Measurement of Effectiveness for an Anti-torpedo Combat System Using a Discrete Event System Specification-based Underwater Warfare Simulator

Kyung-Min Seo, Hae Sang Song, Se Jung Kwon and Tag Gon Kim

The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology published online 18 January 2011
DOI: 10.1177/1548512910390245

Original article

JDMS

Journal of Defense Modeling and Simulation: Applications, Methodology, Technology
XX(X) 1–15

DOI: 10.1177/1548512910390245
http://dms.sagepub.com

SAGE

Measurement of Effectiveness for an Anti-torpedo Combat System Using a Discrete Event Systems Specification-based Underwater Warfare Simulator

Kyung-Min Seo[1], Hae Sang Song[2], Se Jung Kwon[1] and Tag Gon Kim[1]

Abstract

Modeling and simulation (M&S) has long played an important role in developing tactics and evaluating the measure of effectiveness (MOE) for the underwater warfare system. In simulation-based acquisition, M&S technology facilitates decisions about future equipment procurements, such as a mobile decoy or a torpedo. In addition, assessment of submarine tactical development, during an engagement against a torpedo, can be conducted using M&S techniques. This paper presents a case study that applies discrete event systems specification-based M&S technology to develop a simulation of an underwater warfare system, specifically, an anti-torpedo combat system, to analyze the MOE of the system. The entity models required for M&S are divided into three sub-models: controller, maneuver, and sensor model. The developed simulation allows us to conduct a statistical evaluation of the overall underwater warfare system under consideration, an assessment of the anti-torpedo countermeasure's effectiveness, and an assessment of tactics development of the underwater vehicle. Moreover, it can be utilized to support the decision-making process for future equipment procurements. In order to analyze the system effectiveness, we performed extensive combat experiments by varying parameters, such as various tactics and weapon performance. The experimental results show how the factors influence the MOEs of the underwater warfare system.

Keywords

discrete event system, system effectiveness analysis, underwater warfare modeling and simulation

1. Introduction

As vehicles in underwater warfare, such as a torpedo, a decoy, or a submarine, have become more diverse and complicated, it has become increasingly important that underwater warfare systems control risk.[1] In order to determine requirements for underwater warfare systems, design them, and then evaluate the results, various modeling and simulation (M&S) techniques have been used. For instance, in simulation-based acquisition and development, M&S technology has largely facilitated decisions about equipment procurements, such as a mobile decoy. In addition, assessment of submarine tactics development, under the engagement situation against a torpedo, can make use of M&S techniques.[2] Hence, the purpose of this work is: (1) to develop the underwater warfare simulation; and (2) to analyze the effectiveness of an underwater combat system using the simulation. The developed underwater warfare simulation is based on the discrete event systems specification (DEVS) formalism.

For more efficient model development, we propose a generic three-part underwater platform model, which is flexible enough to be easily re-usable for developing different underwater platform models with different behaviors and structures. Furthermore, underwater warfare models

[1] Department of EE, KAIST, Republic of Korea
[2] Department of Computer Engineering of Seowon University, Republic of Korea

Corresponding author:
Kyung-Min Seo, Department of EE, KAIST, 335 Gwahangno, Yuseong-gu, Daejeon, Korea 305-701.
Email: kmseo@smslab.kaist.ac.kr

often are required to apply new algorithms or detailed behaviors during the maintenance stage. For example, the sonar model has various types of sensing algorithms, including cylindrical array (CAS), mine avoidance (MAS), flank array (FAS), passive ranging (PRS), and towed array (TAS) algorithms, any of which may need to be analyzed. In particular, in the case of simulation of a surface ship's maneuvers to evade a torpedo, factors such as the ship's detection range, evasive speed, and countermeasures influence the overall system effectiveness. Hence, it is necessary to develop an underwater model that incorporates a flexible architecture accommodating various algorithms in order to compare measures of effectiveness (MOEs) among them.

This paper contributes to the defense M&S community in two aspects:

- it illustrates how to design an underwater warfare model using the DEVS formalism;
- it provides insights about how various factors, such as tactics and the performance of underwater weapons, influence the MOEs of the system.

This paper is organized as follows. Section 2 presents several of the related works and the DEVS formalism. Section 3 explains the underwater warfare model design, and Sections 4 and 5 explain the implementation and simulation of the underwater warfare model. Section 6 illustrates the effectiveness analysis utilizing the experimental results from the anti-torpedo combat system. Section 7 concludes this research and proposes future extensions for a more complete solution.

2. Related Works

We first surveyed previous works about underwater warfare simulation, specifically by comparing and contrasting the key characteristics of those simulation developments. Secondly, we introduce the DEVS formalism that we apply for modeling the underwater warfare system in this paper.

2.1 Previous Research

In recent years, some efforts have been made to develop M&S techniques for underwater warfare. We surveyed them to show the pros and cons of developed simulations, as described in Table 1.

Armo[3] has studied the simulation of a submarine's evasion against a torpedo. He only concentrated on the relationship between the submarine's maximum speed and its evasive capability against the torpedo. He did not consider other conditions, such as the configuration of countermeasures, evasive tactics of the submarine, and the performance of the torpedo's countermeasure system. In 2007, Cho et al.[4] extended Armo's work. They considered not only the maximum speed, but also the acceleration and the performance of the torpedo's countermeasure systems, such as a decoy and a jammer. In their study, all simulation models were developed using the DEVS formalism and simulated in DEVSim++.[5] Nevertheless, their research also has some disadvantages. Their work is difficult to use to test candidate tactics of underwater platforms. If a user wants to evaluate an alternative evasion tactic for the submarine, he must develop another submarine DEVS model. Because the model cannot be reused, it presents a disadvantage for modelers. Liang and Wang[2] studied the simulation of a submarine's evasion of a torpedo. It can be used to test candidate tactics of a submarine, but this study was not based on mathematical representations, such as the DEVS formalism, and complicated model development or the functional extension of the existing model takes a considerable amount of time. Moreover, it is difficult to use it to compare similar models. Park et al.[6] proposed M&S methodology for small-scale engagement. They divided the model entities into three categories: combat, logical, and environmental entities. In particular, the combat entities are further modeled into Shell and Core Parts to improve their reusability under various combat scenarios. Nevertheless, their study is only focused on small-scale engagement.

Table 1. Comparisons of previous research.

Simulation models	Pros	Cons
Armo[3]	Statistical evaluation for the specific MOE	Only concentrated on the relationship between the submarine's maximum speed and its evasive capability
Cho et al.[4]	Based on mathematical representation Considered not only the maximum speed but also the performance of torpedo's countermeasure systems	Difficult to use to test candidate tactics of underwater platforms
Liang and Wang[2]	Used to test candidate tactics of underwater platforms	Not based on mathematical representation Lengthy development time for bigger models Difficult to compare the models
Park et al.[6]	Based on mathematical representation Model architecture to improve reusability under various combat scenarios	M&S methodology for small-scale engagement Do not consider countermeasure systems Need to be expanded to cover a broad spectrum of combat simulation

They do not consider any submarine's evasive capabilities, such as countermeasure systems and evasive maneuvers. Therefore it needs to be expanded to cover a broad spectrum of combat simulation.

In summary, the developed simulations are focused on analyzing the effectiveness of developed simulations, and each study has some disadvantages, as illustrated in Table 1. Therefore, in this paper, we overcome these disadvantages by using co-modeling methodology.[7] The co-modeling methodology has been proposed to overcome the difficulty of developing domain-specific simulation systems. In some cases, we need to develop an underwater warfare model with an open architecture accommodating various algorithms in order to compare MOEs among algorithms. In these cases, the co-modeling methodology gives us a way to change detailed behaviors of the model.

2.2 DEVS formalism

The DEVS formalism, a set-theoretic formalism, specifies discrete event systems in a hierarchical and modular form. The DEVS formalism provides the framework for information modeling, which has several advantages, such as completeness, verifiability, extensibility, and maintainability,[8] for analyzing and designing complex systems. With the formalism, one can specify a discrete event system more easily by decomposing a large system into smaller component models. The DEVS formalism consists of two kinds of models: the atomic model and the coupled model.

An atomic model is the basic model and has specifications for the dynamics of the model. Formally, a 7-tuple specifies an atomic model M as follows:

$$M = < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$$

where X is a set of input events, Y is a set of output events, S is a set of sequential states, δ_{ext} is $Q \times X \rightarrow S$, an external transition function, where $Q = \{(s,e) | s \in S, 0 \leq e \leq ta(s)\}$ is the total state set of M, δ_{int} is $S \rightarrow S$, an internal transition function, λ is $S \rightarrow Y$, an output function, and ta is $S \rightarrow$ Real, a time advance function.

