ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/264004946

DEVS-based combat modeling for engagement-level simulation

Article *in* SIMULATION: Transactions of The Society for Modeling and Simulation International · July 2014

DOI: 10.1177/0037549714532960

CITATIONS
19

READS
401

4 authors, including:

Kyung-Min Seo
Daewoo Shipbuilding and Marine Engineering
25 PUBLICATIONS **159** CITATIONS
SEE PROFILE

Changbeom Choi
Handong Global University
35 PUBLICATIONS **123** CITATIONS
SEE PROFILE

Tag Gon Kim
Korea Advanced Institute of Science and Technology
182 PUBLICATIONS **1,383** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Efficient Ranking and Selection View project

Project IoT in noisy Environment View project

All content following this page was uploaded by Kyung-Min Seo on 18 July 2014.

The user has requested enhancement of the downloaded file.

SIMULATION
http://sim.sagepub.com/

DEVS-based combat modeling for engagement-level simulation
Kyung-Min Seo, Changbeom Choi, Tag Gon Kim and Jung Hoon Kim
SIMULATION 2014 90: 759 originally published online 12 June 2014
DOI: 10.1177/0037549714532960

The online version of this article can be found at:
http://sim.sagepub.com/content/90/7/759

Published by:
SAGE
http://www.sagepublications.com

On behalf of:
Society for Modeling and Simulation International (SCS)

THE SOCIETY FOR MODELING & SIMULATION INTERNATIONAL

Additional services and information for *SIMULATION* can be found at:

Email Alerts: http://sim.sagepub.com/cgi/alerts

Subscriptions: http://sim.sagepub.com/subscriptions

Reprints: http://www.sagepub.com/journalsReprints.nav

Permissions: http://www.sagepub.com/journalsPermissions.nav

Citations: http://sim.sagepub.com/content/90/7/759.refs.html

>> Version of Record - Jun 30, 2014

OnlineFirst Version of Record - Jun 12, 2014

What is This?

Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
2014, Vol. 90(7) 759–781
© 2014 The Society for Modeling and Simulation International
DOI: 10.1177/0037549714532960
sim.sagepub.com
SAGE SCS

DEVS-based combat modeling for engagement-level simulation

Kyung-Min Seo[1], Changbeom Choi[2], Tag Gon Kim[2] and Jung Hoon Kim[3]

Abstract
This paper presents a modeling method to demonstrate engagement-level military simulation which includes few combat objects, or entities. To this end, the paper, on the basis of the discrete event system specification (DEVS) formalism, centers on two ideas: (1) a combat entity's model structure at the composition level; and (2) behavioral delineation of the entity's elementary component. In detail, we classify the combat entity model into platform and weapon models and create six groups of the model categorized by two dimensions: three activities and two abstractions. And the elementary component in the group interprets an engagement scenario as a flow of executable tasks, which are expressed by DEVS semantics. The stated structures and semantics provide intuitive appeal, reducing the effort required to read and understand the model's behavior. From the combat experiments, we can gain interesting experimental results regarding engagement situations employing underwater weapons and their tactical operations. Finally, we expect that this work will serve an immediate application suited to various engagement situations.

Keywords
Defense, combat entity, discrete event system, formalism, model design, model implementation, compositional reusability, effectiveness analysis

1. Introduction

As modeling and simulation (M&S) has been widely utilized in the defense communities,[1,2] the defense M&S field has developed various levels of combat models (i.e. the *theater*-, the *mission*-, the *engagement*-, and the *engineering*-level models), which are determined by modeled objects and scenarios of interests.[3] Out of these, the *engagement*-level model, which is our interest in this study, focuses on duel level or few-on-few engagement, e.g., missile versus warship or aircraft versus aircraft. Improvement of existing tactics or new tactical development of combat entities is decided and evaluated on the *engagement*-level; many modeling activities in the recent decade have focused on this level.[4–6]

In most M&S development, a modeler chooses the modeling formalism that fits the system context and the modeling objective due to advantages of the formal method.[7] The formal method, which has a mathematical basis, provides the means of precisely defining notions like consistency, completeness, correctness, and verification.[8] Therefore, with the formal method, the modeler can specify, develop, and verify the modeled system in a systematic, rather than an ad hoc manner.[9,10] Among various types of formalisms, the discrete event system specification (DEVS) formalism, introduced by Zeigler,[11] is a set-theoretic specification of discrete event systems. As combat systems are characterized as discrete event systems according to the system taxonomy,[12] and the DEVS formalism also has many particular features to specify discrete event systems, it has been widely used for *engagement*-level combat modeling.[7,13,14] In this respect, DEVS-based combat modeling is a key issue of this study.

In recent years, we have conducted several studies on *engagement*-level combat modeling using the DEVS formalism. DEVS-based combat modeling for underwater

[1]S31 R&D Institute, Daewoo Shipbuilding & Marine Engineering Co., Ltd, Republic of Korea
[2]Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Republic of Korea
[3]Naval Systems R&D Institute, Agency for Defense Development, Republic of Korea

Corresponding author:
Kyung-Min Seo, S31 R&D Institute, Daewoo Shipbuilding Marine Engineering Co., Ltd, 26, Eulji-ro 5-gil, Jung-gu, Seoul 100-210, Republic of Korea.
Email: kmseo.kumsung@gmail.com

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

warfare was proposed in 2010,[15] and a three-part model design of underwater vehicles and effectiveness analysis of an anti-torpedo warfare was conducted in 2011.[16] In spite of our efforts for DEVS-based modeling, these studies need some improvements. Specifically, the previous models assumed that all the combat entities, which corresponded to complete vehicles or machines participating in an engagement scenario, had identical structures and behaviors regardless of their different types. Besides, the models only targeted at a specific engagement – namely, underwater warfare. Accordingly, this study aims to refine these problems of our previous models and generalize them to various engagement scenarios.

Leaving aside our previous studies, many other researchers have developed their own *engagement*-level combat models using the DEVS formalism. For example, some researchers studied human behavior for computer-generated forces;[17] others developed unmanned aerial vehicle (UAV) models for developing a route tactic.[18] Also, several researchers designed underwater warfare models.[19,20] Notwithstanding their practical contributions, there remain some shortcomings. In other words, they did not strengthen particular advantages of the DEVS formalism when they utilized the formalism. Some researchers disregarded hierarchical and compositional model design for combat entities, while others oversimplified their tactical behaviors despite the fact that the above-mentioned expressions can be formulated with the DEVS formalism.

The focus of this study, therefore, is to suggest a theoretical basis of DEVS-based combat modeling and to develop it in a formal and effective way. The central issue of *engagement*-level combat modeling is how we abstract and represent a combat entity, such as an aircraft, a submarine, a missile, or a torpedo. Due to the different tactical behaviors, we sort the combat entity into two types: (1) a platform, which is a vehicle on which weapons are mounted, such as a tank, a submarine, or an aircraft; and (2) a weapon that is loaded onto the platform, such as a missile, a torpedo, or a decoy.

The combat entity, which can be either structurally or behaviorally categorized, is modeled by the DEVS formalism. From a structural perspective, since the combat entity has three kinds of common and core activities, i.e., movement, detection, and decision (or command and control),[7] we classify it into three separate components: maneuver, sensor, and controller models. With hierarchical expressions and coupling schemes provided by the DEVS coupled model, the combat entity model are organized hierarchically and interacts with component models internally or other entity models externally. As another perspective, i.e., a behavioral view, we regard an engagement scenario as a flow of executable tasks, and the task flow is expressed by semantics of the DEVS atomic model. This behavioral semantics provide intuitive appeal, reducing the effort required to read and understand the component model. Moreover, our DEVS-based modeling enhances compositional reusability, which means that component models can be composed to create larger models. For example, a well-described maneuver model of a torpedo model can be reused to create other warship or submarine models.

In summary, the objective of this study is to propose a DEVS-based modeling for *engagement*-level combat simulation. To this end, we propose three factors for modeling technique: (1) overall model description, (2) model design with the DEVS formalism, and (3) model implementation. To prove the efficiency of the proposed DEVS modeling method, we illustrated compositional reusability within a specific engagement scenario – namely, anti-submarine warfare of a friendly warship. In addition, simulation results of various experiments show effectiveness analysis, such as how the factors influence the measures of effectiveness (MOEs) of the engagement.[7] The successful execution of this study greatly describes combat entities with the DEVS formalism, and finally it offers an immediate practical application for testing new tactical development or analyzing weapon performance with various combat scenarios.

This study contains seven sections. Section 2 describes two theoretical grounds: taxonomy of an *engagement*-level combat system and advantages of the DEVS formalism. Section 3 introduces previous studies and compares them as part of a literature review. Sections 4–6 explain the proposed combat modeling method in the following order: overall model structure, DEVS-based model design, and model implementation. Section 7 illustrates the experimental results to prove the efficiency of this work and discuss them, and we conclude our study in Section 8.

2. Problem definition

Before moving to the central part of our work, it is imperative to clarify two key points for developing our arguments: which taxonomy *engagement*-level combat systems belong to and what formal modeling method expresses the systems effectively.

2.1. Taxonomy of engagement-level combat system

There are several types of systems distinguished by time and state spaces: continuous, discrete time, digital, and discrete event systems.[21] We start our arguments with an engagement scenario, which represents a battle with a few combat entities, to distinguish the system type of *engagement*-level combat systems.

Figure 1 illustrates a simplified scenario in a flow chart form. If a platform detects some threats on a scout, it stays in contact with the threat to identify whether it is a target or not. If the threat is regarded as a target, the platform comes up to the target. When the target is within striking

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

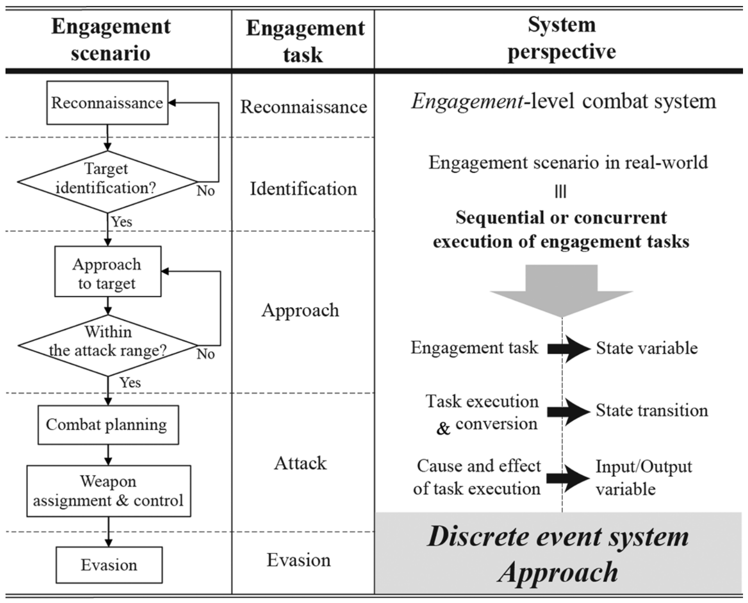


Figure 1. System taxonomy of *engagement*-level combat system regarding engagement scenario.

distance, the platform fires and guides its weapons, and finally, it makes a detour operation. Such engagement scenario can be summarized by defining several tasks necessary to fulfil the engagement mission.

In Figure 1, the simplified scenario has five kinds of engagement tasks: reconnaissance, identification, approach, attack, and evasion. Therefore, the engagement scenario in the real world can comprise the sequential and concurrent execution of these tasks. From a systematic view, tasks are seen as finite and discrete state variables of the combat system; conversions and executions of tasks are regarded as state transitions; and causes and effects of task executions and conversions are expressed as unexpected input/output (I/O) events of the system, which are interchanged with other combat entities. In this context, *engagement*-level combat systems are regarded as discrete event systems with discrete state variables for engagement tasks and I/O events that can occur at any time. In like manner, many researchers have considered the *engagement*-level combat systems as discrete event systems.[22,23]

As illustrated in our introduction, there are various types of formalisms, including the DEVS formalism, the finite state machine, the cellular automata, the Petri-Net, the system dynamics, etc. Of these, the DEVS formalism describes discrete event systems with sound semantics founded on a system theoretic basis.[24] The formalism provides two types of specifications: an atomic model from which larger ones are built and a coupled model for the hierarchy structure of overall models. From the combat modeling perspective, two types of the DEVS formalism are suitable for modeling the engagement scenario between two opposing combat entities. For example, the engagement scenario is decomposable into two opposing combat entities, and each combat entity is also decomposable into multiple sub-entities, depending on the modeling objectives. This decomposition of the scenario resembles the hierarchy structure of the system modules, i.e., the DEVS coupled model; the sub-entities behaves basically, which is the atomic model of the DEVS formalism. This is the primary reason that we use the DEVS formalism for *engagement*-level combat modeling.

