.V.H. (Ngo Van Hien), N.V.H. (Ngo Van He), V.-T.T. and N.-T.B.; software, N.V.H. (Ngo Van Hien), N.V.H. (Ngo Van He), and V.-T.T.; validation, N.V.H. (Ngo Van Hien), N.V.H. (Ngo Van He), V.-T.T. and N.-T.B.; writing—original draft preparation, N.V.H. (Ngo Van Hien), N.V.H. (Ngo Van He), and V.-T.T.; writing—review and editing N.V.H. (Ngo Van Hien), N.V.H. (Ngo Van He), V.-T.T. and N.-T.B.; visualization, N.V.H. (Ngo Van Hien), N.V.H. (Ngo Van He), V.-T.T. and N.-T.B.; supervision, N.V.H. (Ngo Van Hien); project administration, N.V.H. (Ngo Van He); funding acquisition, N.-T.B. All authors have read and agreed to the published version of the manuscript.

Funding: Vietnam National Foundation for Science and Technology Development (NAFOSTED).

Acknowledgments: This research was funded by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 107.03-2019.302. The authors would like to warmly express their thanks for the support.

Conflicts of Interest: The authors declare no conflict of interest.

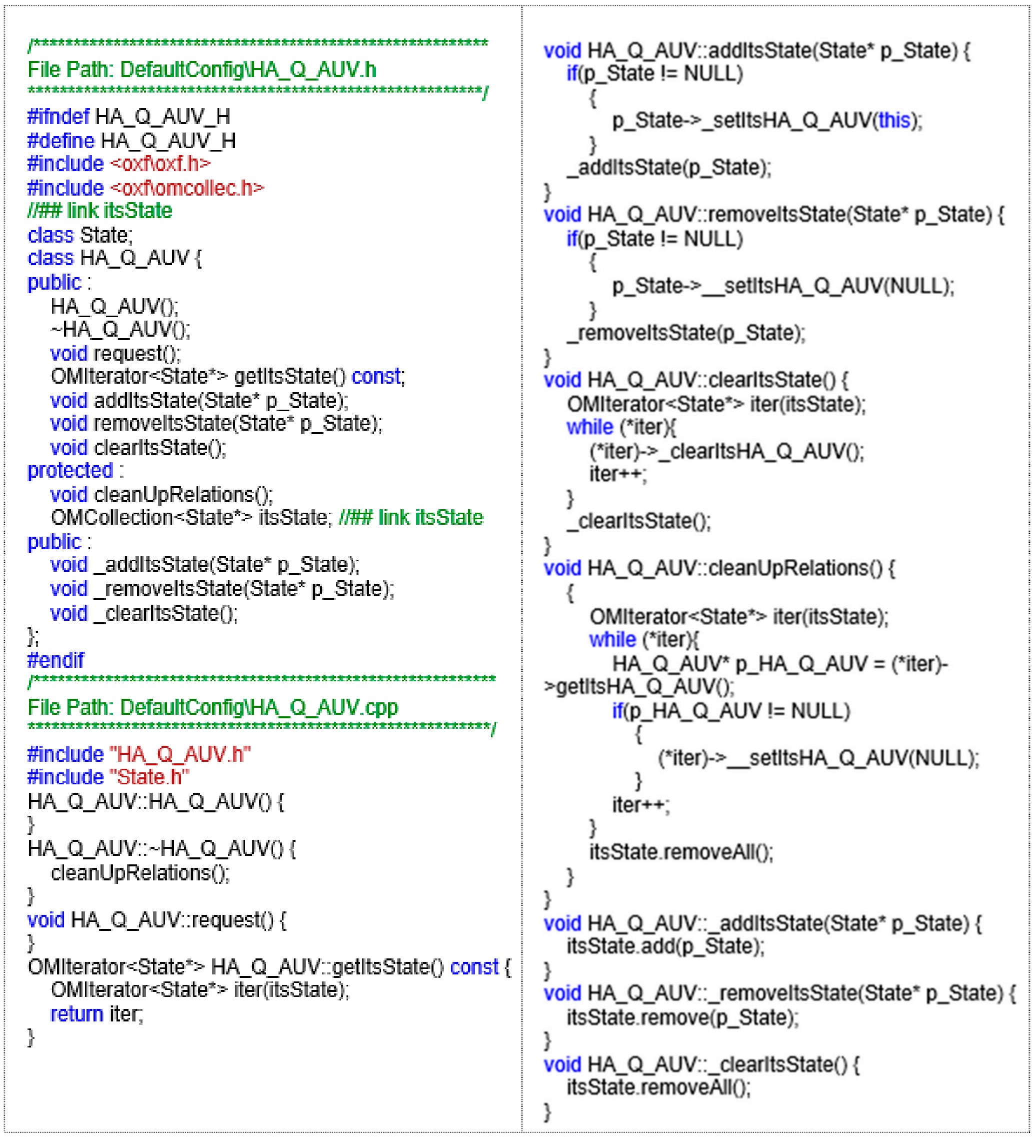
Abbreviations

AUV	Autonomous underwater vehicle	MES	Marine environment system
CIM	Computation independent model	MOF	Meta Object Facility
CLF	Control Lyapunov functions	OMG	Object Management Group
DoF	Degrees of freedom	OOSEM	Object-Oriented Systems Engineering Method
EKF	Extended Kalman filter	OPM	Object Process Methodology
GPS	Global positioning system	PID	Proportional–integral–derivative
HA	Hybrid automata	PIM	Platform independent model
Harmony-SE	*Harmony* for systems engineering	PSM	Platform specific model
HDS	Hybrid dynamic system	RPY	Roll, pitch, and yaw

IB	Integral backstepping	RUP-SE	Rational Unified Process for Systems Engineering
INCOSE	International Council on Systems Engineering	SMC	Sliding-mode control
IDE	Implementation development environment	SNAME	Society of Naval Architects and Marine Engineers
IGCB	Instantaneous global continuous behavior	SysML	Systems modeling language
IMU	Inertial measurement unit	UAF	Unified architecture framework
LQ	Linear quadratic	UKF	Unscented Kalman filter
LOS	Line-of-sight	UML	Unified modeling language
MBSE	Model-based systems engineering	UT	Unscented transform
MDA	Model-driven architecture	XMI	XML metadata interchange
MDS	Measurement display system	XML	Extensible markup language

Appendix A