A coupled model provides the method of assembly of several atomic and/or coupled models to build complex systems hierarchy. Formally, a coupled model is defined as follows:

$$DN = < X, Y, M, EIC, EOC, IC, SELECT >$$

where X is a set of input events, Y is a set of output events, M is a set of all component models, $EIC \subseteq DN.X \times \cup M.X$ is the external input coupling, $EOC \subseteq \cup M.Y \times DN.Y$ is the external output coupling, $IC \subseteq \cup M.Y \times \cup M.X$ is the internal coupling, and SELECT: $2^M - \varphi \rightarrow M$ is the tie-breaking selector.

An overall system consists of a set of component models, either atomic or coupled, thus being in a hierarchical structure. Each DEVS model, either atomic or coupled, has correspondence to an object in the real-world system to be modeled. Within the DEVS framework, model design may be performed in a top-down fashion and model implementation in a bottom-up manner.

3. Model Design

We developed the proposed underwater warfare models at two layers: a discrete event system (DES) layer and an object model (OM) layer.[7] The DES layer represents abstract behavior of an object, such as a torpedo, a decoy, or a submarine, using the DEVS formalism, whereas the OM layer details behavior of the same object using flow charts or mathematical equations. This approach enables one to develop an underwater warfare model with an open architecture accommodating various algorithms in order to compare MOEs among algorithms. The proposed model provides the user with a generic model architecture for an underwater warfare model, by means of which a detailed behavioral description relating to underwater warfare systems can be constructed.[9]

3.1 Overall Model Structure

The developed overall model architecture is described in Figure 1. The overall model architecture consists of two main models: a simulation model for the user's system and an experimental frame model for analysis of the user's system. The simulation model consists of four underwater platform models: a submarine, a surface ship, a torpedo, and a decoy. The attacking platform is a submarine, and the target platform is a surface ship. A heavyweight torpedo is used, which is a submarine-launched torpedo for surface ship targets. In addition, as countermeasures we consider anti-torpedo sonar decoys.

These four models are instances of the underwater platform model indicated in Figure 2. The dotted arrow is actually a dependency relation that has been given a special meaning by the stereotype ⟪instantiate⟫, and the ⟪instantiate⟫ turns an ordinary dependency into an instantiation relationship between a class and objects of that class: underwater platform models.[10] This feature enables the user to create and add other underwater platform models more easily. Each underwater platform model consists of three parts (maneuver, sensor, and controller model), as described in Figure 1. The functional role of each model is described in the following section.

3.2 Underwater Platform Model

The DEVS coupled model describes the event message exchange relations among the DEVS atomic models. The following specification represents the DEVS coupled

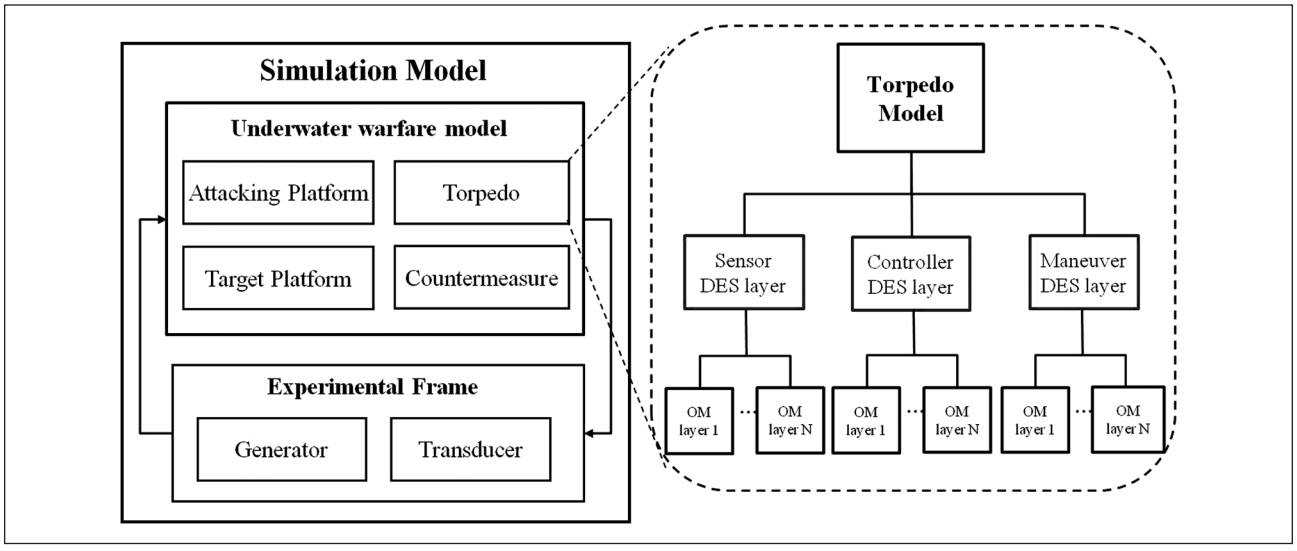

Figure 1. Required elements for the simulation.

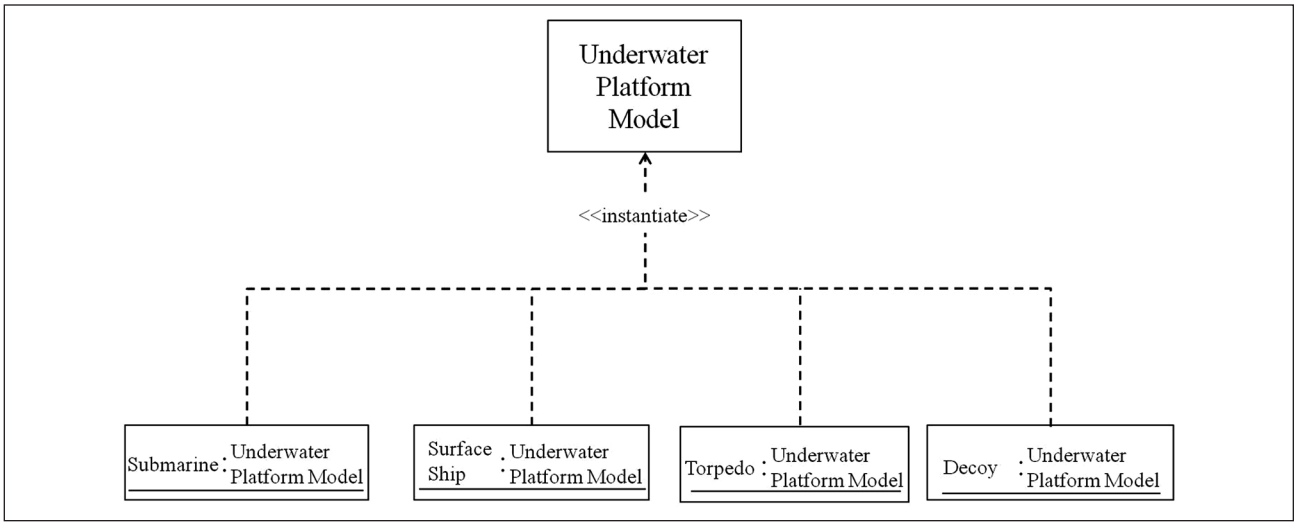

Figure 2. Instantiation of underwater platform model.

model of the underwater platform model. Figure 3 shows the diagram of the model. As described earlier, the entire underwater platform models, such as submarines, torpedoes and ships, have an identified structure of the DEVS coupled model. Most weapon systems, including underwater warfare vehicles, have three major functions, which are movement, detection, and command and control (C2). Therefore, we classified the underwater platform model according to specific functions, and the underwater platform model is divided into three sub-models: the controller model, the sensor model, and the maneuver model. Each sub-model consists of a DES layer represented by the DEVS formalism and an OM layer described as detailed behaviors. We call it a generic three-part underwater platform model. As shown later in this paper, this generic three-part model is flexible enough to represent any underwater platform's structure and behavior. Therefore, each sub-model is reusable for other platform models. In the following sections, we will describe each sub-model in more detail. In addition, the three-part modeling methodology allows us to do straightforward model comparison and to achieve the complicated model development with less cost.