2.2. Combat modeling with the DEVS formalism

As the DEVS formalism is a general formal method, it fundamentally satisfies common advantages of the formal method, such as consistency,[25] completeness[26] or verification.[27] We shall now summarize and explain several particular advantages of the DEVS formalism. For a more clear understanding, we classify the advantages in terms of two aspects: theoretical and applicable perspectives, which are depicted in Figure 2.

In the theoretical perspective, the DEVS formalism, first, enables the modular and hierarchical design by using I/O ports and coupling schemes on the basis of system theoretic principles. This allows very complex models to be built by connecting different DEVS models, either atomic or coupled models, in a hierarchical manner. Also, it supports scalability and reusability through the use of a DEVS model as a component in another DEVS model.[28] For example, Kwon et al. expanded existing DEVS models by adding new jammer DEVS models to develop a mixed tactic with decoys and jammers.[29]

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

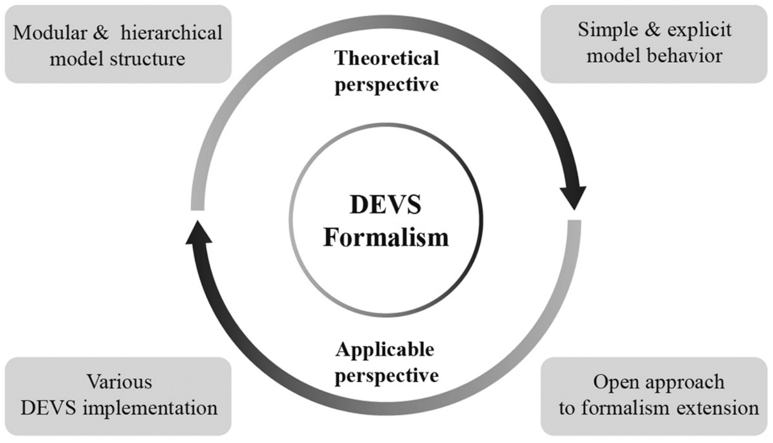

Modular & hierarchical model structure
Theoretical perspective
Simple & explicit model behavior
DEVS Formalism
Applicable perspective
Various DEVS implementation
Open approach to formalism extension

Figure 2. Particular advantages of DEVS formalism from two perspectives.

Next, the DEVS formalism provides simple but clear semantics for the basic model behavior. The basic model, or the DEVS atomic model, relies on only three sets and four functions. With these semantics, the formalism can specify a specific state at any point of time as well as interact with other models with I/O events which are caused by state transitions. Thus, the formalism provides a complete and transparent representation of an object to be modeled, reducing the effort required to read and understand it. Due to the expressive power of the DEVS formalism, many researchers have attempted to apply the DEVS formalism to a specific simulation environment or transduce other formal methods to the DEVS formalism.[30–32]

Furthermore, the DEVS formalism supports an open approach to formalism development, allowing the researcher to explore extended or specialized formalism.[25] For example, Barros proposed the dynamic structure DEVS (DSDEVS) formalism,[33] which allows changes in model structure during execution. Chow proposed the parallel DEVS (P-DEVS) for parallel execution benefits.[34] Hong et al. proposed the real time DEVS (RT-DEVS) for executing DEVS models within a real-world environment.[35] From a combat modeling perspective, Sung and Kim's study represents a formalism extension for a practical application.[36] Combat modeling is difficult because it requires complex knowledge backgrounds, e.g., defense domain knowledge and M&S skills. They proposed a collaborative modeling methodology to develop a domain-specific discrete event system,[36] and their study forms the foundation of our DEVS-based combat modeling.

Finally, the DEVS formalism presents an explicit separation between model specification and its implementation, or simulation development. In other words, implementing DEVS models is easily achievable by utilizing an implementation framework supporting the DEVS formalism, such as DEVSim++,[37] DEVSJava,[38] or CD++,[39] etc. Among them, the proposed DEVS models in this study will be implemented in the DEVSim++. In addition, many DEVS implementations also mean that the DEVS formalism has become universally recognized for discrete event systems.

Synthetically, the above-mentioned advantages indicate that the DEVS formalism gives not only the power of formal rigor but also enables models' practical application to real-world discrete event systems. Moreover, modeling a discrete event system is a key of combat modeling, and the DEVS formalism, which specializes in expressing a discrete event system, is well-suited for describing combat models. In the following section, we introduce some previous works for DEVS-based combat modeling and discuss their strengths and weaknesses.

3. Literature review

Some commercial tools for *engagement*-level combat modeling, e.g., ODIN,[40] virtual maritime system,[41] or BRAWLER,[42] are already available in the market. The greatest advantage of such tools is that they realize detailed physical modeling of combat entities, such as kinetics, acoustic signal modeling, or shape modeling. Nonetheless, the problems with these tools are that a user can only utilize them within the scope of their provided functions.[43] Therefore, the creation of various engagement scenarios and environments is only performed within a limited scope. This is in contrast to DEVS-based model development, which supports clear model semantics from structural and behavioral aspects. Since these commercial tools are not our comparative targets in this study, we focus on the DEVS-based combat models in academic areas and compare them.

Table 1 describes several previous studies for *engagement*-level combat modeling using the DEVS formalism.[17–20] We summarize their characteristics concerning DEVS

Table 1. Comparison with previous works for *engagement*-level combat modeling using the DEVS formalism.

Previous study	Military force	Level of model	Theoretical perspective (model structure)	Theoretical perspective (model behavior)	Practical application
Andrien et al.[17]	Armed force	Entity-level	All activities for a combat entity model were integrated in only one model.	Model behaviors for a composite mission were classified with course of tasks.	It has flexibility due to separation between a scenario and a combat entity.
Moreno et al.[18]	Air force	Entity-level	An air defense unit was divided into detection radar, tracking radar, and missile models.	Decision making and moving behaviors were ignored.	It seems to be underused except as a specific engagement.
Cho et al.[19]	Naval force	Entity-level	All activities for a combat entity model were integrated in only one model.	A combat entity model covered detailed behaviors for a mission.	It seems to be underused except in a specific engagement.
Park et al.[20]	Naval force	Entity-level	Two kinds of components were proposed for a combat entity: physical and logical parts.	Detailed tactical descriptions were not expressed in the combat entity model.	Physical and logical parts of the combat entity model can be reused for other applications.

advantages illustrated in Figure 2. The common strength of these studies, though not stated in Table 1, is that they drew meaningful simulation results in each military domain, e.g., UAVs' path planning or a submarine's evasive capability. All the studies concentrate on entity-level modeling that describes an individual combat entity rather than unit-level modeling to aggregate several entities into a higher object. Since an *engagement*-level combat modeling focuses on duel level or few-on-few engagement, most studies prominently featured utilizations and interactions of individual combat entities.

Despite these contributions, their studies still warrant some improvements. From the structural perspective, Andrien et al. and Cho et al. took a combat entity to be a whole physical and cognitive part.[17,19] Specifically, modeling of a combat entity is not decomposable any further; this means that the combat entity model performs all activities, such as moving, sensing, or decision making, in one integrated model. In this case, each activity is processed in one state of the combat entity model. Since the model can simulate only one state at a certain time and processes all the states sequentially, it cannot perform multiple activities concurrently. Moreover, this unified modeling method for a combat entity has a weakness in flexibility of model composition. For instance, if a modeler wants to improve an alternative evasive tactic for a submarine, he/she must modify the submarine DEVS model wholly, not partially. This is the typical misuse of the DEVS formalism. In contrast, Moreno et al. divided an air defense unit model into detection radar, tracking radar, and missile models,[18] and Park et al. departmentalized a combat entity depending on physical and logical attributes.[20]

In terms of the model behavior, some researchers could not tactically describe a combat entity. *Engagement*-level combat modeling needs to describe logical activities such as decision, command, and control as well as physical activities. However, unfortunately, Moreno et al. oversimplified a combat entity model – their DEVS model did not explain how to approach the target or how to launch weapons.[18] Park et al. also modeled a combat entity with minimum-level tactical activities despite separation of a logical part from the whole model.[20] As a consequence of oversimplifying descriptions, the combat entity model can simulate only a very low variability of the engagement scenarios, which gives rise to some insufficiencies related to the realism of an engagement scenario. Additionally, it is difficult for anyone but the modeler to understand oversimplified models exactly. On the other hand, Andrien et al. and Cho et al. proposed good modeling approaches for describing model behaviors.[17,19] The former defined a set of tasks allowing the goal, and their execution is represented by a DEVS atomic model; the latter described an engagement scenario using the unified modeling language (UML) before DEVS modeling.[9]

To sum up, the previous studies suffer from either inefficient structural model design or insufficient representation of an engagement scenario despite the use of the DEVS formalism. Due to the weaknesses of the studies, their studies have a better chance of becoming one-time research, remaining underutilized. Therefore, the focus of this study is to make the best use of the DEVS formalism, and to overcome these disadvantages and suggest some empirical results.

4. Overall model structure

In this section, as a top-down approach for effective description, we begin our proposed work from an overall model structure.

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

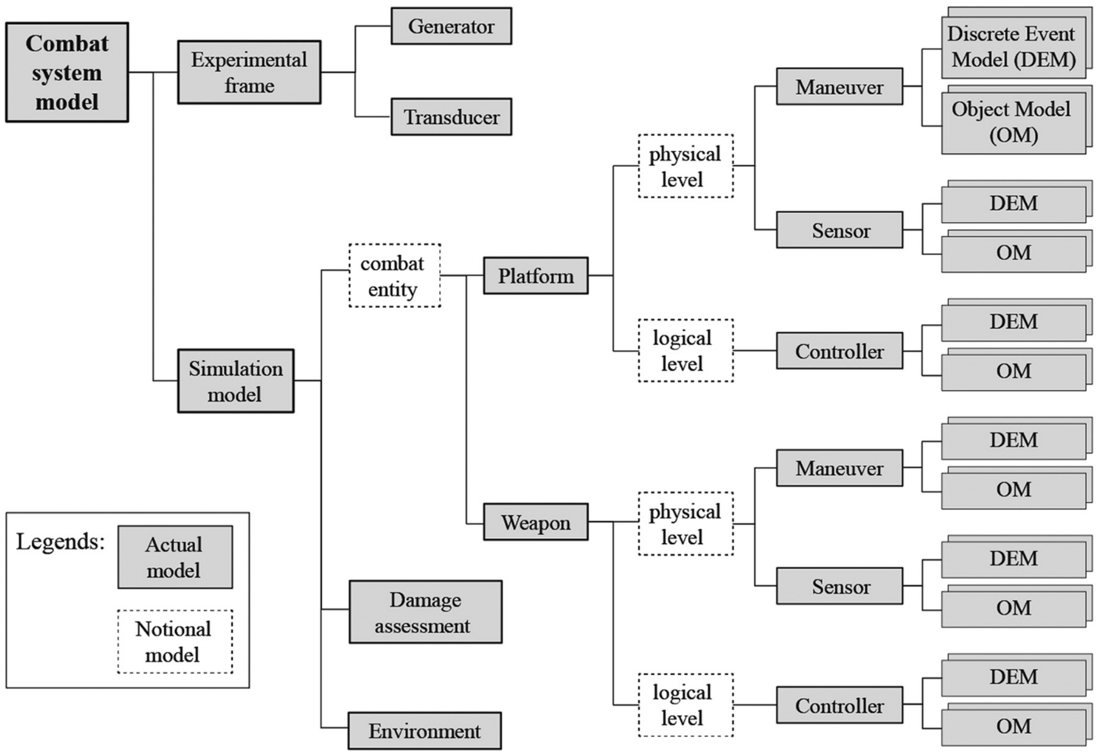


Figure 3. Overall model structure of combat system model.