An example of HA implementation based on the state pattern (Figure 7) and C++ codes is shown as Figure A1.



```
/*************************************************************
File Path: DefaultConfig\HA_Q_AUV.h
*************************************************************/
#ifndef HA_Q_AUV_H
#define HA_Q_AUV_H
#include <oxf\oxf.h>
#include <oxf\omcollec.h>
//## link itsState
class State;
class HA_Q_AUV {
public :
    HA_Q_AUV();
    ~HA_Q_AUV();
    void request();
    OMIterator<State*> getItsState() const;
    void addItsState(State* p_State);
    void removeItsState(State* p_State);
    void clearItsState();
protected :
    void cleanUpRelations();
    OMCollection<State*> itsState; //## link itsState
public :
    void _addItsState(State* p_State);
    void _removeItsState(State* p_State);
    void _clearItsState();
};
#endif
/*************************************************************
File Path: DefaultConfig\HA_Q_AUV.cpp
*************************************************************/
#include "HA_Q_AUV.h"
#include "State.h"
HA_Q_AUV::HA_Q_AUV() {
}
HA_Q_AUV::~HA_Q_AUV() {
    cleanUpRelations();
}
void HA_Q_AUV::request() {
}
OMIterator<State*> HA_Q_AUV::getItsState() const {
    OMIterator<State*> iter(itsState);
    return iter;
}
```

```
void HA_Q_AUV::addItsState(State* p_State) {
    if(p_State != NULL)
        {
            p_State->_setItsHA_Q_AUV(this);
        }
    _addItsState(p_State);
}
void HA_Q_AUV::removeItsState(State* p_State) {
    if(p_State != NULL)
        {
            p_State->__setItsHA_Q_AUV(NULL);
        }
    _removeItsState(p_State);
}
void HA_Q_AUV::clearItsState() {
    OMIterator<State*> iter(itsState);
    while (*iter){
        (*iter)->_clearItsHA_Q_AUV();
        iter++;
    }
    _clearItsState();
}
void HA_Q_AUV::cleanUpRelations() {
    {
        OMIterator<State*> iter(itsState);
        while (*iter){
            HA_Q_AUV* p_HA_Q_AUV = (*iter)-
>getItsHA_Q_AUV();
            if(p_HA_Q_AUV != NULL)
                {
                    (*iter)->__setItsHA_Q_AUV(NULL);
                }
            iter++;
        }
        itsState.removeAll();
    }
}
void HA_Q_AUV::_addItsState(State* p_State) {
    itsState.add(p_State);
}
void HA_Q_AUV::_removeItsState(State* p_State) {
    itsState.remove(p_State);
}
void HA_Q_AUV::_clearItsState() {
    itsState.removeAll();
}
```