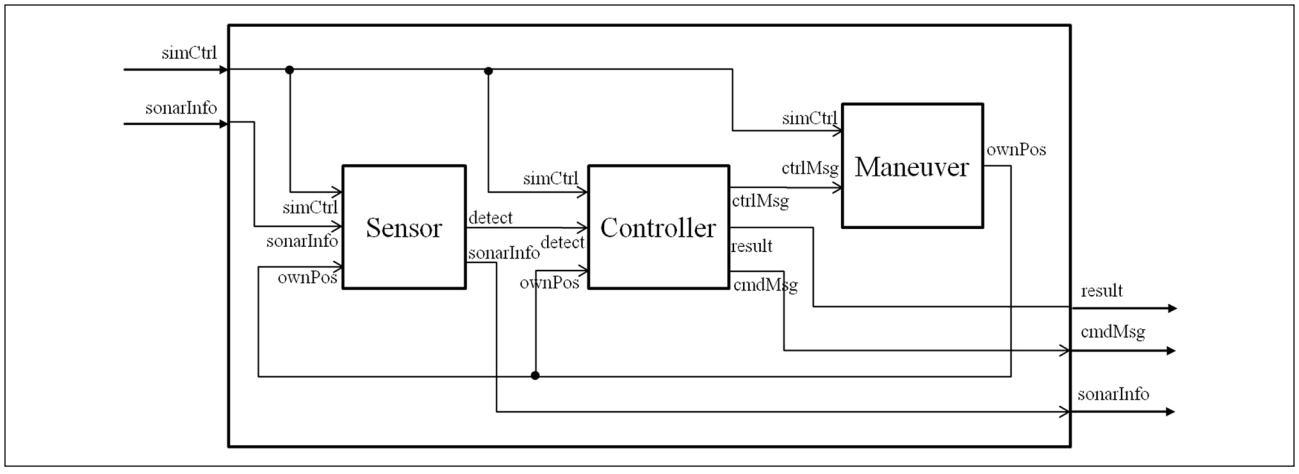

Figure 3. DEVS coupled model diagram of the underwater platform model.

$CM_{\text{UNDERWATER\_PLATFORM}} = < X, Y, \{M_i\}, EIC, EOC, IC, Sel >$,
X = {"simCtrl", "sonarInfo"}
Y = {"result", "cmdMsg", "sonarInfo"}
$\{M_i\}$ = {Sensor, Controller, Maneuver}
EIC = {(CM.simCtrl, Sensor.simCtrl),
(CM.simCtrl, Controller.simCtrl),
(CM.simCtrl, maneuver.simCtrl),
(CM.sonarInfo, Sensor.sonarInfo)}
EOC = {(Controller.result, CM.result),
(Controller.cmdMsg, CM.cmdMsg),
(Sensor.sonarInfo, CM.sonarInfo)}
IC = {Sensor.detect, Controller.detect},
(Controller.ctrlMsg, Manuver.ctrlMsg),
(Maneuver.ownPos, Sensor.ownPos),
(Maneuver.ownPos, Controller.ownPos)}
Sel({Sensor, Controller}) = Controller

3.2.1. Controller Model The controller model takes on the role of dynamic decision-making under some uncertainty. The major roles of the controller models are as follows:

- target detection: detect targets based on data from sensor models;
- identification: identify threats from detected targets;
- battle planning: decide how to deal with the identified threat;
- weapon-to-target assignment: after completing a set of decisions, assign weapons to engage each threat;
- hit assessment: after engagement, decide whether hit occurred or not.

The following specification represents the DES layer and the OM layer of the controller model in detail. Figure 4 is the DEVS atomic model diagram for the controller model. In Figure 4, the shaded circle means the initial state. Since the set-theoretic specification can be easily turned into the diagram, we will use this diagram to show the DEVS atomic model of this paper.

DES Layer

$AM_{\text{CONTROL\_DES}} = < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$,
X = {"simCtrl", "detect", "ownPos"}
Y = {"result", "cmdMsg", "ctrlMsg"}
δ_{ext}: WAIT × "simCtrl" → WAIT
WAIT × "ownPos" → WAIT
WAIT × "detect" → IDENTIFY
IDENTIFY × "detect" → IDENTIFY
IDENTIFY × "detect" → CONTROL
IDENTIFY × "ownPos" → IDENTIFY
IDENTIFY × "simCtrl" →WAIT
CONTROL × "simCtrl" → WAIT
δ_{int}: CONTROL → IDENTIFY
CONTROL → WAIT
λ: CONTROL → "cmdMsg"
CONTROL → "ctrlMsg"
CONTROL → "result"
ta(WAIT) = ∞, ta(IDENTIFY) = ∞
ta(CONTROL) = t_{control} (time step for C2 operation)

OM Layer

O_f: Identification
O_f: (x, y, z, dx, dz, v, $flag$) = f(x, y, z, dx, dz, v)
O_g: *BattlePlanning*
O_g: (x, y, z, dx, dz, v, δt) = f(x, y, z, dx, dz, v, $flag$)

where (x, y, z) represent the current position, which is considered in the Cartesian coordinate system xyz, and dx, dz, v, and δt represent pitch and yaw angles, speed, and time step for the updating position, respectively, and $flag$ reports whether the detected target actually becomes a threat.

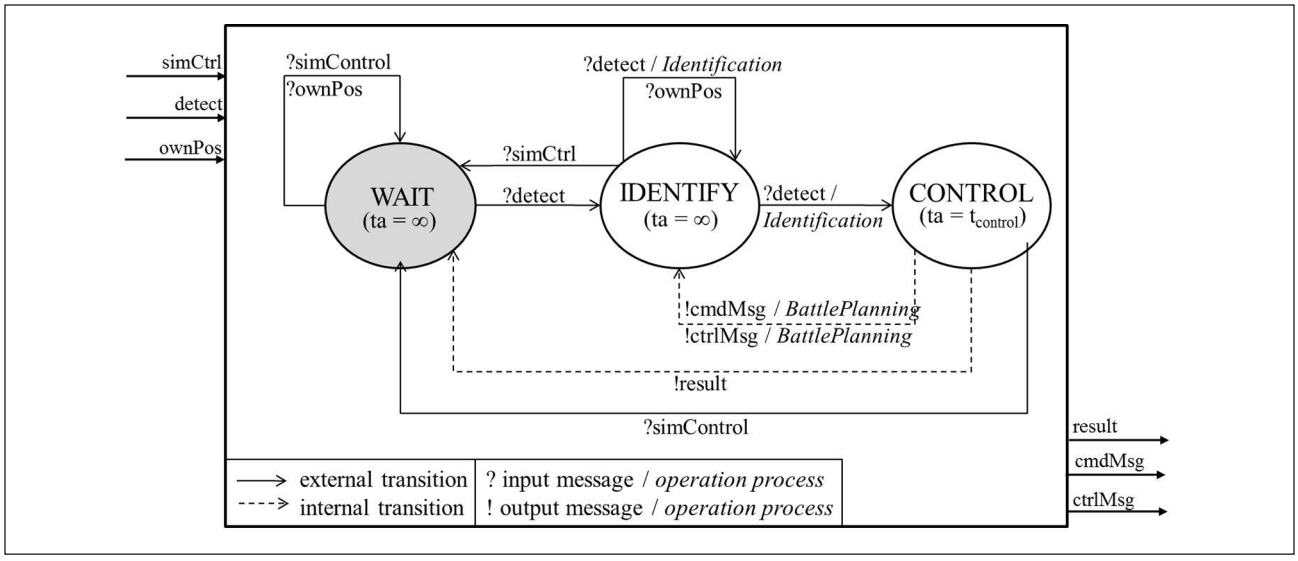

Figure 4. DEVS atomic model diagram of the controller model.

For one simple example of the O_g, the flow chart of the torpedo controller is depicted in Figure 5. In this example, we consider a heavyweight torpedo, which is a submarine-launched torpedo for surface ship targets. The torpedo controller model includes three different types of search modes; Snake Search, Circular Search, and Straight Search. The name of the mode reveals the navigation patterns of the torpedo. In accordance with the combination of these three search modes, various searching algorithms can be developed, and it has an effect on the MOEs.

In some cases, the DES layer of the controller model can be revised easily according to future specific operations. For example, in case of underwater warfare that should reflect network centric warfare, link operations, such as link-11, should be needed in the underwater platform model. These specific operations can be supplied to the controller model, and the DES layer can be modified in accordance to corresponding functions.

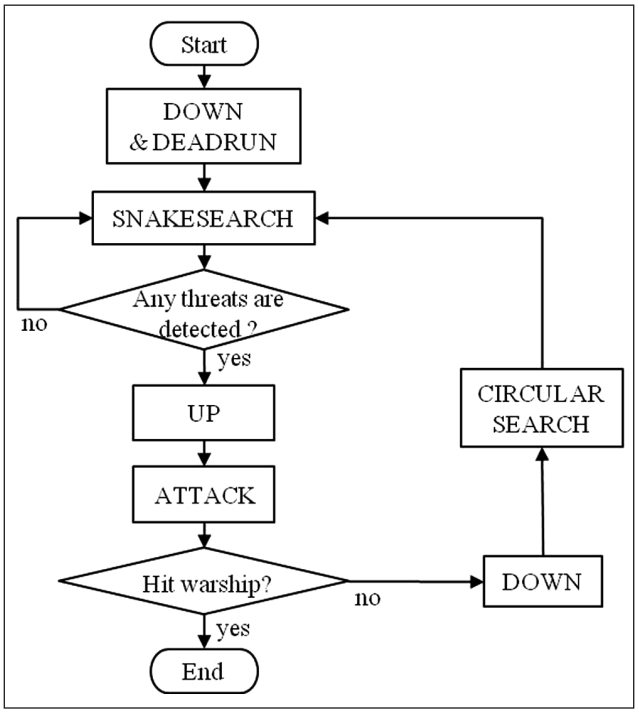

Figure 5. Flow chart of torpedo controller model for the *BattlePlanning* operation.