Figure 3 represents the decomposition tree for the overall model structure which systematically organizes a family of models. In Figure 3, shaded solid-line boxes are practically designed models; transparent dotted-line boxes indicate notional models for easy comprehension. For example, a combat entity is a conceptual model and it is realized as a platform or a weapon model. In addition, the left-most side box is the superordinate category of the overall model, which means that a combat system model is outermost and the top layer of the overall model structure. The combat system model is decomposed into sub-models that perform the specific roles, keeping to the right side.

The combat system model is basically divided into an experimental frame and a simulation model. The simulation model represents a target system that a modeler is interested in modeling; the experimental frame specifies the conditions under which the simulation model is observed or experimented with, e.g., generation of the simulation models' inputs or collection of the model's outputs.[24] A clear separation of the simulation model and the experimental frame enables the application of alternative engagement scenarios without amendment of the simulation model. Since our focus in this study is the simulation model, and the *engagement*-level battle is usually focused on combat entities' tasks and accomplishments, we concentrate on the simulation model, especially a combat entity model.

In a hierarchical fashion, the simulation model also consists of several lower models: two or more combat entity models, a damage assessment model, and an environment model. The damage assessment model evaluates the engagement situation, (e.g., the offensive combat entity model attacks the target or the defensive model defends against the attack); the environmental model reflects environmental effects such as weather patterns, ambient temperature, artificial features and more. As mentioned above, since most *engagement*-level combat modeling features utilization and interactions of combat entity models, effective modeling of the combat entity is the central issue of our study.

Basically, the combat entity has three kinds of core activities: movement and detection for physical activities and decision for a logical activity. Since these activities are performed simultaneously as well as sequentially during engagement, we horizontally classify the model into three separate components: a maneuver, a sensor, and a controller models. This three-part modeling method was proposed in our previous work already,[16] and we refine the basic concept of the previous work to fit the context of this study. The sophisticated point compared with our previous work is that we classify a combat entity model into two types practically: a platform model for a tank, a submarine, or an aircraft and a weapon model that is for a missile, torpedo, or decoy loaded on the platform model. The principle reason for two types of categorization is logical behavioral

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

difference. The platform model decides tactical operation, commands the order, and controls launched weapons if necessary, while the weapon model executes preset tactical rules or it can be controlled by the controllable platform. This causes different model behaviors between them even though the modeling structures of both are identical.

Next, we move the focus to the model abstraction levels affected by the collaborative modeling method.[36] The combat modeling is difficult because the modeling requires complex knowledge background: that is, the defense-domain knowledge as well as the M&S knowledge at the same time.[7] To be specific, each sub-model, i.e. the maneuver, the sensor, and the controller model, is modeled into two levels in terms of a layered structure: (1) a discrete event model (DEM) layer for the M&S knowledge and (2) an object model (OM) layer for the defense-domain knowledge. The DEM layer represents the abstract behavior of an object using the DEVS formalism, and it is suitably employed to describe models macroscopically. For microscopic modeling, we develop the OM layer to represent detailed behavior of the same object,[44] which is non-decomposable and alternative. For example, the DEMs of the controller model perform the engagement tasks, described in Figure 1, according to the event sequences, whereas the OMs conduct detailed and individual actions to fulfill the tasks such as identification function or weapon control algorithm.

To sum up, we suggest a model structure of a combat situation to support an understanding of the brief model construction and implied relationships between upper models and their components. Naturally, there is no information concerning how to map the upper model on its components or detailed model descriptions about how it works. We will describe these viewpoints with the formal specification, i.e., the DEVS formalism, in the following section.

5. DEVS-based model design

This section explains DEVS representations of the combat system model. As DEVS models represented by the set-theoretic specification can be easily turned into graph diagrams, we use DEVS diagrams for more straightforward understanding (from now on, in the text we use *italics* for overall DEVS specifications expressed in the following notation and figures). Further information on the DEVS diagram can be found in Song et al.'s study.[45]

Figure 4 is the top-level diagram for the *Combat system model*. For modeling of an engagement scenario, one or more friendly and hostile combat entities, specified as platforms or weapons, are necessary; therefore, the *Combat system model* contains multiple combat entity models. In Figure 4, several *Platform* and *Weapon* models are comprised in combat entities with the unique subscript behind the word 'Platform' or 'Weapon'. In comparison of the two models, some I/O ports differ due to their particular behaviors. For example, the *Platform* model needs I/O ports for launching weapons and guiding them (i.e. *guidance_info, wp_launch*, and *wp_guidance*); the *weapon* model has different I/O ports to be controlled by the platform (i.e. *entity_gen, wp_guidance*, and *guidance_info*). In this context, we inform that the *Combat system model* in Figure 4 includes *weapon* models for being controlled, not fire-and-forget. The focus of this study is to represent the *Simulation model* that contains several combat entities, a *Damage assessment*, and an *Environment* models with the DEVS formalism. With these issues in mind, in the following sub-section, we first take a look at the *Platform* model design which is a type of the combat entity.

5.1. Platform model design

Suppose the situation in which the *Platform* model tracks a target. The *Platform Controller* model decides the appropriate tactic for tracking and sends a command order (e.g. how to approach to the target) to the *Platform Maneuver* model. The *Maneuver* model receives the command order and maneuvers depending on the command. Separately, the *Platform Sensor* model detects the target and sends the detected information to the *Platform Controller* model. The *Controller* model takes a new decision operation on the basis of the detected information. This process is repeated in the *Platform* model during simulation. It is only influenced by other models through interfaces that mean I/O relations. Accordingly, this feature leads to enhanced modularity and encapsulation of the sub-model. The following notations represent the DEVS coupled description of the *Platform* model, and Figure 5 shows its diagram.

Notation 1. DEVS Coupled description of Platform model:

$\boldsymbol{CM_{Platform}} = \boldsymbol{<X, Y, \{M_i\}, EIC, EOC, IC, Sel>}$,
X = {"*scen_info*", "*engage_result*", "*move_result*", *env_info*", "*guidance_info*"}
Y = {"*move_result*", "*wp_launch*", "*wp_guidance*"}
$\{M_i\}$ = {*Sensor, Controller, Maneuver*}
EIC = {($CM_{Platform}$.*scen_info, Sensor.scen_info*),
($CM_{Platform}$.*scen_info, Maneuver.scen_info*),
($CM_{Platform}$.*engage_result, Sensor.engage_result*),
($CM_{Platform}$.*engage_result, Controller.engage_result*),
($CM_{Platform}$.*engage_result, Maneuver.engage_result*),
($CM_{Platform}$.*move_result, Sensor.move_result*),
($CM_{Platform}$.*env_info, Sensor.env_info*),
($CM_{Platform}$.*env_info, Controller.env_info*),
($CM_{Platform}$.*env_info, Maneuver.env_info*),
($CM_{Platform}$.*guidance_info, Controller.guidance_info*)}
EOC = {(*Maneuver.move_result*, $CM_{Platform}$.*move_result*),
(*Controller.wp_launch*, $CM_{Platform}$. *wp_launch*),
(*Maneuver.wp_guidance*, $CM_{Platform}$.*wp_guidance*)}

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

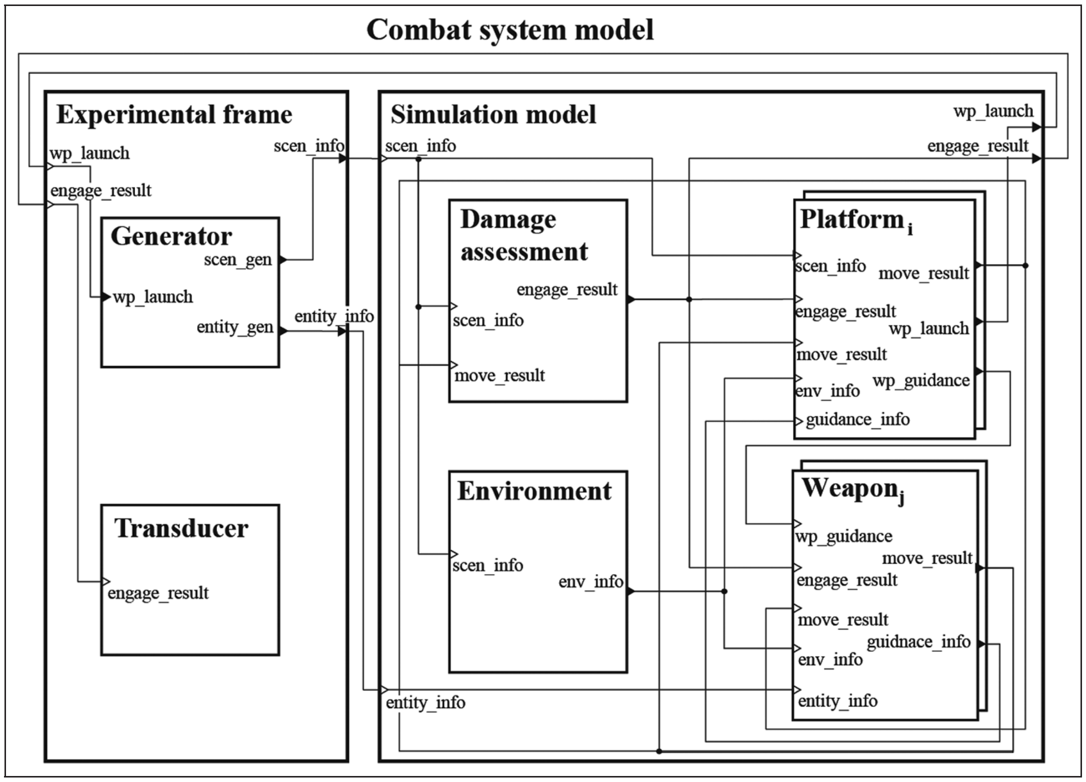


Figure 4. Graphical notation of *Combat system model* using DEVS diagram.

IC = {(Sensor.threat_info, Controller.threat_info), (Controller.move_cmd, Maneuver.move_cmd), (Maneuver.move_finished, Controller.move_finished), (Maneuver.fuel_exhausted, Sensor.fuel_exhausted), (Maneuver.fuel_exhausted, Controller.fuel_exhausted)}
Sel({Sensor, Controller}) = Controller

In the following sub-sections, we describe the three component models, i.e., the *Controller*, the *Maneuver*, and the *Sensor* models, in detail.

5.1.1. Controller model design. The *Controller* model performs tactical decision-making processes. It takes on the role of dynamic decision making under some uncertainty. The major tasks to be performed or achieved for engagement are illustrated in Table 2.

As described in Figure 1, these tasks are carried out concurrently or sequentially, and continuative execution of tasks can be represented by the discrete event model. To be specific, tasks are described by single state or an integrated state variable, and the transitions between states indicate conversion of tasks. In addition, execution of tasks is performed within the relevant state. We accomplish these operations by DEM and OM layers. The DEM layer describes an arrangement of tasks and tasks' conversion; the OM layer represents detailed execution of tasks. The following notations describe the DEM and the OM layers of the *Controller* model. In this study, we do not describe detailed operations of the OM layer, which means that we do not explain how the task can be performed. Since OMs could be designed variously in accordance with the type or resolution of the combat entity model, we just explain the role and interface of the OMs.