Figure A1. An example of HA implementation based on the state pattern.

References

1. Hien, N.V.; He, N.V.; Truong, V.T.; Diem, P.G. *Specifying the Model-Driven Architecture and Real-Time Unified Modeling Language to Implement an AUV Controller. Research Project Report, Funded by State of Vietnam, KC03.TN05/11-15*; Hanoi University of Science and Technology: Hanoi, Vietnam, 2013.
2. Sivčev, S.; Coleman, J.; Omerdić, E.; Dooly, G.; Toal, D. Underwater manipulators: A review. *Ocean Eng.* **2018**, *163*, 431–450. [CrossRef]
3. Wynn, R.B.; Huvenne, V.A.I.; Bas, T.P.L.; Murton, B.J.; Connelly, D.P.; Bett, B.J.; Ruhl, H.A.; Morris, K.J.; Peakall, J.; Parsons, D.R.; et al. Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience. *Mar. Geol. Int. J. Mar. Geol. Geochem. Geophys.* **2014**, *352*, 451–468. [CrossRef]
4. Petillot, Y.R.; Antonelli, G.; Casalino, G.; Ferreira, F. Underwater Robots: From Remotely Operated Vehicles to Intervention-Autonomous Underwater Vehicles. *IEEE Robot. Autom. Mag.* **2019**, *26*, 94–101. [CrossRef]
5. Han, M.; Lyu, Z.; Qiu, T.; Xu, M. A Review on Intelligence Dehazing and Color Restoration for Underwater Images. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 1820–1832. [CrossRef]
6. Bao, J.; Li, D.; Qiao, X.; Rauschenbach, T. Integrated navigation for autonomous underwater vehicles in aquaculture: A review. *Inf. Process. Agric.* **2020**, *7*, 139–151. [CrossRef]
7. Eiler, J.H.; Grothues, T.M.; Dobarro, J.A.; Shome, R. Tracking the Movements of Juvenile Chinook Salmon Using an Autonomous Underwater Vehicle under Payload Control. *Appl. Sci.* **2019**, *9*, 2516. [CrossRef]
8. Sheng, M.; Tang, S.; Qin, H.; Wan, L. Clustering Cloud-Like Model-Based Targets Underwater Tracking for AUVs. *Sensors* **2019**, *19*, 370. [CrossRef]
9. Sabet, M.T.; Daniali, H.M.; Fathi, A.; Alizadeh, E. Identification of an Autonomous Underwater Vehicle Hydrodynamic Model Using the Extended, Cubature, and Transformed Unscented Kalman Filter. *IEEE J. Ocean. Eng.* **2018**, *43*, 457–467. [CrossRef]
10. Gibson, S.B.; Stilwell, D.J. Hydrodynamic Parameter Estimation for Autonomous Underwater Vehicles. *IEEE J. Ocean. Eng.* **2020**, *45*, 385–394. [CrossRef]
11. Yao, F.; Yang, C.; Zhang, M.; Wang, Y. Optimization of the Energy Consumption of Depth Tracking Control Based on Model Predictive Control for Autonomous Underwater Vehicles. *Sensors* **2019**, *19*, 162. [CrossRef]
12. Henzinger, T.A.; Kopke, P.W.; Puri, A.; Varaiya, P. What's Decidable about Hybrid Automata? *J. Comput. Syst. Sci.* **1998**, *57*, 94–124. [CrossRef]
13. Hien, N.V.; Soriano, T. Implementing hybrid automata for developing industrial control systems. In Proceedings of the 8th IEEE-ETFA, Antibes-Juan les Pins, France, 15–18 October 2001; Volume 2, pp. 129–137, ISBN 0-7803-7241-7.
14. Carloni, L.P.; Passerone, R.; Pinto, A.; Sangiovanni, V.A. *Languages and Tools for Hybrid Systems Design*; Now Publishers Inc.: Boston, MA, USA, 2006.
15. Fishwick, P.A. (Ed.) *Handbook of Dynamic System Modeling*; Taylor & Francis Group: New York, NY, USA, 2007.
16. Qing, X.; Karimi, H.R.; Niu, Y.; Wang, J. Decentralized unscented Kalman filter based on a consensus algorithm for multi-area dynamic state estimation in power systems. *Int. J. Electr. Power Energy Syst.* **2015**, *65*, 26–33. [CrossRef]
17. Karimi, H.R. A sliding mode approach to $H\infty$ synchronization of master–slave time-delay systems with Markovian jumping parameters and nonlinear uncertainties. *J. Frankl. Inst.* **2012**, *349*, 1480–1496. [CrossRef]
18. Pettersen, K.Y.; Fossen, T.I. Guidance of Autnomous Underwater Vehicles. In *Encyclopedia of Robotocs*; Ang, M.A., Khatib, O., Sicilano, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2018.
19. Lei, M. Nonlinear diving stability and control for an AUV via singular perturbation. *Ocean Eng.* **2020**, *197*, 11. [CrossRef]
20. Khalaji, A.K.; Tourajizadeh, H. Nonlinear Lyapounov based control of an underwater vehicle in presence of uncertainties and obstacles. *Ocean Eng.* **2020**, *198*, 9. [CrossRef]
21. Li, S.; Wang, X.; Zhang, L. Finite-Time Output Feedback Tracking Control for Autonomous Underwater Vehicles. *IEEE J. Ocean. Eng.* **2015**, *40*, 727–751. [CrossRef]
22. Zhang, L.; Liu, L.; Zhang, S.; Cao, S. Saturation Based Nonlinear FOPD Motion Control Algorithm Design for Autonomous Underwater Vehicle. *Appl. Sci.* **2019**, *9*, 4958. [CrossRef]