3.2.2. Maneuver Model

The maneuver model represents the movement of the underwater platform model. The DES layer controls the overall event messages. In particular, the model receives the request for the position information and sends the information to the controller model. The OM layer calculates the next position using maneuver equations. The following specification represents the DES and OM layers of the controller model in detail. Figure 6 is the DEVS atomic model diagram for the maneuver model.

DES Layer Refer to Figure 6.
OM Layer

$$O_h: CalculateNextPosition$$
$$O_h: (x_n, y_n, z_n, dx, dz, v) = f(x_c, y_c, z_c, dx, dz, v, \delta t)$$
$$x_n = x_c + v\cos(dx)\cos(dz) \cdot \delta t$$
$$y_n = y_c + v\sin(dx)\cos(dz) \cdot \delta t$$
$$z_n = z_c - v\sin(dz) \cdot \delta t$$

where (x_c, y_c, z_c) and (x_n, y_n, z_n) represent the current position and the next position after calculation. They are considered in the Cartesian coordinate system xyz, and dx,

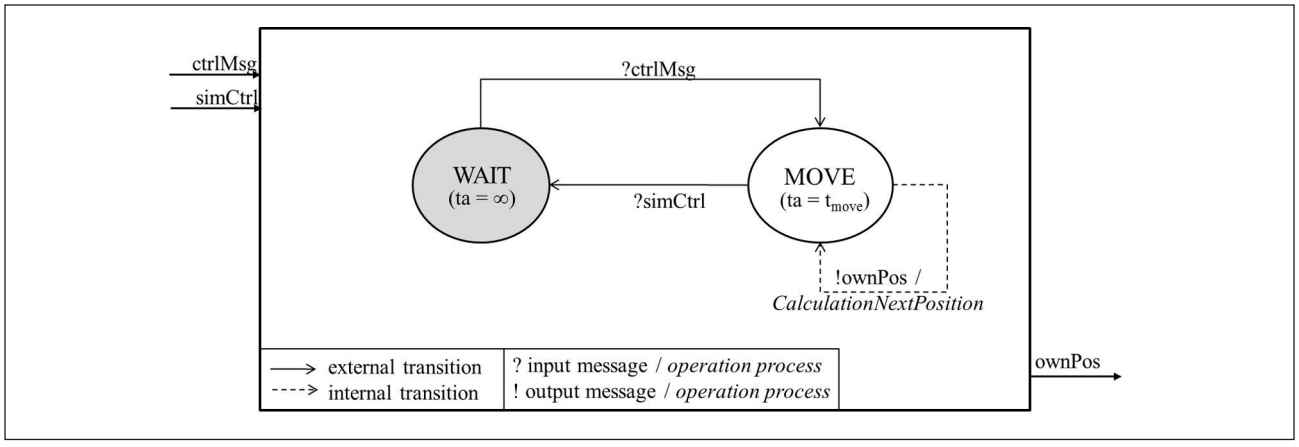

Figure 6. DEVS atomic model of the maneuver model.

dz, *v*, and δt represent pitch and yaw angles, speed, and time step for updating position, respectively.

3.2.3. Sensor Model

The underwater warfare entities have a number of acoustic receivers, a number of acoustic reflectors, and a number of radiated noise emitters distributed along their hulls. The passive sonar listens to the sound radiated by a target using a hydrophone, and detects signals against a background of the ambient noise of the sea and self-noise of the sonar platform. Therefore, the passive sonar system needs a transmitter for radiating the sound and a receiver for listening to the radiated sound. The active sonar uses a transmitter to generate a pulse of sound that travels through the water to a target and is returned as an echo to a hydrophone. In this case, the kind of sensor is a reflector, transmitter, and receiver. For the passive sonar system, the transmitter and the receiver should be modeled in the sensor model and for the active sonar system all the three parts should be modeled. In this research, we considered only the passive sonar system. Therefore, the transmitter and receiver are modeled. Figure 7 shows the DEVS model diagrams for the sensor model.

DES Layer Refer to Figure 7.
OM Layer of the Sensor Receiver Model

O_i: *UpdateInfo*
I_i: $f(x, y, z. dx, dz, v, SL, id)$
O_j: *TargetDetection*
I_j: $(x, y, z. dx, dz, v) = f(x, y, z. dx, dz, v, SL, id)$

where all of the variables except *id* and *SL* are identical to those in the controller OM layer model. *id* represents the threat's id, distinguishing it from other threats, and *SL* is the underwater noise emitted by the instance of the underwater platform model.

4. Model Implementation

The previous section described the design of the DES layers and the OM layers for the underwater warfare model according to the co-modeling methodology.[7] In this section, implementation of the designed DES and OM layers for simulation will be described.

Several simulation environments, such as DEVSim++,[5] CD++,[11] and jointMEASURE,[12] have been developed. Among them, we utilized DEVSim++ for the simulation environment. Figure 8 represents the implementation of the designed model. The experimental frame model and the DES layers of the underwater platform model are implemented using C++ with the DEVSim++ library, and the detailed algorithms and equations of the OM layers are implemented using C++. As mentioned previously, the change in detailed algorithms and dynamic equations is more frequent than the change in abstract behavior. To support this phenomenon effectively, a shared library, such as the dynamically linked library (DLL), is used. It enables users to switch algorithms or dynamic equations at run-time without re-compiling.[13]

In other words, the OM layer provides various algorithm candidates implemented with the DLL. A communication interface called *function prototype* in C++ is used to link the DES layer and corresponding OM layer.[7] The DES can pick an OM in the DLL pool via a function call. When selecting the OM, the DES is only interested in the inputs, the output, and the name of the *function prototype*. This means that the OM is available for another DES if the DES is implemented using the same name of the *function prototype* and the same input and output (I/O). Therefore, the simulation platform can be quickly constructed by simply plugging in the entities with minor adjustments in light of the scenarios. By splitting the platform objects into two parts for modeling, the reusability of the platform objects can be significantly enhanced.