Notation 2. DEVS Coupled model description of Platform model:

DEM Layer - $AM_{Updater}$, AM_{Actor}

$\boldsymbol{AM_{Controller\_Updater} = < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >}$,
X = {"threat_info", "scen_info"}
Y = {"target_info"}
S = {WAIT, IDENTIFICATION}
δ_{ext} *: WAIT × "scen_info" → WAIT*
WAIT × "threat_info" → IDENTIFICATION
IDENTIFICATION × "threat_info" → IDENTIFICATION
δ_{int} *: IDENTIFICATION → WAIT*
λ : IDENTIFICATION → "target_info"
ta : WAIT → ∞
IDENTIFICATION → t_{IDNTFY}
(Response time for identification operation)

$\boldsymbol{AM_{Controller\_Actor} = < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >}$,
X = {"move_finishied", "engage_result", "scen_info", "target_info", "guidance_info"}

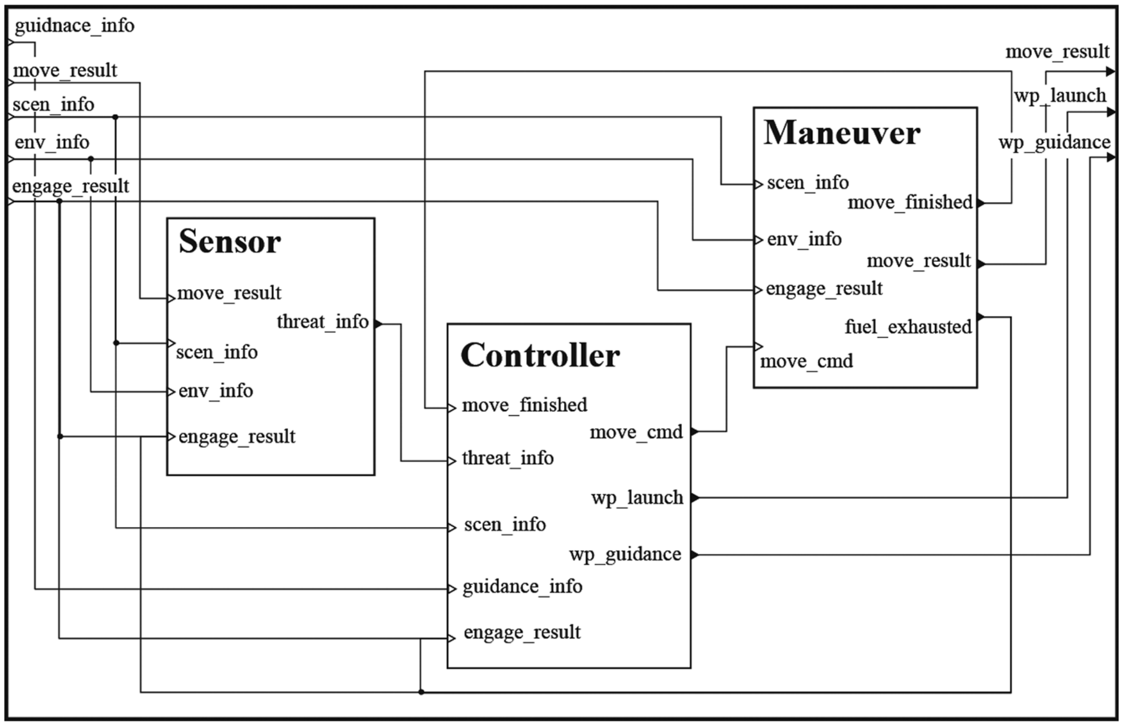


Figure 5. DEVS diagram of *Platform* coupled model.

Table 2. Engagement tasks and model representation.

Task	Task description	State representation
Reconnaissance	Decision of tactical maneuver and detection within an operating area	*RECONNAISSANCE*
Identification	Target evaluation and identification based on detected threats from the sensors	*IDENTIFICATION*
Approach	Target tracking with estimation of range, bearing, course, and velocity of the target	*APPROACH*
Combat	Combat planning such as weapon assignment	*COMBAT*
Control	Guidance and control of the launched weapon for effective utilization	*CONTROL*
Evasion	Tactical evasion after a fight	EVASION

$Y = \{\text{``move\_cmd''}, \text{``wp\_launch''}, \text{``wp\_guidance''}\}$
$S = \{IDLE, RECONNNAISSANCE, APPROACH, COMBAT, EVASION, CONTROL, END\}$
$\delta_{ext}: IDLE \times \text{``scen\_info''} \rightarrow RECONNAISSANCE$
$IDLE \times \text{``move\_finished''} \rightarrow RECONNAISSANCE$
$IDLE \times \text{``target\_info''} \rightarrow APPROACH$
$IDLE \times \text{``move\_finished''} \rightarrow IDLE \rightarrow APPROACH$
$COMBAT \times \text{``target\_info''} \rightarrow ATTACK \rightarrow COMBAT$
$EVASION \times \text{``target\_info''} \rightarrow EVASION$
$IDLE \times \text{``move\_finished''} \rightarrow EVASION$
$IDLE \times \text{``guidance\_info''} \rightarrow CONTROL$
$CONTROL \times \text{``target\_info''} \rightarrow CONTROL$
$IDLE \times \text{``target\_info''} \rightarrow IDLE$
$IDLE \times \text{``engage\_result''} \rightarrow END$
$CONTROL \times \text{``engage\_result''} \rightarrow END$
$\delta_{int}: RECONNAISSANCE \rightarrow IDLE$
$APPROACH \rightarrow IDLE$
$APPROACH \rightarrow COMBAT$
$COMBAT \rightarrow EVASION$
$EVASION \rightarrow IDLE$
$CONTROL \rightarrow IDLE$
$\lambda: RECONNAISSANCE \rightarrow \text{``move\_cmd''}$
$APPROACH \rightarrow \text{``move\_cmd''}$
$COMBAT \rightarrow \text{``wp\_launch''}$
$EVASION \rightarrow \text{``move\_cmd''}$
$CONTROL \rightarrow \text{``wp\_guidance''}$
$ta: IDLE \rightarrow \infty$
$RECONNAISSANCE \rightarrow t_{RECON}$
$APPROCH \rightarrow t_{APPRCH}$
$COMBAT \rightarrow t_{COMBAT}$
$EVASION \rightarrow t_{EVASION}$
$CONTROL \rightarrow t_{CTRL}$
$END \rightarrow \infty$

OM Layer

$OM_{Identification}$: Behavior description for target identification
Target data = Identification(Threats data)
OM_{Recon}: Behavior description for reconnaissance
Next search-pattern method = Recon(Current method)
OM_{Apprch}: Behavior description for approach to target
Approach method = Apprch(Targets/own data)
OM_{Attack}: Behavior description for combat planning
Engagement order = Attack(Targets/own data)
$OM_{Evasion}$: Behavior description for evasive action
Evasion order = Evasion(Targets/own data)
OM_{Ctrl}: Behavior description for weapon control
Control order = Ctrl(Targets/own weapon data)

Now, we shall explain the DEVS representations of the *Platform Controller* model in detail. Depending on the characteristics of tasks and the I/O properties of the *Controller* model, we classify the model into two sub-models: an *Updater* model and an *Actor* model. The *Updater* model receives threat information entering the *Controller* model, and updates and identifies whether it is a target or not. Then the *Actor* model operates proper tactical processes from target tracking to tactical evasion with identified target information. Therefore, the DEM layer has two DEMs (i.e., the *Updater* and the *Actor* models), and they are designed by the DEVS atomic models. On the other hand, OMs basically present detailed behaviors for executing above tasks.

Equally to the DEVS coupled model, we can graphically represent above-mentioned textual specification as a diagram of a DEVS atomic model. Figure 6 shows DEVS model diagrams of the *Controller* model: Figure 6(a) shows a DEVS diagram of the *Controller* coupled model, whereas Figure 6(b) and (c) illustrate diagrams of two atomic models that are the components of the *Controller* coupled model. Since the atomic model diagrams are somewhat more complicated than that of the coupled model, we provide explanatory notes shown in the middle box in Figure 6(b). The bottom box shows all OMs and their interfaces that are connected to the relevant DEMs. From now on, we only describe the model with the DEVS diagram without using DEVS textual notations.

The *Actor* atomic model conducts a task about identification and the *Updater* model accomplishes all tasks except identification. These executable tasks explained in Table 2 are represented by states in the DEVS atomic models. For instance, the *IDENTIFICATION* state in the *Updater* atomic model performs target identification, and the *APPROACH* state in the *Actor* model conducts a task for target approach.

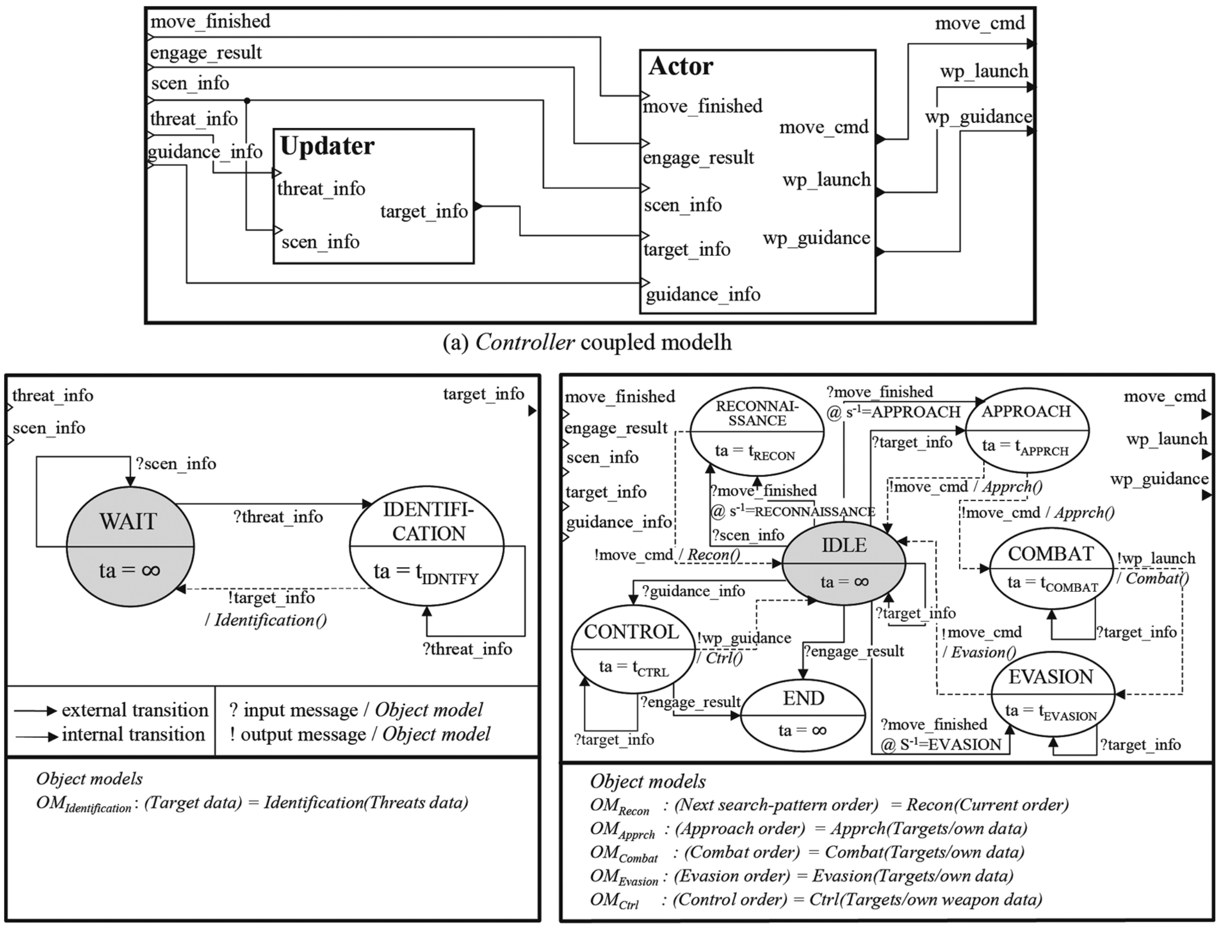
Let us explain the process of the two atomic models' state transitions more specifically. The *Updater* model in the *WAIT* state receives threatening information, *threat_info*, from the *Sensor* model and turns into the *IDENTIFICATION* state recording the information. In the *IDENTIFICATION* state, the model identifies whether the threat can be a target or not, and sends the target information, *target_info*, if necessary, and then transits the *WAIT* state. In this case, execution of the task, identification, is performed by the OM named by *Identification()*.