23. Valluru, S.K.; Kaur, M.; Kartikeya, K.; Goel, A.; Dobhal, D. Experimental Investigation of Fully Informed Particle Swarm Optimization tuned Multi Loop L-PID and NL-PID Controllers for Gantry Crane System. *Procedia Comput. Sci.* **2020**, *171*, 130–138. [CrossRef]
24. Sarhadi, P.; Noei, A.R.; Khosravi, A. Model reference adaptive PID control with anti-windup compensator for an autonomous underwater vehicle. *Robot. Auton. Syst.* **2016**, *83*, 87–93. [CrossRef]
25. Guerrero, J.; Torres, J.; Creuze, V.; Chemori, A.; Campos, E. Saturation based nonlinear PID control for underwater vehicles: Design, stability analysis and experiments. *Mechatron. Sci. Intell. Mach.* **2019**, *61*, 96–105. [CrossRef]
26. Kong, F.; Guo, Y.; Lyu, W. Dynamics Modeling and Motion Control of an New Unmanned Underwater Vehicle. *IEEE Access* **2020**, *8*, 30119–30126. [CrossRef]
27. Makdah, A.A.R.A.; Daher, N.; Asmar, D.; Shammas, E. Three-dimensional trajectory tracking of a hybrid autonomous underwater vehicle in the presence of underwater current. *Ocean Eng.* **2019**, *185*, 115–132. [CrossRef]
28. Alaeddini, A.; Morgansen, K.A.; Mesbahi, M. Augmented state feedback for improving observability of linear systems with nonlinear measurements. *Syst. Control Lett.* **2019**, *133*, 8. [CrossRef]
29. Cho, G.R.; Park, D.G.; Kang, H.; Lee, M.J.; Li, J.H. Horizontal Trajectory Tracking of Underactuated AUV using Backstepping Approach. *IFAC-PapersOnLine* **2019**, *52*, 174–179. [CrossRef]
30. Ellenrieder, K.D.V. Stable Backstepping Control of Marine Vehicles with Actuator Rate Limits and Saturation. *IFAC-PapersOnLine* **2018**, *51*, 262–267. [CrossRef]
31. Guerrero, J.; Antonio, E.; Manzanilla, A.; Torres, T.; Lozano, R. Autonomous Underwater Vehicle Robust Path Tracking: Auto-Adjustable Gain High Order Sliding Mode Controller. *IFAC-PapersOnLine* **2018**, *51*, 161–166. [CrossRef]
32. Zhang, G.C.; Huang, H.; Qin, H.D.; Wan, L.; Li, Y.M.; Cao, J.; Su, Y.M. A novel adaptive second order sliding mode path following control for a portable AUV. *Ocean Eng.* **2018**, *151*, 82–92. [CrossRef]
33. Xu, H.; Zhang, G.C.; Sun, Y.S.; Pang, S.; Ran, X.R.; Wang, X.B. Design and Experiment of a Plateau Data-Gathering AUV. *J. Mar. Sci. Eng.* **2019**, *7*, 376. [CrossRef]