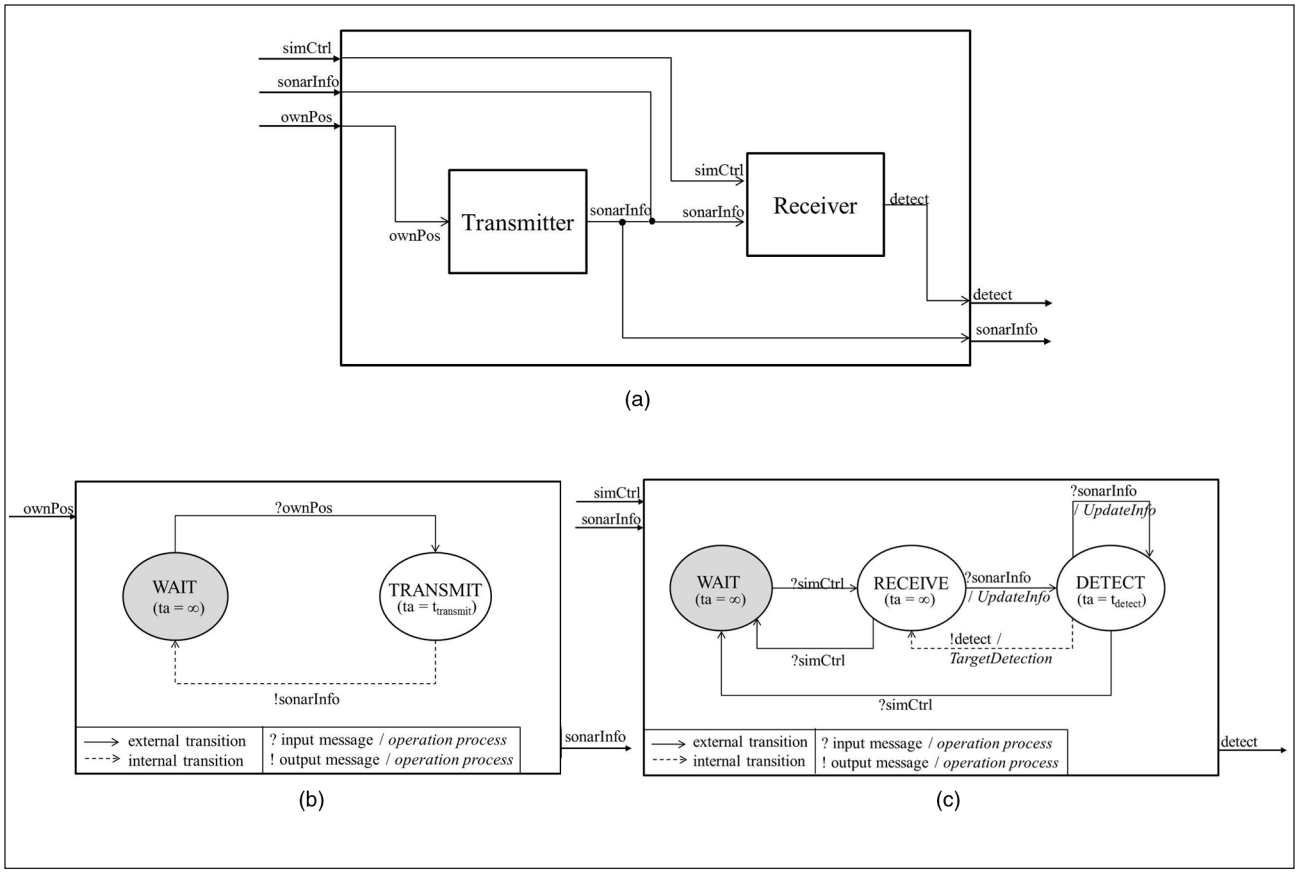

Figure 7. DEVS models of the sensor model. (a) DEVS coupled model of the sensor model. (b) DEVS atomic model of the sensor transmitter model. (c) DEVS atomic model of the sensor receiver model.

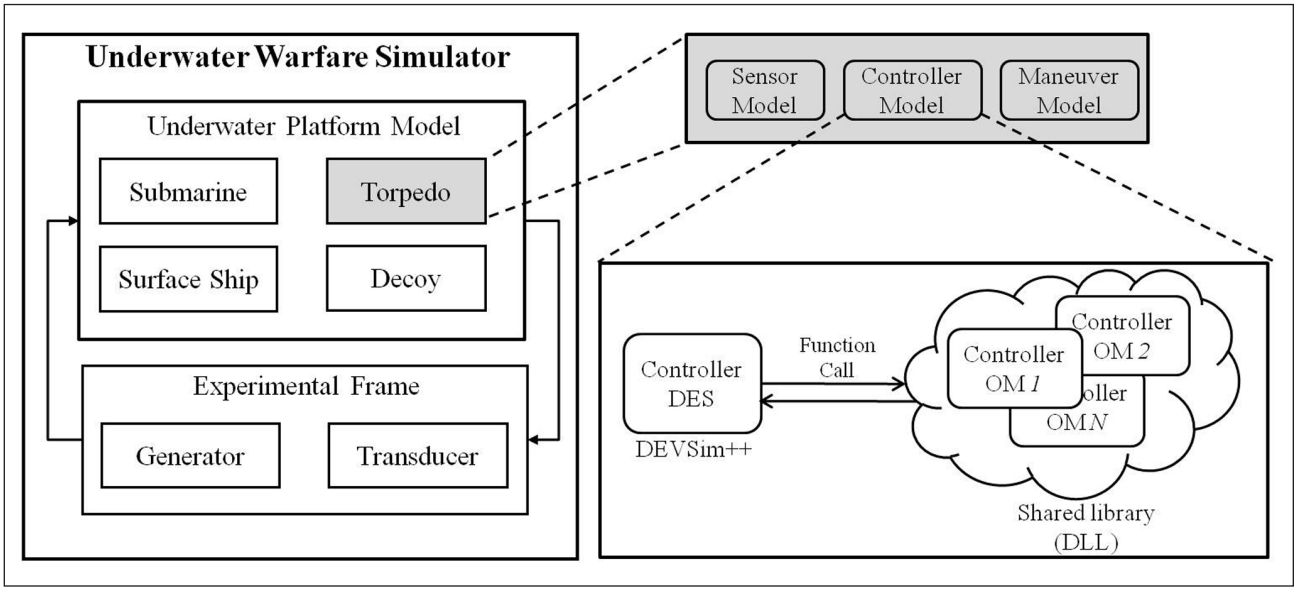

Figure 8. Structure of the overall model implementation.

Figure 9 shows the implementation of a torpedo maneuver model. The DES layer describes the maneuver behavior of the torpedo, and the OM layer describes the dynamic equation for the torpedo. Whenever the state of the torpedo maneuver model is MOVE, the DES layer calls the *CalculateNextPosition* function to the OM layer. The *CalculateNextPosition* function can be changed as the user applies alternative maneuver algorithms. The *CalculateNextPosition* function may realize various maneuver algorithms and tactics, such as uniform motion, parabolic motion, and accelerated motion. Each function is developed as a separate DLL, for example, Maneuver Equation 1 and Maneuver Equation 2 in Figure 9.

5. Model Simulation

Now that we have done the model implementation using C++, we will design the overall architecture of the underwater warfare simulation.

Figure 10 illustrates the high-level view of the discrete event simulation for underwater warfare. The underwater warfare simulation is not a live simulation, but a constructive simulation. Generally, a constructive simulation has various simulation parameters. We identify these parameters as the scenario parameters and enumerate the input parameters in Table 2. As our goal is to evaluate the simulation's MOEs, such as the survivability of the surface ship and the operational success rate in various situations, we will vary the input parameters in a later case study. The simulation result is represented in SIMDIS, which is a set of software tools that provide two- and three-dimensional interactive graphical and video displays of live and post-processing simulation, test, and operational data.[14]

The developed simulation allows for a statistical evaluation of underwater warfare system effectiveness through Monte Carlo simulation. The feature of Monte Carlo simulation depicted in Table 2 allows for random variations in certain parameters and simulated events to develop probabilistic assessments of system effectiveness. For example, the torpedo is launched randomly within the scenario guidelines and the reliability of the decoy is influenced by the normal random variable.

6. Case Study – Anti-torpedo Warfare Simulator

This section illustrates the anti-torpedo simulator developed for effectiveness analysis.

6.1 Torpedo Engagement

In this case study, the attacking platform and the target platform are the submarine and the surface ship, respectively. The anti-torpedo sonar decoys are used for countermeasures, and the heavyweight torpedo is applied, which is a submarine-launched torpedo sent toward the surface ship. The objective of the simulation is to measure the survivability of the surface ship as the MOE. The simulator is useful for increasing the survivability of the surface ship by applying various parameters in each model. The brief scenario for the developed simulator, illustrated in Figure 11, is as follows.

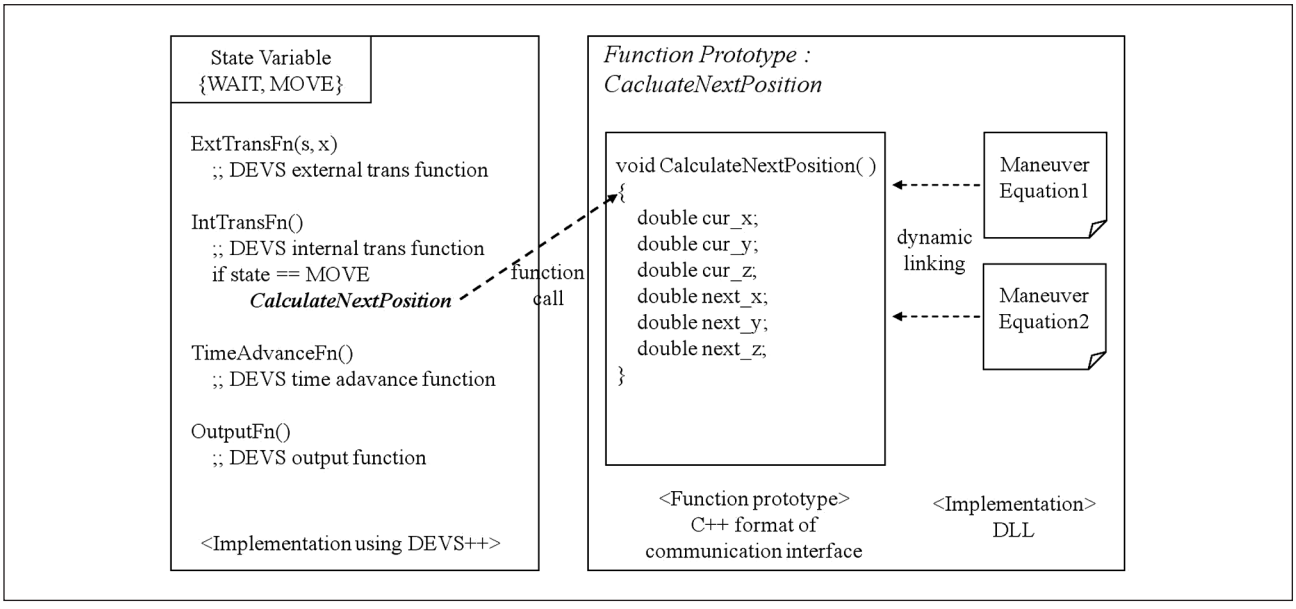

Figure 9. Implementation of the maneuver model of the torpedo.