Next, the *Actor* model carries out behaviors in two situations: when a target is found and when no targets are discovered. At the beginning of the engagement, since there is no target, the *Actor* model in the *IDLE* state receives the initial model information, *scen_info*, and transits the *RECONNAISSANCE* state. The *RECONNAISSANCE* state in the model manifests decisions of the roving patrol type through the OM_{Recon}. With the result of OM_{Recon}, it sends the maneuver order, *move_cmd*, to the *Maneuver* model, and turns into the *IDLE* state. If the model receives the completion event, *move_finished*, from the *Maneuver* model, it repeats the above process.

When the *Actor* model receives the target information, *target_info*, at the *IDLE* state, it turns into the *APPROACH* state. And then, it performs three tasks (i.e., approach, combat, and evasion) by state transitions from the *APPROACH* to the *EVASION* state. In the *APPROACH* and *EVASION* states, the *Actor* model determines the tactical moving order and send *move_cmd* to the maneuver model, whereas, in the *ATTACK* state, the model formulates combat plans by OM_{Attack} and send *wp_launch*. If guiding launched weapons is practicable, the model in the *CONTROL* state sends the control event, *wp_guidance* to the associated *Weapon* model. When guided weapons are not available in the engagement scenario, the *Actor* model does not conduct state transitions regarding the *CONTROL* state due to the absence of an input event, *guidance_info*. Finally, the *Actor* model turns into the *END* state if it receives the input event, *engage_result*, which means that it is killed or the simulation terminates.

5.1.2 Maneuver model design. The *Maneuver* model represents the movement of the combat entity model to execute tasks physically. As noted before, the DEM layer describes abstract behaviors with state transitions and controls the overall event messages, whereas the OM layer is in charge of concrete and specific behaviors such as a maneuver algorithm. In the same way as with the *Controller* model, the *Maneuver* model consists of an *Updater* model and an *Actor* model.

Figure 7 shows DEVS diagrams of the *Maneuver* model. Initially, the *Updater* model receives the command event, *move_cmd* in the *WAIT* state and changes the state to the *INTERPRETATION* state. In the *INTERPRETATION* state, the model converts the command to physical information for dynamics through the $OM_{Cmd\_Inerpreter}$. Thereafter it sends the information, *cmd_info*, to the *Actor* model and turns into the *WAIT* state. On the other hand, the *Actor* model receives the initial information, *scen_info*, in the *IDLE* state and prepares for actions. There are largely two



(a) *Controller* coupled modelh

(b) *Updater* atomic model

(c) *Actor* atomic model

Figure 6. DEVS diagrams of *Platform Controller* model.

states of the *Actor* model for actions: the *MOVE* state and the *FUEL* state.

The *Actor* model performs the movement at the *MOVE* state with the physical information, *cmd_info*, from the *Updater* model, thus this state handles the maneuver algorithm through the $OM_{Motion\_Equation}$. For example, the DEM for action can utilize different maneuver algorithms to connect with the OM, which may be the basic equation that velocity equals velocity × time, or the advanced equation, which takes environment effects into consideration. Therefore, we adapt various kinetic algorithms or tactical operations more flexibly, minimizing the need for modification of models. During the model staying at the *MOVE* state, it can reflect the environment effect from the input event, *env_info*. After the model completes the moving command with the OM_{Cmd_Check}, it sends *move_finsished* to the *Controller* model.

In addition, the model checks the platform's endurance which is defined as the number of the days the platform remains at the engagement. The *Actor* model in the *FUEL* state computes the operating time for the platform's endurance in accordance with the OM_{Fuel_Check}, and send the output event, *fuel_exausted*, if the entire elapsed time exceeds the operating time. Similar to the *Controller* model, the *Maneuver* model turns into the *IDLE* state if it receives the input event, *engage_result*, which means that it was killed.

5.1.3 Sensor model design. Detection and homing of other combat entities is one of the main activities of combat entity modeling. The *Sensor* model is a part of the platform detecting maneuvering threats according to its own algorithm. Similar to the *Controller* and *Maneuver* models, the *Sensor* model is also classified into two groups, and all of the DEVS diagrams of the *Sensor* model are illustrated in Figure 8. To put it briefly, we design the *Sensor* model to perform the periodic scan of all the other combat entities and detect a threat solving various detection algorithms. Therefore, the *Updater* model stores the scanned information, and the *Actor* model solves the detection algorithm.

Let us give a full explanation about the model descriptions. The *Updater* model receives other platforms' physical information as well as its own one through the input

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

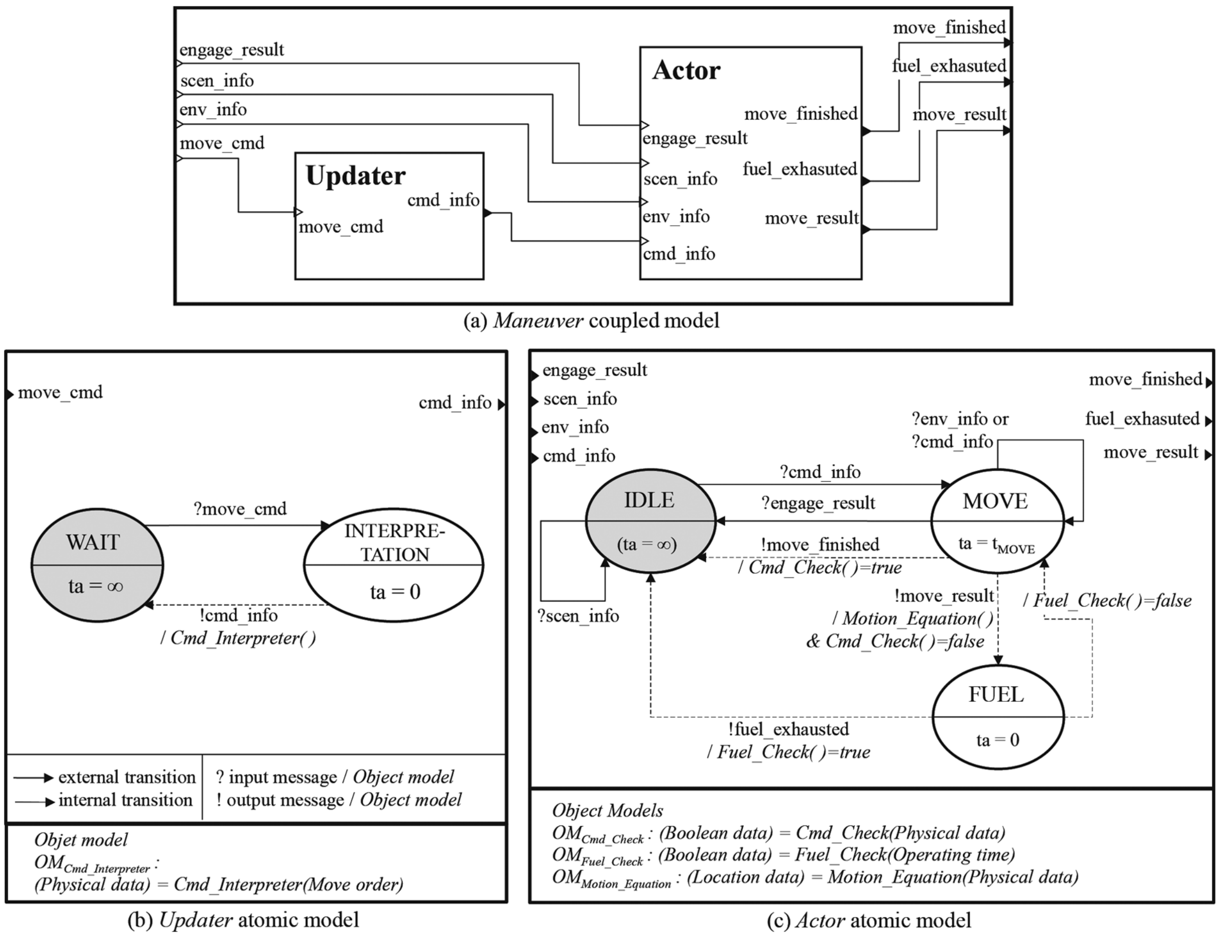

Actor
Updater
engage_result
scen_info
env_info
move_cmd
cmd_info
move_finished
fuel_exhasuted
move_result
(a) Maneuver coupled model
WAIT
ta = ∞
INTERPRE-TATION
ta = 0
?move_cmd
!cmd_info / Cmd_Interpreter( )
external transition
internal transition
? input message / Object model
! output message / Object model
Objet model
OM_Cmd_Interpreter :
(Physical data) = Cmd_Interpreter(Move order)
(b) Updater atomic model
IDLE
(ta = ∞)
MOVE
ta = t_MOVE
FUEL
ta = 0
?cmd_info
?engage_result
?env_info or ?cmd_info
?scen_info
!move_finished / Cmd_Check( )=true
!move_result / Motion_Equation( ) & Cmd_Check( )=false
/ Fuel_Check( )=false
!fuel_exhausted / Fuel_Check( )=true
Object Models
OM_Cmd_Check : (Boolean data) = Cmd_Check(Physical data)
OM_Fuel_Check : (Boolean data) = Fuel_Check(Operating time)
OM_Motion_Equation : (Location data) = Motion_Equation(Physical data)
(c) Actor atomic model

Figure 7. DEVS diagrams of *Platform Maneuver* model.

event, *move_result*, and stores them using the $OM_{Data\_Integrator}$. When the model receives the request for the stored information from the *Actor* model, it delivers it to the *Actor* model.

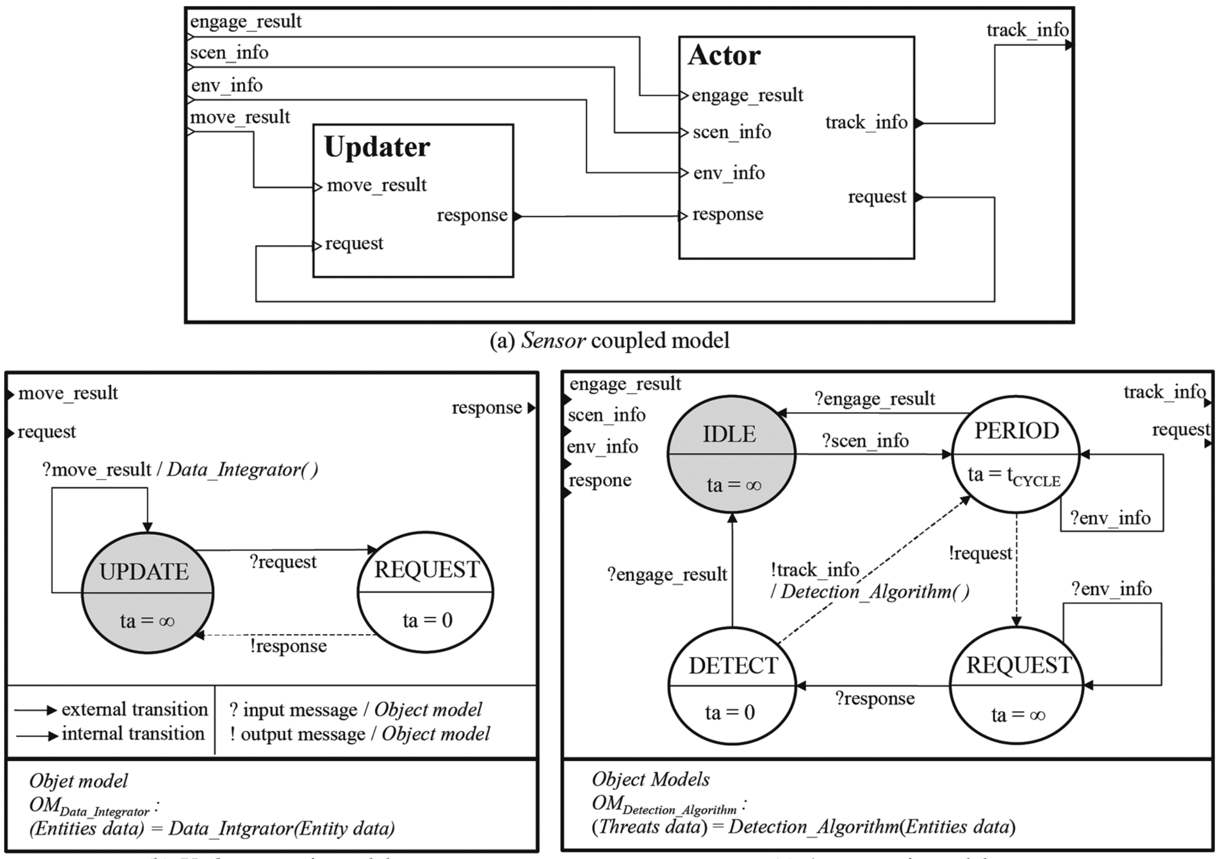
In the *Actor* model, it receives the initial information, *scen_info*, and changes the current *IDLE* state to the *PERIOD* state. In the *PERIOD* state, the model holds on the periodic time for scanning while updating the input events. After the periodic time, t_{CYCLE}, the model sends the output event, *request*, for requesting the scanned data and turns into the *REQUEST* state. The model in the *REQUEST* state waits for the response from the *Updater* model. When the input event, *response*, enters the model, it changes the *REQUEST* state to the *DETECT* state. In the *DETECT* state, the *Actor* model conducts the detection algorithm designed in the $OM_{Detection\_Algorithm}$. Since the combat entity has several detection systems for detecting various frequency sounds, the *Sensor* model can contain multiple OMs for sensing.