34. Wang, X.; Zhang, G.; Sun, Y.; Cao, J.; Wan, L.; Sheng, M.; Liu, Y. AUV near-wall-following control based on adaptive disturbance observer. *Ocean Eng.* **2019**, *190*, 17. [CrossRef]
35. Guerrero, J.; Torres, J.; Creuze, V.; Chemori, A. Adaptive disturbance observer for trajectory tracking control of underwater vehicles. *Ocean Eng.* **2020**, *200*, 13. [CrossRef]
36. Wang, J.; Wang, C.; Wei, Y.; Zhang, C. Sliding mode based neural adaptive formation control of underactuated AUVs with leader-follower strategy. *Appl. Ocean Res.* **2020**, *94*, 9. [CrossRef]
37. Elhaki, O.; Shojaei, K. A robust neural network approximation-based prescribed performance output-feedback controller for autonomous underwater vehicles with actuators saturation. *Eng. Appl. Artif. Intell.* **2020**, *88*, 16. [CrossRef]
38. Kumar, N.; Rani, M. An efficient hybrid approach for trajectory tracking control of autonomous underwater vehicles. *Appl. Ocean Res.* **2020**, *95*, 10. [CrossRef]
39. Yan, Z.; Wang, M.; Xu, J. Robust adaptive sliding mode control of underactuated autonomous underwater vehicles with uncertain dynamics. *Ocean Eng.* **2019**, *173*, 802–809. [CrossRef]
40. Han, S.; Wang, H.; Tian, Y.; Christov, N. Time-delay estimation based computed torque control with robust adaptive RBF neural network compensator for a rehabilitation exoskeleton. *ISA Trans.* **2020**, *97*, 171–181. [CrossRef] [PubMed]
41. Alvarez, J.; Arceo, J.C.; Armenta, C.; Lauber, J.; Bernal, M. An Extension of Computed-Torque Control for Parallel Robots in Ankle Reeducation. *IFAC-PapersOnLine* **2019**, *52*, 1–6. [CrossRef]
42. OMG. *Documents Associated with Unified Modeling Language™ (UML$^{®}$ Version 2.5.1): OMG Formal/17-12-05*; OMG: Needham, MA, USA, 2017. Available online: http://www.omg.org/spec/UML/ (accessed on 19 April 2019).
43. OMG. *SysML Specifications Version 1.6: OMG Formal/19-11-01*; OMG: Needham, MA, USA, 2019. Available online: https://www.omg.org/spec/SysML/ (accessed on 22 March 2020).
44. INCOSE. *Systems Engineering Vision 2025*; INCOSE: San Diego, CA, USA, 2014.
45. INCOSE. Model-Based Systems Engineering (MBSE). Available online: https://www.incose.org/ (accessed on 22 January 2020).