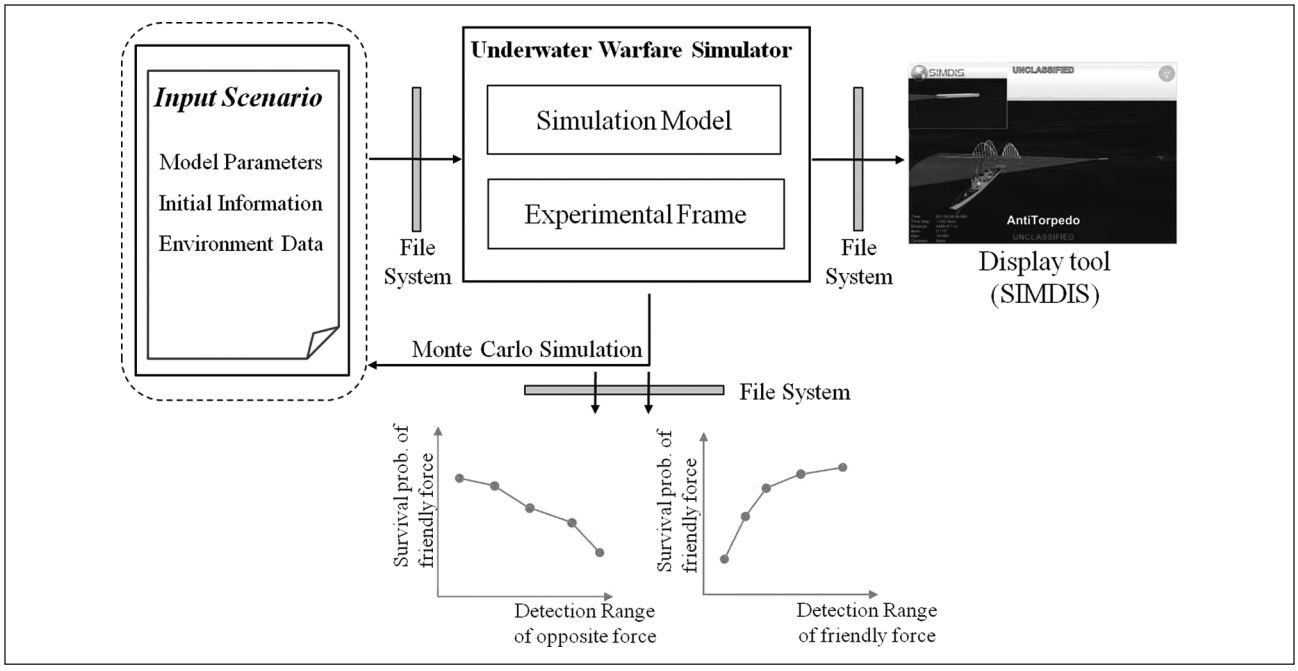

Figure 10. High-level view of the underwater warfare system.

1. A submarine launches a torpedo toward the detected surface ship.
2. The torpedo searches the threats through its own searching algorithm.
3. The surface ship detects the torpedo heading toward it, which is located in the boundary of the detection range.
4. The surface ship has only the decoy system to defend it against threats. After operating the decoy system, the surface ship makes a detour to evade the torpedo.
5. The torpedo detects the decoy instead of the surface ship and attacks the decoy.
6. The torpedo searches for other threats through its own searching algorithm.

The developed overall model architecture is described in Figure 12. The simulation model consists of four underwater platform models: a submarine, a surface ship, a torpedo, and a decoy. The attacking platform is a submarine, and the target platform is a surface ship. A heavyweight torpedo is used, which is a submarine-launched torpedo for surface ship targets. In addition, as countermeasures we consider anti-torpedo sonar decoys.

Before analyzing the effectiveness of the underwater warfare system, we must define the metrics that can be and are being used to evaluate effectiveness of the system. In this paper, the MOE can be analyzed by varying specified input parameters in Table 3. After running the simulator, we analyzed the results, which are described in Section 6.2. One hundred replications per scenario were tested for the statistical evaluation, and the total number of replications was 40,000.

6.2 Experimental Results

Figures 13–15 show the experimental results applied to the developed simulator. All experiments are applicable to statistical evaluation of effectiveness of the anti-torpedo combat system through Monte Carlo simulation. Experiment 1 is for assessment of anti-torpedo countermeasure effectiveness, and experiments 2 and 3 support the decision-making process for future equipment procurements. In Figures 13–15, the x-axis represents the detection range of the surface ship, and the y-axis shows the probability of the surface ship's surviving.

Experiment 1

The first experimental result shows the probability of the surface ship surviving according to the detection range of the surface ship and the decoy operating system depicted in Figure 13. Basically, the experimental result shows that a higher detection range results in a higher probability of survival. Another analysis is the survival probability according to the patterns of the decoy operating system. Four patterns of the decoy operating system are applied: they are depicted in Table 3. Comparing pattern 1 with pattern 2, the mobile decoy system performs better than the static decoy system, and comparing pattern 3 with pattern 4, the experimental result shows that the kind of decoys used at the front of the

Table 2. Input parameters for underwater warfare simulation.

Platform	Parameter name	Default value	Implications
Surface ship	Length, width	135, 5 (m)	The volume of the surface ship
	Speed	15 (knts)	The movement speed of the surface ship
	Revolution angle per second	3 (degree/s)	The revolution angle per second of the surface ship
	Detection range	3000 (m)	The detection range of the surface ship
Torpedo	Live time	2400 (s)	The live time of the torpedo
	Sonar beam angle	24 (degree)	The sonar beam angle of the torpedo
	Sonar beam steering angle	0 (degree)	The sonar beam steering angle of the torpedo
	Angle to launch	10 (degree)	The angle to launch from the attacking platform, which is influenced by the random variable
	Angle of sweep-forward	40 (degree)	The maximum turning angle when the snake search pattern is used
	Revolution angle per second	3 (degree/s)	The revolution angle per second of the surface ship
	Speed	18, 23, 35 (knts)	Low speed Middle speed High speed The different speed is applied according to the pattern of torpedo movement
Submarine	Length, width	230, 10 (m)	The volume of the submarine
	Speed	12 (knts)	The movement speed of the submarine
	Revolution angle per second	3 (degree/s)	The revolution angle per second of the submarine
	Detection range	15,000 (m)	The detection range of the submarine
Decoy	Motion type	Static	The motion types are mobile and static decoy
	Launch type	Rocket	The launch types are rocket and air pressed
	Num. of decoys	4	The number of decoys
	Speed of decoy	12 (knts)	The speed of the mobile decoys (ignored if the user does not use mobile decoys)
	Reliability	90 (%)	The probability of working normally (influenced by the random variable)
	Launch time	3 (degree/s)	The delay time after launch command
	Operation time	540 (s)	Time period from launch to expiration
	Source level	140 (dB)	The source level of the decoy

surface ship is greatly influenced by the survival probability. This experimental result illustrates that an effective way to create a successful mix of decoys is heavily dependent on the coordination of the timing and position to be released and the type of decoys. A successful anti-torpedo tactic should consist of the deployment of mixed decoys and the coordination with the surface ship's maneuver.

Experiment 2

Figure 14 shows the second experimental result. As with the first experimental result, when the detection range is higher, the probability of the surface ship surviving is higher. In addition, when the detection range is fixed, the probability of the surface ship surviving is higher with higher decoy speeds. In particular, in the case that the speed of the decoy is greater than 9 knots, the decoy speed does not have a significant influence on the surface ship's survival.