Until now, we have explained the three component models of the *Platform* model – the *Controller*, the *Maneuver*, and the *Sensor* models – for moving, sensing, and deciding activities, respectively. Just like the *Platform* model, the *Weapon* model, also, conducts the same activities. There are, however, some differences between the two types of combat entity model, which come from different tactical behaviors. Fundamentally, the *Weapon* model undertakes simple engagement tasks, e.g., search, identification, and approach, which are regarded as a basic type of the *Weapon* model. In addition, as information technologies develop, it enables that launched weapons are being controlled, which is an optional task for the *Weapon* model. This optional task depends on whether the *Weapon* model is a basic type for fire-and-forget or an advanced type for being guided. Therefore, fundamental tasks are comprised in a basic type of DEVS model, and the existing DEVS semantics can be reused for an advanced type, which is the additional advantage of the DEVS formalism. The detailed DEVS specification of the *Weapon* model can be seen in Appendices A and B.

5.2. Model relation between controller and maneuver models

From the previous sub-section, we have examined the detailed DEVS-based design of the combat entity model.

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014



(b) *Updater* atomic model (c) *Actor* atomic model

Figure 8. DEVS diagrams of Platform *Sensor* model.

During model execution, or simulation, the components of the model fulfill their roles, which have connections with other components. In this section, we explain these connections partially to understand the relationships of the component models more easily.

Figure 9 shows simplified DEVS diagrams of a combat entity, which is regardless of a platform or a weapon. In Figure 9, we use bold lines and characters to highlight event sequences that we intend to explain. Let us begin our viewpoint at the occurrence of the event, *target_info*, in the $DEM_{Controller\_Updater}$. The $DEM_{Controller\_Actor}$ receives the input event, *target_info*, and changes the current state to the *APPROACH* state. After the $t_{APPRACH}$, which is the time advance value of the *APPROACH* state, elapses, the $DEM_{Controller\_Actor}$ calls the OM_{Apprch} to obtain a maneuver pattern for target approach. In the case of a submarine, combinations of three kinds of approach pattern, i.e., point, lead, and lag,[46] are formulated in the OM_{Apprch}. In this example, the OM_{Apprch} decides *Pattern 1* based on the target and own data, and returns the command to the $DEM_{Controller\_Actor}$. With this returned message, the $DEM_{Controller\_Actor}$ sends the output event for moving order, *move_cmd*, to the $DEM_{Maneuver\_Updator}$.

The $DEM_{Maneuver\_Updator}$ turns the current *WAIT* state into the *INTERPRETATON* state upon receiving *move_cmd*. In the *INTERPRETATON* state, it summons the $OM_{Cmd\_Interpreter}$ to convert the command to the physical data such as angles, velocity, or distance. Then the $DEM_{Maneuver\_Updater}$ sends these data to the $DEM_{Maneuver\_Actor}$. After the $DEM_{Manuever\_Actor}$ receives the input event, *cmd_info*, it changes to the *MOVE* state. Finally, in the *MOVE* state, the model solves the actual maneuver equation through the $OM_{Motion\_Equation}$ and transmits the result to the external model. In the case of the *MOVE* state, the time step size for solving the maneuver equation is represented by the time advance value of the *MOVE* state, t_{MOVE}, and this value can vary according to the command type. For instance, in the *RECONNAISSANCE* state, the time step is a large value because there is not actual engagement. Otherwise, if an engagement situation occurs and the target is identified, we can utilize the very small time steps for the *APPROACH* and *ATTACK* states for more accurate simulation. This is the additional advantage of the proposed modeling design.

Now, we leave two atomic models for the *Simulation model*. These models are simpler than the other models

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

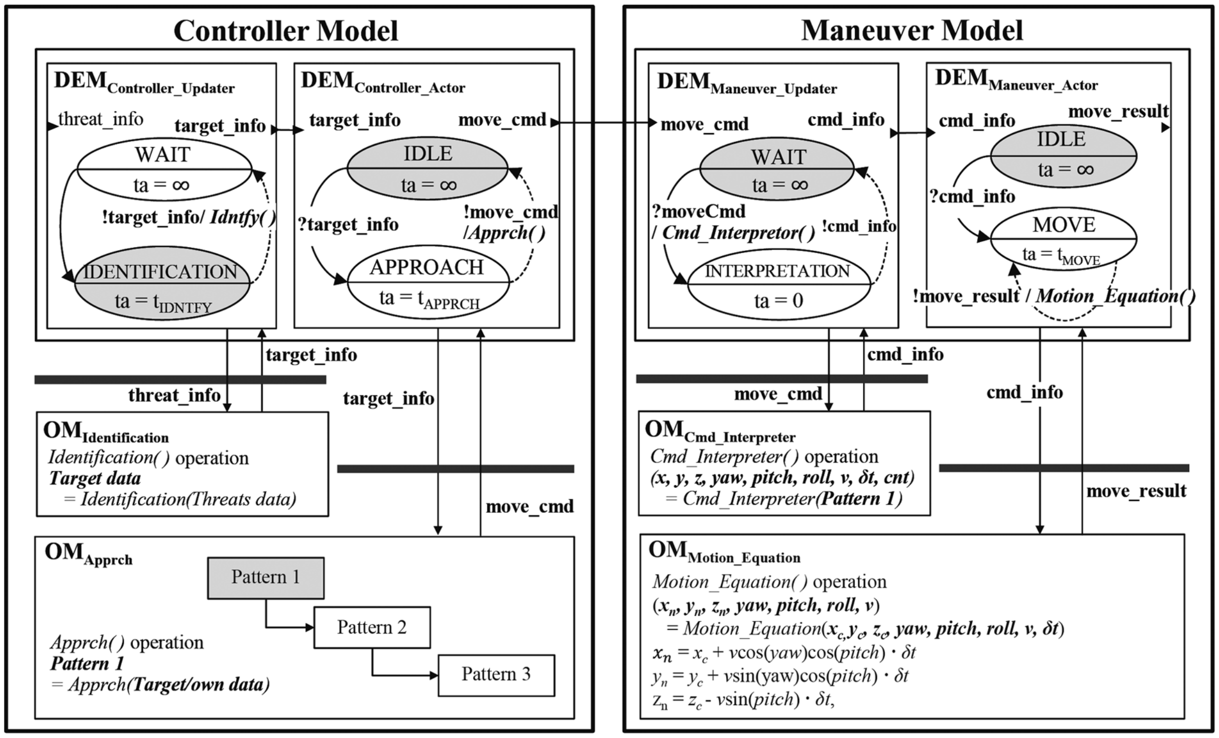


Figure 9. Relation between *Controller* and *Maneuver* models.

described. Therefore, we describe these two models briefly and move on to model implementation.

5.3. Damage assessment & environment model design

The *Damage assessment* model investigates the effect that weapons damage targets. This model receives physical information of all the combat entities and processes the mechanisms by which weapons can inflict damage at an interval of a periodic cycle, t_{PERIOD}. Thereafter the engagement result is transferred to the combat entity models. Figure 10 shows these processes with a DEVS diagram of the *Damage assessment* atomic model.

During engagement, the enemy detection probability and one's own movement are dependent on environmental effects such as terrain features and weather patterns. Thus, these environmental effects affect the combat entities' physical components such as the *Maneuver* and the *Sensor* models. Figure 11 illustrates how the *Environment* model is modeled by the DEVS formalism. The *Environment* model sends time varying information about environment effects to all platform and weapon models.

So far, we examined modeling of the *Combat system model* from a macroscopic perspective as well as a microscopic view, which are focused on DEVS-based modeling method. The way we model combat entities, such as their interactions based on core activities and events and state transitions for engagement tasks, plays a pivotal role for *engagement*-level combat M&S. The purpose of this modeling is to execute the models eventually for the simulation, which represents how to effectively communicate information between the inside and outside of combat entities. In the following section, we introduce implementation of the designed model for simulation.

6. Model implementation

After the DEVS modeling, we need to implement the DEVS model for simulation. This section introduces several implementation frameworks for DEVS-based models and shows how we implement our DEVS models.

6.1. Technical implementation of DEVS-based combat model

Implementing models that are specified with the DEVS formalism is easily achievable by utilizing an implementation framework supporting the formalism. For example, the DEVS models are implementable using DEVSim++,[37] DEVSJava,[38] CD++,[39] or SiMA,[47] etc. As far as a model is implemented by following the template of the DEVSim++ library, the implemented model is executable by a general simulation engine provided by DEVSim++.

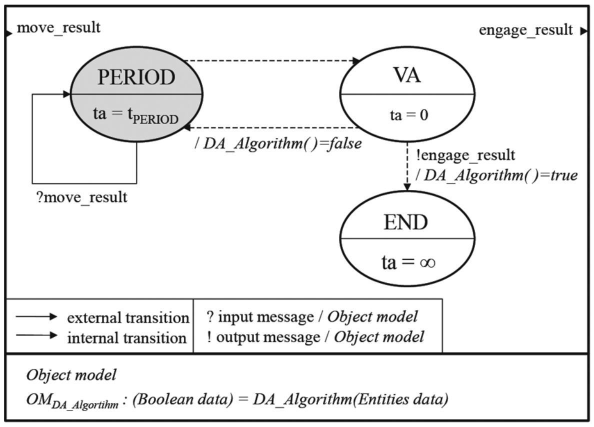


Figure 10. DEVS diagram of *Damage assessment* atomic model.

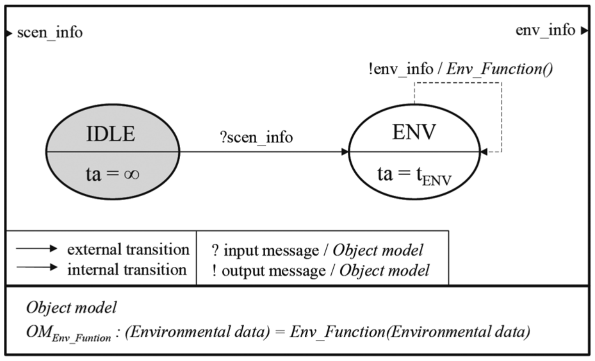

scen_info
env_info
!env_info / Env_Function()
IDLE
ta = ∞
?scen_info
ENV
ta = t_ENV
external transition
internal transition
? input message / Object model
! output message / Object model
Object model
OM_Env_Funtion : (Environmental data) = Env_Function(Environmental data)

Figure 11. DEVS diagram of *Environment* atomic model.