46. Board, B.E. The Guide to the Systems Engineering Body of Knowledge (SEBoK), V2.2. Available online: https://www.sebokwiki.org/ (accessed on 10 September 2020).
47. Estefan, J.A. *Survey of Model-Based Systems Engineering (MBSE) Methodologies. Rev B INCOSE Technical Publication, Document No. INCOSE-TD-2007-003-01*; INCOSE: San Diego, CA, USA, 2008.
48. Douglass, B.P. The Telelogic Harmony/ESW process for realtime and embedded development, IBM Corporation Software Group Somers, NY 10589, USA. *White Pap.* **2008**, *2008*, 12.
49. Douglass, B.P. *Real-Time Agility: The Harmony/ESW Method for Real-Time and Embedded Systems Development*, 1st ed.; Pearson Education: Boston, MA, USA, 2009.
50. Lykins, H.; Friedenthal, S.; Meilich, A. Adapting UML for an Object Oriented Systems Engineering Method (OOSEM). In Proceedings of the INCOSE International Symposium, Minneapolis, MN, USA, 16–20 July 2020; pp. 490–497.
51. INCOSE. Object-Oriented SE Method. Available online: https://www.incose.org/incose-member-resources/working-groups/transformational/object-oriented-se-method (accessed on 12 September 2020).
52. Cantor, M. Rational Unified Process® for Systems Engineering: RUP SE Version 2.0. *Ibm Ration. EdgeWhite Pap.* **2003**, *2003*, 17.
53. Ingham, M.D.; Rasmussen, R.D.; Bennett, M.B.; Moncada, A.C. Generating requirements for complex embedded systems using State Analysis. *Acta Astronaut.* **2006**, *58*, 648–661. [CrossRef]
54. Dori, D. *Object-Process Methodology: A Holistic Systems Paradigm*; Springer: New York, NY, USA, 2002.
55. Dori, D. *Model-Based Systems Engineering with OPM and SysML*; Springer: New York, NY, USA, 2016.
56. OMG. *Model Driven Architecture (MDA): Guide Revision 2.0 of MDA Guide Version 1.0.1 (12 June 2003)*; OMG Document ormsc/2014-06-01; OMG: Needham, MA, USA, 2014. Available online: http://www.omg.org/cgi-bin/doc?ormsc/14-06-01 (accessed on 25 July 2019).
57. OMG. MDA Success Stories. Available online: https://www.omg.org/mda/products_success.htm (accessed on 24 April 2020).
58. Sebastián, G.; Gallud, J.A.; Tesoriero, R. Code generation using model driven architecture: A systematic mapping study. *J. Comput. Lang.* **2020**, *56*, 11. [CrossRef]
59. OMG. *Unified Architecture Framework, Version 1.1: Formal/19-11-07*; OMG: Needham, MA, USA, 2020. Available online: https://www.omg.org/spec/UAF (accessed on 12 August 2020).
60. Agner, L.T.W.; Soares, I.W.; Stadzisz, P.C.; Simão, J.M. A Brazilian survey on UML and model-driven practices for embedded software development. *Syst. Softw.* **2013**, *86*, 997–1005. [CrossRef]
61. Rashid, M.; Anwar, M.W.; Khan, A.M. Toward the tools selection in model based system engineering for embedded systems—A systematic literature review. *J. Syst. Softw.* **2015**, *106*, 150–163. [CrossRef]
62. Freire, L.O.; Oliveira, L.M.; Vale, R.T.S.; Medeiros, M.; Diana, R.E.Y.; Lopes, R.M.; Pellini, E.L.; Barros, E.A. Development of an AUV control architecture based on systems engineering concepts. *Ocean Eng.* **2018**, *151*, 157–169. [CrossRef]
63. Hien, N.V.; He, N.V.; Diem, P.G. A model-driven implementation to realize controllers for Autonomous Underwater Vehicles. *Appl. Ocean Res.* **2018**, *78*, 307–319. [CrossRef]
64. Soriano, T.; Hien, N.V.; Tuan, K.M.; Anh, T.V. An object-unified approach to develop controllers for autonomous underwater vehicles. *Mechatron. Sci. Intell. Mach.* **2016**, *35*, 54–70. [CrossRef]
65. Anwar, M.W.; Rashid, M.; Azam, F.; Kashif, M. Model-based design verification for embedded systems through SVOCL: An OCL extension for SystemVerilog. *Des. Autom. Embed. Syst.* **2017**, *21*, 1–36. [CrossRef]
66. Anwar, M.W.; Rashid, M.; Azam, F.; Kashif, M.; Butt, W.H. A model-driven framework for design and verification of embedded systems through SystemVerilog. *Des. Autom. Embed. Syst.* **2019**, *23*, 179–223. [CrossRef]
67. Soriano, T.; Pham, H.A.; Hien, N.V. Analysis of coordination modes for multi-UUV based on Model Driven Architecture. In Proceedings of the 12th France-Japan and 10th Europe-Asia Congress on Mechatronics, Tsu, Japan, 10–12 September 2018.
68. OMG. *UML Profile for MARTE: UML for Model-Driven Development of Real Time and Embedded Systems (RTES)*; OMG formal/19-04-02; OMG: Needham, MA, USA, 2019. Available online: https://www.omg.org/spec/MARTE/:OMG (accessed on 26 May 2020).
69. Douglass, B.P. *Real-Time UML Workshop for Embedded Systems*, 2nd ed.; Elsevier: Oxford, UK, 2014.
70. Selic, B.; Gerard, S. *Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE*; Elsevier: Amsterdam, The Netherlands, 2014.