Experiment 3

The third experimental result, depicted in Figure 15, shows the survival probability according to the detection range of the surface ship and the decoy operating time. Like the

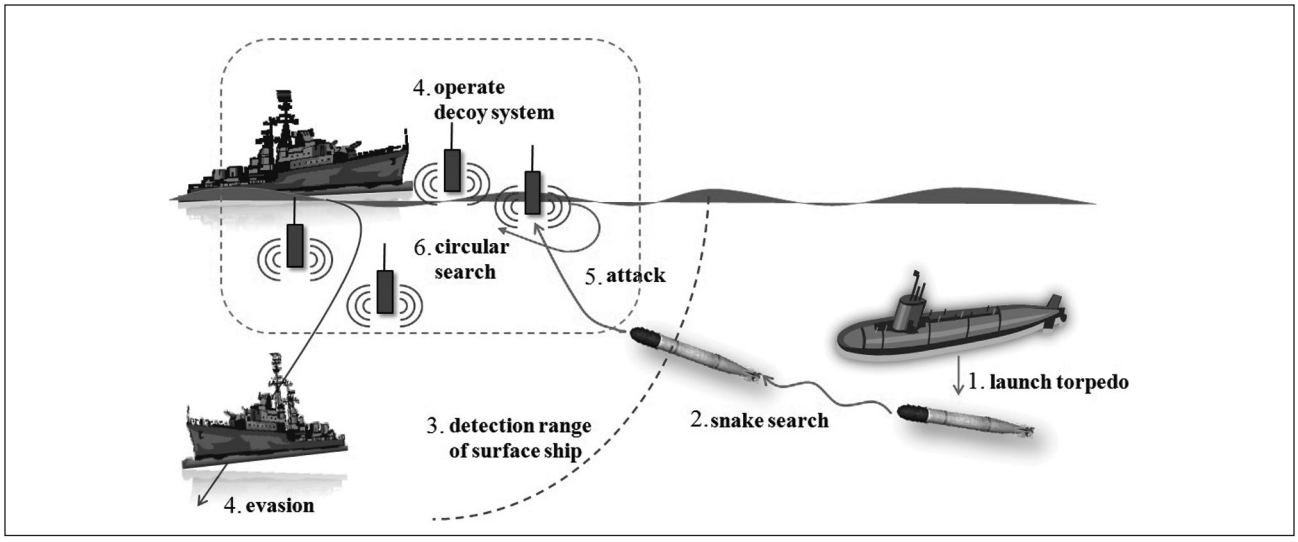

Figure 11. Brief scenario of anti-torpedo warfare.

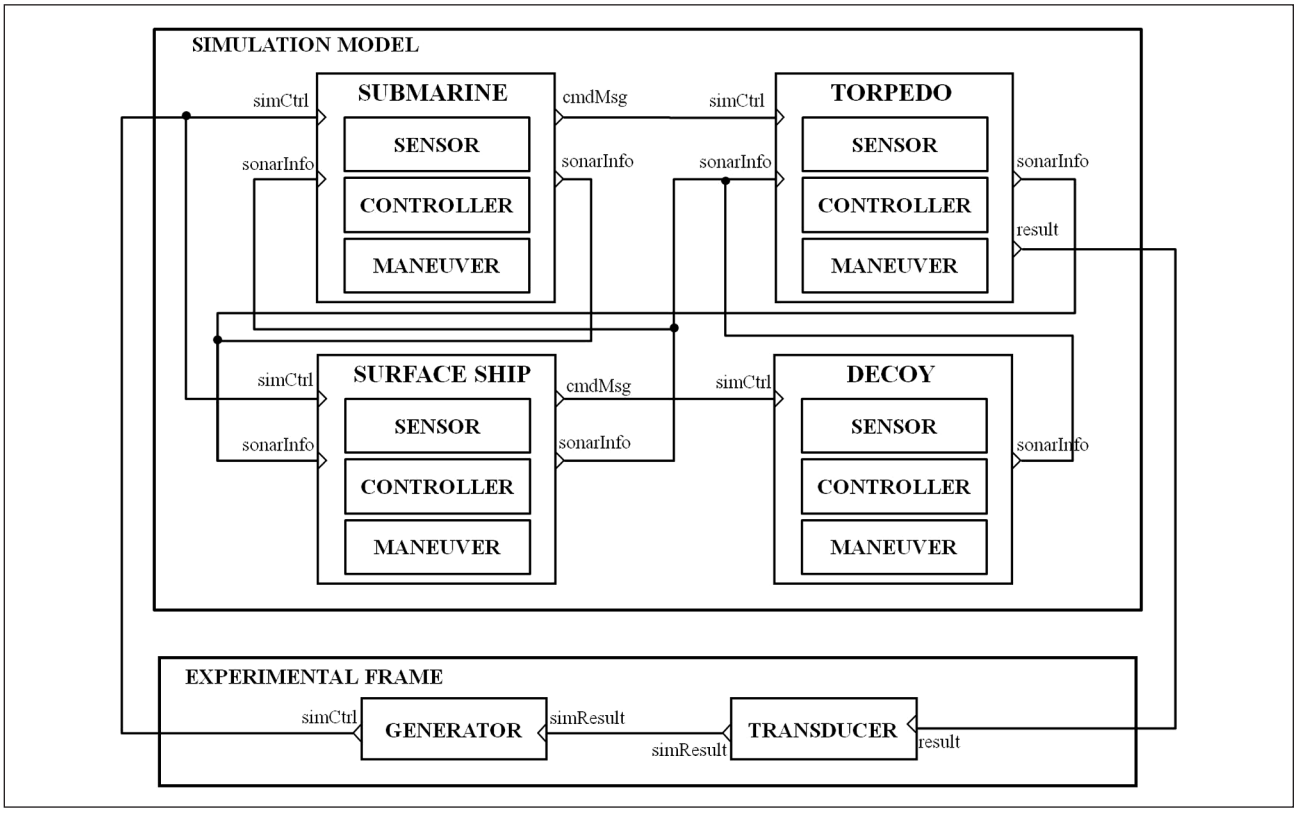

Figure 12. Overall model structure for underwater warfare.

second experimental result, when the detection range is fixed, the probability of the surface ship surviving is higher with longer operating time of the decoy. In the case that the speed of the decoy is longer than 360 seconds, the operating time of the decoy does not have a significant influence on the surface ship's survival. The second and third experimental results support the decision-making process for future equipment procurements.

Table 3. Virtual experiment design table.

Variable name	Variation cases	Implications
Detection range of surface ship	2000, 2500, 3000, 3500, 4000 (5 cases)	The different detection range results in differences in the survival of the surface ship.
	default: 3000 m	
Pattern of decoy operating system	From pattern 1 to pattern 4 (4 cases)	The different pattern results in differences in the survival of the surface ship. Pattern 1: four static decoys Pattern 2: four mobile decoys
	default: pattern 3	Pattern 3: two static decoys at the front of warship and two mobile decoys at the rear Pattern 4: two mobile decoys at the front of warship and two static decoys at the rear
Operating time of decoy	120, 240, 360, 480 540 s (5 cases)	The different operating time of the decoy results in differences in the survival of the surface ship.
	default: 480 s	
Speed of mobile decoy	3, 6, 9, 12, 15 knts (5 cases)	The different speeds of the mobile decoy result in differences in the survival of the surface ship.
	default: 12 knts	
Total	600 cells (4×5^3 cases)	No. of replications per case: 100 times

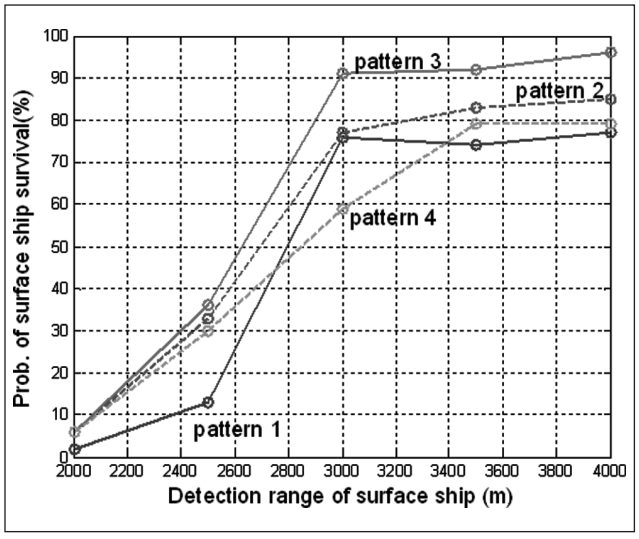

Figure 13. Experimental result for patterns of the decoy operating system.