This means that a modeler is freed from implementing the same model repeatedly for multiple simulations.

The DEVSim++, which has been widely used to implement various areas of DEVS models, is the implementation framework that realizes the DEVS formalism in the C++ language.[48] In this study, the *Combat system model* is implemented by using the DEVSim++ for model simulation. All the DEMs of the *Combat system model* are implemented using the DEVSim++, and the detailed algorithms and equations of the OMs are realized using the C++ language. Commonly, the DEM layer could contain more than one DEM according to the need, and each individual DEM would also link up with one or several OMs in the OM layer by sharing interfaces between the two layers. This is a natural situation for flexible simulation to change detailed behaviors in OMs.

To support linking between two layers effectively, we use a shared library technique such as the dynamically linked library (DLL), which enables the modeler to switch the algorithms or dynamic equations in OMs during simulation without recompiling.[16] That is, the OM provides various algorithm candidates, which is implemented with the DLL; and linking the DEM to the corresponding OM is based on a communication interface called *function prototype* in the C++ language.

Figure 12 shows an implementation example of a DEM and an OM for the *Controller Actor* model. Whenever the DEM is in the *APPROACH* state, it calls the *Apprch()* to the OM. The *Apprch()* can be changeable as the modeler applies alternative algorithms. In Figure 12, each alternative function is developed as a separate DLL, and we express it in different patterned blocks. The DEM can pick a proper OM in the DLL pool if it knows inputs, outputs, and the name of the function prototype. Inversely, the OM is available for other DEMs if they are implemented using the same function prototype and the same I/O.[16] This enhances reusability in terms of the various behaviors of the developed models.

7. Case study

In this section, we introduce a case study about a specific engagement scenario. The scenario is anti-submarine warfare (ASW), which is one-to-one engagement—that is, a friendly warship versus a hostile submarine.[16] The goals of the case study are twofold: (1) to determine how factors are improved when we use the proposed modeling techniques; and (2) to determine what results of the simulation analysis can give better information to the M&S users, such as decision makers.

7.1. Engagement scenario

In the ASW, the warship is attacked by an anti-surface torpedo fired by the submarine. The brief scenario illustrated in Figure 13 is as follows (italic words in parenthesis and Figure 13 mean engagement tasks that the warship should perform):

1. A friendly warship makes a reconnaissance within an operating area. (*Reconnaissance*)
2. A hostile submarine launches a torpedo to the friendly warship after detection.
3. The launched torpedo explores some targets with its own searching rule.
4. The warship identifies an approaching threat, the torpedo. (*Identification*)
5. The warship decides combat planning with multiple decoy systems. (*Combat*)
6. After operating decoy systems, the warship makes a detour to be far from the torpedo. (*Evasion*)
7. The torpedo is deceived by the decoys and repeats re-search.

In Figure 13, the submarine fires a single torpedo against the warship and the warship operates four decoys with a certain tactical pattern. Thus, this engagement

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

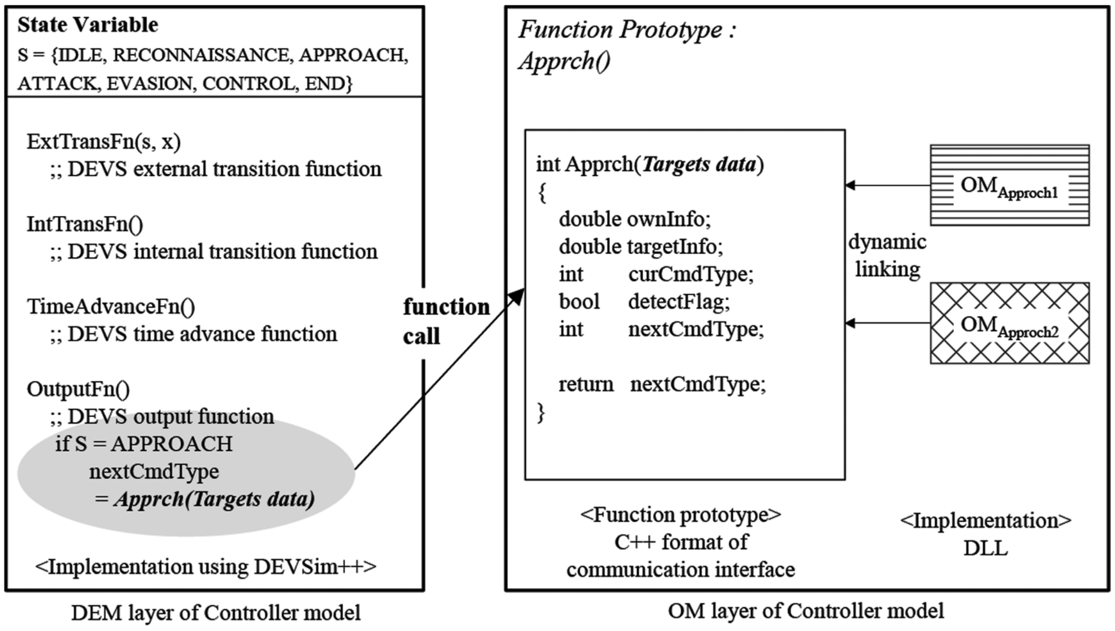


Figure 12. Implementation example of *Controller* model (taken from Sung and Kim).[36]

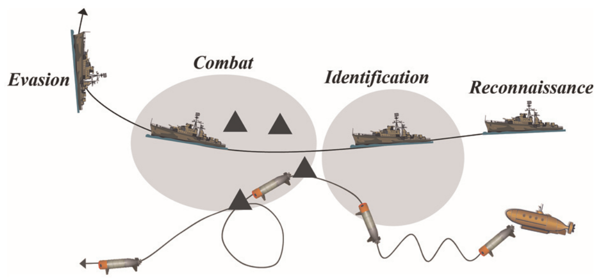


Figure 13. Brief engagement scenario of anti-submarine warfare (ASW) (taken from Seo et al.).[16]

scenario requires seven combat entity models: two *Platform* models and five *Weapon* models.

7.2. M&S development and experimental design

Before the model implementation, we first set up the survival rate of the friendly warship as a measure of effectiveness (MOE). Key variables for the initial param are provided in our previous work.[16] Thereafter, we implemented an *Experimental frame* and the *Simulation model* using DEVSim++. The brief model structure of the *Simulation model* is depicted in Figure 14. We implemented the two *Platform* models with the same DEM structures, though they are distinguished by different OMs and initial param. This is the same with the case of five *Weapon* models. This compositional reusability is to achieve the first goal of the case study, which is described in the following sub-section.

Table 3 illustrates an experimental design with four experiments to accomplish the second goal, i.e., effectiveness analysis. The first two experiments in Table 3 are easily accomplished to change just the model parameters. On the other hand, the two backward experiments are not related to the model parameters but the model behaviors, which mean that modification of the designed model is inevitable. In these cases, the proposed modeling technique demonstrates its advantage in minimizing model revision due to the well-classified model design. For example, for the third case of Figure 13, we only redesign the OM_{Combat} of the *Controller* model of the warship rather than the whole part, preserving the interfaces between the two layers. Similarly, we only implement the alternative $OM_{Tactical\_Search}$ of the *Controller* model of the torpedo for the fourth case.

7.3. Experimental results

Now we describe our experimental results in two ways. The first way is to show compositional reusability in the model development process, which is an advantage of the proposed modeling. Second, we analyze simulation results, varying the four experimental cases in Table 3.

7.3.1. Compositional reusability in model development. Biggerstaff and Richter categorize reusability techniques into two types[49]: generational reusability and

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

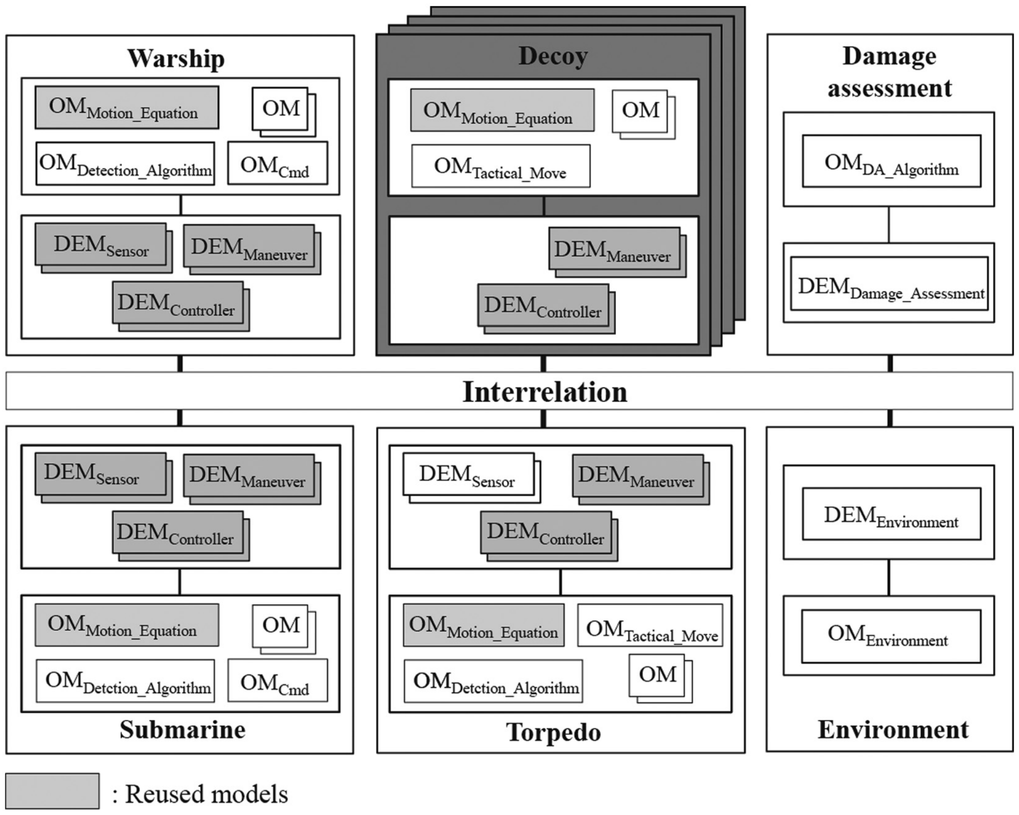


Figure 14. Brief model structure of *Simulation model* and reusability representation.

compositional reusability. The generational reusability is the pattern of generating software; the compositional reusability is composing multiple entities to create a larger model. Since simulation model reuse is more composition technology than generation technology,[50] we focus on compositional reusability, or hierarchical reusability, that occurs during model implementation in this study.

Figure 14 shows the brief model structure of the *Simulation model* for the case study. Shaded boxes in Figure 14 represent models to be reused just once. This means that multiple reuse can occur at any time. All the DEMs of combat entity models except the torpedo's DEM_{Sensor} are reused, and furthermore, decoy models are reused entirely. On the other hand, OMs are not frequently reused compared to DEMs, this is caused by the fact that a combat entity model is characterized as its distinct OMs.

Even though we just provide schematic information for compositional reusability instead of quantitative data, we assure readers to understand that the proposed model design has a well-defined structure divided into common and characterized parts: common parts mean DEMs and characterized parts correspond to OMs. Therefore, this well-structured model design guarantees multi-level reusability, which will be explained in the next discussion section.

7.3.2. Effectiveness analysis: regression analysis of experimental results. For effectiveness analysis, this sub-section statistically analyzes which factors contribute to the MOE, i.e., the survival rate of the friendly warship, as well as how strongly and robustly they are contributing. To do this, we built a linear regression model for the experimental results with standardized coefficients and p-values for independent experiments (see Table 4).

In Table 4, we identify three major findings: The first is the regression coefficient of each experiment. The experiment with a higher coefficient value has a greater influence on the MOE than the experiment with a lower value. For instance, the detection range of the warship is the most significant among the four, and it implies that the most important point is detecting the enemy as quickly as possible. The second finding is the R^2 of the experiment. The R^2 has a value between 0 and 1, and it measures how well the resulting line in the regression model matches the original data points. Since the R^2 value is 0.831 in these experiments, we can predict the output sufficiently with this regression model. The third finding is the p-value of each experiment. With the p-value, we determine which input variables are directly related to the output variable. As we see in Table 4, all experiments are relevant to significant input variables for the output variable.