71. Selic, B. Using UML for modeling complex real-time systems. *Lect. Notes Comput. Sci.* **1998**, *1474*, 250–260. [CrossRef]
72. SNAME. *Nomenclature for Treating the Motion of a Submerged Body through a Fluid*; SNAME: New York, NY, USA, 1950.
73. Fossen, T.I. *Handbook of Marine Craft Hydrodynamics and Motion Control*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
74. Figueiredo, A.B.; Matos, A.C. MViDO: A High Performance Monocular Vision-Based System for Docking A Hovering AUV. *Appl. Sci.* **2020**, *10*, 2991. [CrossRef]
75. Martínez, N.L.; Ortega, J.F.M.; Castillejo, P.; Martínez, V.B. Survey of Mission Planning and Management Architectures for Underwater Cooperative Robotics Operations. *Appl. Sci.* **2020**, *10*, 1086. [CrossRef]
76. García, J.G.; Espinosa, A.G.; Urquizo, E.C.; Valdovinos, L.G.G.; Jiménez, T.S.; Cabello, J.A.E. Autonomous Underwater Vehicles: Localization, Navigation, and Communication for Collaborative Missions. *Appl. Sci.* **2020**, *10*, 1256. [CrossRef]
77. Yao, F.; Yang, C.; Liu, X.; Zhang, M. Experimental Evaluation on Depth Control Using Improved Model Predictive Control for Autonomous Underwater Vehicle (AUVs). *Sensors* **2018**, *18*, 2321. [CrossRef]
78. Wan, E.A.; Merwe, R.V.D. The Unscented Kalman Filter. In *Kalman Filtering and Neural Networks*; Haykin, S., Ed.; Wiley: New York, NY, USA, 2001; pp. 221–280.
79. Bar-Shalom, Y.; Li, X.R.; Kirubarajan, T. *Estimation with Applications to Tracking and Navigation-Theory Algorithms and Software*; John Wiley & Sons: Hoboken, NJ, USA, 2001.
80. Allotta, B.; Caitib, A.; Costanzi, R.; Fanelli, F.; Fenucci, D.; Meli, E.; Ridolfi, A. A new AUV navigation system exploiting unscented Kalman filter. *Ocean Eng.* **2016**, *113*, 121–132. [CrossRef]
81. Allotta, B.; Chisci, L.; Costanzi, R.; Corato, F.D.; Fantacci, C.; Fenucci, D.; Meli, E.; Ridolfi, A. An unscented Kalman filter based navigation algorithm for autonomous underwater vehicles. *Mechatron. Sci. Intell. Mach.* **2016**, *39*, 185–195. [CrossRef]
82. Dong, L.; Xu, H.; Feng, X.; Han, X.; Yu, C. An Adaptive Target Tracking Algorithm Based on EKF for AUV with Unknown Non-Gaussian Process Noise. *Appl. Sci.* **2020**, *10*, 3413. [CrossRef]
83. Lekkas, A.M.; Fossen, T.I. Integral LOS Path Following for Curved Paths Based on a Monotone Cubic Hermite Spline Parametrization. *IEEE Trans. Control Syst. Technol.* **2014**, *22*, 2287–2301. [CrossRef]
84. Shojaei, K.; Dolatshahi, M. Line-of-sight target tracking control of underactuated autonomous underwater vehicles. *Ocean Eng.* **2017**, *133*, 244–252. [CrossRef]
85. Zheng, Z.; Zou, Y. Adaptive integral LOS path following for an unmanned airship with uncertainties based on robust RBFNN backstepping. *ISA Trans.* **2016**, *65*, 210–219. [CrossRef]
86. Liu, F.; Shen, Y.; He, B.; Wan, J.; Wang, D.; Yin, Q.; Qin, P. 3DOF Adaptive Line-Of-Sight Based Proportional Guidance Law for Path Following of AUV in the Presence of Ocean Currents. *Appl. Sci.* **2019**, *9*, 3518. [CrossRef]
87. Lantos, B.; Márton, L. *Nonlinear Control of Vehicles and Robots*; Springer: London, UK, 2011.
88. Wan, J.; He, B.; Wang, D.; Yan, T.; Shen, Y. Fractional-Order PID Motion Control for AUV Using Cloud-Model-Based Quantum Genetic Algorithm. *IEEE Access* **2019**, *7*, 124828–124843. [CrossRef]
89. Zhou, J.; Zhao, X.; Chen, T.; Yan, Z.; Yang, Z. Trajectory Tracking Control of an Underactuated AUV Based on Backstepping Sliding Mode with State Prediction. *IEEE Access* **2019**, *7*, 181983–181993. [CrossRef]
90. Yan, Z.; Yang, Z.; Zhang, J.; Zhou, J.; Jiang, A.; Du, X. Trajectory Tracking Control of UUV Based on Backstepping Sliding Mode with Fuzzy Switching Gain in Diving Plane. *IEEE Access* **2019**, *7*, 166788–166795. [CrossRef]
91. IBM. IBM Rational's Methodology, Software, Online Documentation and Training Kits. Available online: https://my15.digitalexperience.ibm.com/b73a5759-c6a6-4033-ab6b-d9d4f9a6d65b/dxsites/151914d1-03d2-48fe-97d9-d21166848e65/academic/home (accessed on 20 April 2020).
92. Papyrus. Eclipse Papyrus for Real-Time ("Papyrus-RT"). Available online: https://www.polarsys.org/papyrus-ic/products (accessed on 20 April 2020).
93. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley: Oxford, UK, 1995.
94. Douglass, B.P. *Design Patterns for Embedded Systems in C: An Embedded Software Engineering Toolkit*, 1st ed.; Elsevier: Oxford, UK, 2011.