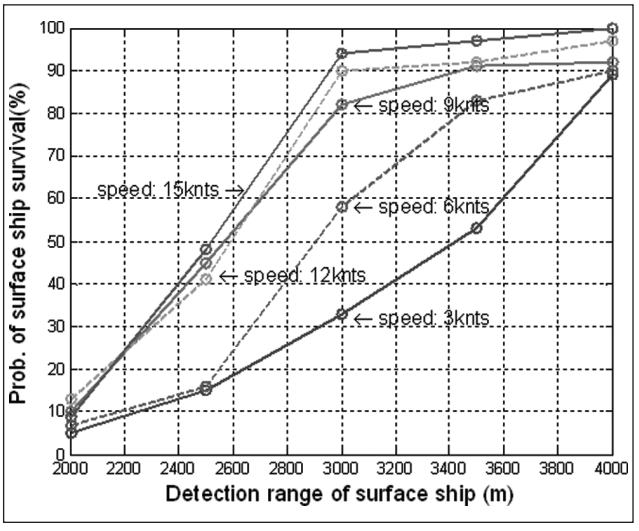

Figure 14. Experimental result for the speed of the mobile decoy.

There are some reasons that the experimental results from Figures 13–15 may be inaccurate. The input parameters are not real values, and external environmental factors, such as wind speed and ocean current, are not considered. Despite these limitations, we can perceive trends using the proposed simulator, and if users apply real values for input parameters, they can acquire more meaningful experimental results and the results enable military commanders to examine the likelihood of combat outcome prior the actual engagement; correspondingly, they can adjust the battle plan or use the information to devise future military tactics.

7. Conclusion

This paper proposes to design an underwater warfare model using the DEVS formalism to provide insights about how various factors, such as tactics and the performance of underwater weapons, influence the MOEs of the system. The developed simulation supports users in evaluating the

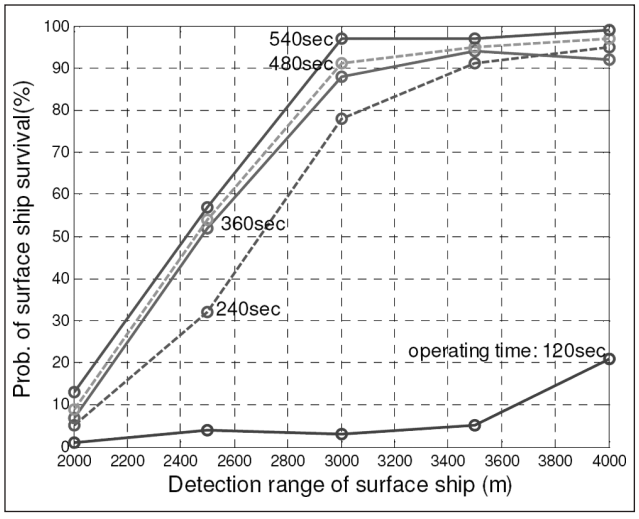

Figure 15. Experimental result for the operating time of the decoy.

effectiveness of underwater warfare systems through Monte Carlo simulation and assesses tactical development and anti-torpedo countermeasure effectiveness.

To overcome the disadvantages given in related works, we introduce the co-modeling methodology for flexible model architecture and use the methodology to model the underwater warfare system. In the underwater warfare model, change in detailed algorithms and dynamic equations, such as sensor algorithms, is more frequent than the change in the abstract behavior. Therefore, by separating abstract behavior and detailed algorithms, we can test candidate tactics or algorithms of underwater platforms with minimal modification of the model. Finally, we propose the generic three-part modeling methodology that provides the generic representations of underwater platforms. Each submodel is reusable for other platform models.

Using the proposed simulation, we can determine how various factors, such as tactics and the performance of underwater weapons, influence effectiveness of the system. In particular, in the case of simulation of a surface ship's evasion of a torpedo, we parameterize four factors, which are the detection range of a surface ship, the pattern of the decoy operating system, and the speed and operating time of the decoy. Experimental results support assessment of anti-torpedo countermeasure effectiveness and the decision-making process for future equipment procurements.

In this research, we did not consider external environmental factors, such as wind speed and ocean current, and we only modeled the passive sonar system. Therefore, it is not adequate to correctly handle multi-static acoustics within the underwater environment using multiple sonars. It requires much effort in modeling and developing multiple sonars and environmental factors, yet that is our goal for future works.

Funding

This work was supported by the Defense Acquisition Program Administration and the Agency for Defense Development (ADD) under contract UD080042AD, Korea.

8. References

1. Robinson S. Modes of simulation practice: approaches to business and military simulation. *Simulat Model Pract Theor* 2002; 10: 513–523.
2. Liang K-H and Wang K-M. Using simulation and evolutionary algorithms to evaluate the design of mix strategies of decoy and jammers in anti-torpedo tactics. *In: Proceedings of the 2006 Winter Simulation Conference, 2006*, pp.1299–1306.
3. Armo KR. The relationship between a submarine's maximum speed and its evasive capability. Master's Thesis, Naval Postgraduate School, *Monterey*, CA, 2000.
4. Cho D-Y, Son M-J and Lee K-Y. Analysis of a submarine's evasive capability against an antisubmarine warfare torpedo using DEVS modeling and simulation. *In: Proceedings of the SpringSim2007*, 2007, pp.307–315.
5. Kim TG and Park SB. The DEVS formalism: hierarchical modular systems specification in C++. *In: Proceedings of the European Simulation Multi-conference*, York, UK: June 1992, pp.152–156.
6. Park SC, Kwon Y, Seong K and Pyun J. Simulation framework for small scale engagement. *Comput Ind Eng* 2010; 59: 463–472.
7. Sung CH, Hong S-Y and Kim TG. Layered structure to development of OO war game models using DEVS framework. *In: Proceedings of SCSC2005*, July 2005, pp.65–70.
8. Zeigler BP, Praehofer H and Kim TG. Theory of modeling and simulation, (second ed.). London: Academic Press, 2000.
9. Seo K-M, Hong JH and Kim TG. DEVS-based underwater warfare simulation development for effectiveness analysis. *In: Proceedings of the Summer Simulation Multi-conference*, Ottawa, Canada, 2010.
10. Arlow J and Neustadt I. UML 2 and the unified process, (second ed.). Canada Addison Wesley, Canada, 2005.
11. Wainer G. CD++: a toolkit to develop DEVS models. *Software Pract Ex* 2002; 32: 1261–1306.
12. Hall SB, Zeigler BP and Sarjoughian HS. Joint MEASURE™: distributed simulation issues in a mission effectiveness analytic simulator. *In: Proceedings of the Simulation Interoperability Workshop*, Orlando, 1999.
13. Beazley D, Ward B and Cooke I. The inside story on shared libraries and dynamic loading. *Comput Sci Eng* 2001; 3: 90–97.
14. U.S. Naval Research Laboratory. *SIMDIS user's manual*. 2006.

Author Biographies

Kyung-Min Seo received his BS degree in electrical engineering from Pusan National University and his MS degree in electrical engineering and computer science (EECS) from the Korea Advanced Institute of Science and

Technology (KAIST), in 2006 and 2008, respectively. Currently, he is a PhD candidate at the System Modeling Simulation Labora-tory (SMSLAB) at the KAIST. His research interests include methodology for the M&S of hybrid systems and underwater warfare M&S.

Hae Sang Song received his MSD and PhD in Electrical Engineering from the KAIST. He worked for a couple of years on systems engineering at the Institute of Advanced Engineering (IAE) in 1999–2000. He also worked for a mobile venture company for about two years. He has been an associate professor at the Department of Computer Engineering at Seowon University since 2002. He co-works in the Systems Modeling Simulation Lab as a visiting researcher for defense M&S researches and projects. His major interest resides in modeling simulation, analysis, and control of discrete event dynamic systems.

Se Jung Kwon received his BS degree in Computer Science from the KAIST in 2009. He is currently a MS student in the Department of Electrical Engineering at the KAIST. His research interests include simulation algorithms for DES and DEVS execution environment.

Tag Gon Kim received his PhD in computer engineering with specialization in systems M&S from the University of Arizona, Tucson, AZ in 1988. He was an Assistant Professor in Electrical and Computer Engineering at the University of Kansas, Lawrence, KS from 1989 to 1991. He joined the Electrical Engineering Department, KAIST, Daejeon, Korea in Fall, 1991 and has been a Full Professor in the EECS Department since Fall, 1998. He was the President of The Korea Society for Simulation (KSS). He was the Editor-In-Chief for Simulation: Transactions for the Society for Computer Modeling and Simulation International (SCS). He is a co-author of the text book 'Theory of Modeling and Simulation', Academic Press, 2000. He has published about 200 papers in M&S theory and practice in international journals and conference proceedings. He is very active in defense M&S in Korea. He was/is a consultant for defense M&S technology at various Korea government organizations, including the Ministry of Defense, the Defense Agency for Technology and Quality (DTAQ), the Korea Institute for Defense Analysis (KIDA), and the ADD. He is a Fellow of SCS and a Senior Member of the Institute of Electrical & Electronics Engineers (IEEE).