95. Arduino. Open-Source Electronics Prototyping Platform for Hardware and Software. Available online: http://www.arduino.cc/ (accessed on 19 January 2020).
96. Price, W.G.; Bishop, R.E.D. *Probalistic Theory of Ship Dynamics*; Chapman and Hall.: London, UK, 1974.
97. InvenSense. Sensor System on Chip. Available online: http://www.invensense.com/ (accessed on 22 March 2020).
98. u-blox. Product Selector. Available online: https://www.u-blox.com/en/product-search (accessed on 18 February 2020).
99. OpenModelica. OpenModelica Software, Version 1.14. Available online: https://www.openmodelica.org/ (accessed on 20 April 2020).
100. Fritzson, P. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*, 2nd ed.; Wiley-IEEE Press: Hoboken, NJ, USA, 2015.
101. Zhang, B.; Chu, H.; Sun, T.; Jia, H.; Guo, L.; Zhang, Y. Error Prediction for SINS/GPS after GPS Outage Based on Hybrid KF-UKF. *Math. Probl. Eng.* **2015**, *2015*, 10. [CrossRef]
102. Raitoharju, M.; Piché, R. On Computational Complexity Reduction Methods for Kalman Filter Extensions. *IEEE Aerosp. Electron. Syst. Mag.* **2019**, *34*, 2–19. [CrossRef]
103. OMG. *XML Metadata Interchange Version 2.5.1: OMG Formal/15-06-07*; OMG: Needham, MA, USA, 2015. Available online: https://www.omg.org/spec/XMI/ (accessed on 17 May 2020).