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

Table 3. Four experimental designs for case study (extended from Seo et al.).[16]

Experimental design	Variation cases	Implications
Detection range of warship (Experiment 1)	2000, 2500, 3000, 3500, 4000 m (5 cases) **default :** 3000 m	This experiment is achieved by varying an initial parameter. The parameter is used in the *Sensor* model of the warship.
Velocity of mobile decoy (Experiment 2)	3, 6, 9, 12, 15 knots (5 cases) **default :** 12 knots	This experiment is achieved by varying an initial parameter. The parameter is used in the *Maneuver* model of the decoy
Operating pattern of decoys (Experiment 3)	Pattern 1, 2, 3, 4 (4 cases) **default :** Pattern 3	This experiment is achieved by varying the OM_{Combat} that is used in the *Controller* model of the warship. Pattern 1 uses only static decoys; pattern 2 to 4 mix static and mobile decoys. Pattern 1: four static decoys are used. Pattern 2: four mobile decoys are used. Pattern 3: two static decoys at the front of warship and two mobile decoys at the rear are used. Pattern 4: two mobile decoys at the front of warship and two static decoys at the rear are used.
Search pattern of torpedo (Experiment 4)	Type 1, 2, 3, 4 (4 cases) **default :** Type 4	This experiment is achieved by varying the $OM_{Tactical\_Search}$ that is used in the *Controller* model of the torpedo Type 1 means a straight-running torpedo; Types 2 to 4 mean pattern-running torpedoes. Type 1: only the straight moving segment is used, Type 2: straight and winding moving segments are used. Type 3: winding and circular moving segments are used. Type 4: all three moving segments are used.
Total	400 ($5^2 \times 4^2$) cases	Number of replications per a case : 100 times

7.3.3. Effectiveness analysis: trend of specific experimental cases. Now, we explain a changing trend of the MOE. Since total experiments contain 400 cases, we cannot interpret all the cases in this study. Therefore, we chose certain cases, especially the first and the second experiments, and analyzed them.

Figure 15 shows simulation results according to the first and the forth experiments in Table 3. The x-axis represents the detection range of the warship which is the most influencing factor among four experiments, and the y-axis shows the probability of the warship's survival, i.e., the MOE. Four lines in the graph are for one straight-running and three pattern-running torpedo models that are relevant to the forth experiment.

For modeling of pattern-running torpedoes, we used three different moving types composed of two or more moving segments. Three kinds of segments, i.e., straight, winding, and circular segments, were used in our experiment, which differently influence on the torpedo's observation angle and velocity. For example, the straight segment is the fastest among the three, but it has the narrowest angle of sensing in the forward direction. On the other hand, the circular segment can sense in all directions with the lowest velocity. Lastly, the winding segment is a compromise between the straight and the circular segments. Three moving types integrating with these segments differently were developed in the $OM_{Tactical\_Search}$ of the torpedo model separately, and it can interconnect with the $DEM_{Controller\_Actor}$ through the same interface. Therefore, we developed alternative $OM_{Tactical\_Search}$ to evaluate various straight-running and pattern-running torpedoes.

Table 4. Regression coefficient and p-value from the regression analysis by four experiments (* for p-value < 0.05).

Experiment	Regression coefficient	p-value
Experiment 1	0.847	1.32×10^{-57} (*)
Experiment 2	0.091	6.0×10^{-3} (*)
Experiment 3	0.199	2.59×10^{-6} (*)
Experiment 4	0.182	2.13×10^{-6} (*)
R^2	0.831	

In Figure 15, we fist can see that the warship should have the ability of at least 3000 m for detection to achieve more than 90 percent of survivability. That is, if the warship detects some threat out of 3000 m, it could secure enough time to employ decoy systems and make a detour. This is a typical example to determine a required operational capability (ROC) of the warship. Next, from the side of the torpedo, we can find that Type 4 is the most threatening moving type among four types because it results in the lowest survivability of the warship. On the contrary,

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

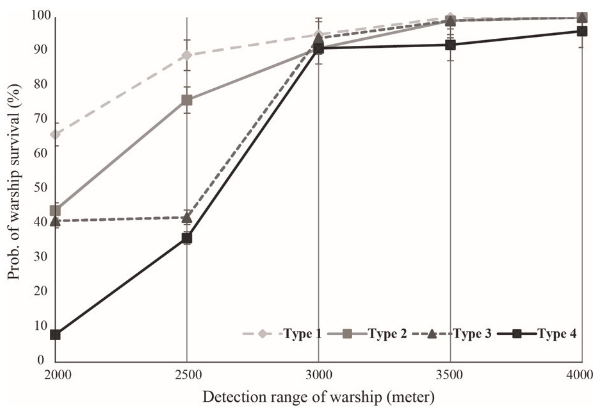


Figure 15. Changes of warship survival rate by varying Experiments 1 and 4.

Type 1 (i.e., a straight-running torpedo) has little effect, which is the principle reason that straight-running torpedoes are not operating any more in military. In comparison with Types 2 and 3, when the detection range of the warship is less than 3000 m, Type 3 is more effective than Type 2. This may come from that the circular segment is more effective when a target is far away from all directions rather than nearby at the forward direction.

Note that any moving types for a pattern-running torpedo cannot be effective if the warship has sufficient detection ability and efficient countermeasure systems. In this case, the only way for the torpedo to attack a target successfully is to approach to the target as quickly as possible without any being discovered.

7.4. Discussion

One of the main advantages of DEVS-based modeling is the increased reusability of algorithms, models, and engines (i.e. DEVS implementations).[7] The proposed combat modeling is discussed under the DEVS formalism to provide a basis and rationale for compositional reusability, which is the goal of the first experimental results. For instance, one $OM_{Motion\_Equation}$ in Figure 14 is reusable in every combat entity for similar moving behaviors. This illustrates the algorithm-level reusability. Also, DEMs of a combat entity model or a combat entity model itself can be reusable, which is an indication of the model-level reusability. Finally, the designed DEVS models are reusable for any DEVS implementation, which is relevant to engine-level reusability. Illustrations for the algorithm- and model-level reusability in Figure 14 show the guidelines for a modeler to improve reusability of DEVS models.

Next, we give a better interpretation for a specific ASW through statistical analysis. In other words, we provide several results of effectiveness analysis to gain insights into the ROC of platforms or tactical operations of weapons. For example, this experiment shows that how the effective tactical operation of the weapon affects the MOE considerably with given capabilities. In the Republic of Korea navy, mobile decoys and pattern-running torpedoes, introduced in this case study, have developed as a product or are being evaluated tactically. We ensure that the proposed modeling framework provides a guideline regarding whether to develop combat platforms and weapons or to assess innovative tactical operations.

8. Conclusion

This study has described a fine-grained DEVS modeling approach for an engagement-level combat system, especially a combat entity. For effective modeling of a combat entity, we classify it into platform and weapon models and broke the combat entity model into three functional submodels with two abstraction levels. It provides a model structure that improves the multi-level compositional reusability of the combat entity model. Also, for straightforward understanding of an engagement scenario, we considered the scenario as a flow of executable tasks. The task flow is expressed by the formal semantics, which provides intuitive appeal, reducing the effort required to read and understand the model and reflecting the real-world scenario effectively.

We realized the above-mentioned points through the use of the DEVS formalism which delineates model coupling schemes and behaviors through a modular and hierarchical design. With the use of the proposed modeling techniques, we can conduct constructive simulations, perform various engagement scenarios, and assess the efficiency of weapons, minimizing additional modeling efforts.

M&S-based development of a combat entity, such as a warship or a decoy, commonly begins at small-scale engagement. Subsequently, we limited the abstraction level of defense M&S to be applied to *engagement*-level models in this study. If the proposed model is utilized for large scale engagement, such as a combined operation, we need to expand the proposed modeling technique to cover a broad spectrum of combat situations. In this situation, we will achieve interoperation with other simulation models through high-level architecture (HLA) or test- and training-enabling architecture (TENA), and this development process will be addressed in one of our future works.

Under the present conditions, the main beneficiaries of this study will be the military strategists of the Korean forces. The Korean Agency for Defense Agency (ADD) has made full use of the proposed simulation model. Furthermore, we expect that this work will provide guidance for decisions about purchasing equipment, such as next-generation weapons and platforms, or developing innovative tactical operations.

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

Funding

This work was partially supported by the Agency for Defense Development under contract (UD090024DD) and the Defense Acquisition Program Administration and the Agency for Defense Development under contract (UD110006MD).

References

1. Upadhya KS and Srinivasan NK. A simulation model for availability under battlefield situations. *Simulation* 2000; 74: 332–339.
2. Smith RD. Essential techniques for military modeling and simulation. In: *Proceedings of the 1998 winter simulation conference*, Washington, DC, December 13–16, 1998, pp.805–812.
3. Piplani LK, Mercer JG and Roop RO. *Systems acquisition manager's guide for the use of models and simulations*. Report of the DSMC 1993–1994. Fort Belvoir, VA: Defense Systems Management College Press, 1994.
4. Ting SP and Zhou S. Dealing with dynamic changes in time critical decision-making for MOUT simulations. *Comput Anim Virtual Worlds* 2009; 20: 427–436.
5. Liang KH and Wang KM. 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*, Monterey, CA, December 3–6, 2006, pp.1299–1306.
6. Karasakal O. Air defense missile-target allocation models for a naval task group. *Comput Oper Res* 2008; 35: 1759–1770.
7. Tolk A. *Engineering principles of combat modeling and distributed simulation*. New Jersey: Wiley, 2012.
8. Wing JM. A specifier's introduction to formal methods. *Computer* 1990; 23(9): 8–24.
9. France R, Evans A, Lano K, et al. The UML as a formal modeling notation. *Comput Stand Interfaces* 1998; 19: 325–334.
10. Bohnenkamp H, D'Argenio PR, Hermanns H, et al. MODEST: A compositional modeling formalism for hard and softly timed systems. *IEEE Trans Software Eng* 2006; 32: 812–830.
11. Zeigler BP. *Multi-facetted modeling and discrete event simulation*. Academic Press, 1984.
12. Hill RH, Miller JO and McIntyre GA. Applications of discrete event simulation modeling to military problems. In: *Proceedings of the 2001 winter simulation conference*, Arlington, VA, December 9–12, 2001, pp.780–788.
13. Sung CH, Hong JH and Kim TG. Interoperation of DEVS models and differential equation models using HLA/RTI: hybrid simulation of engineering and engagement level models. In: *Proceedings of the 2009 spring simulation multi-conference*, San Diego, CA, March 22–27, 2009, pp.387–392.
14. Hong JH, Seo KM, Seok MG, et al. Interoperation between engagement-and engineering-level models for effectiveness analyses. *J Defense Model Simul* 2011; 8(3): 143–155.
15. Seo KM, Hong JH and Kim TG. DEVS-based underwater warfare simulation development for effectiveness analysis. In: *Proceedings of the 2010 summer simulation multi-conference*, Ottawa, ON, Canada, July 11–15, 2010.
16. Seo KM, Song HS, Kwon SJ, et al. Measurement of effectiveness for an anti-torpedo combat system using a discrete event systems specification-based underwater warfare simulator. *J Defense Model Simul* 2011: 8(3): 157–171.
17. Andrien K, Caussanel J and Giambiasi N. DEVS model for CGF scenario. In: *Proceedings of the 17th IMACS world congress*, Paris, France, July 11–15, 2005.
18. Moreno A, Torre L, Risco-Martin JL, et al. DEVS-based validation of UAV path planning in hostile environments. In: *Proceedings of the international defense and homeland security simulation workshop*, Vienna, Austria, September 19–21, 2012, pp.135–140.
19. Cho DY, Son MJ, Kang JH, et al. Analysis of a submarine's evasive capability against an antisubmarine warfare torpedo using DEVS modeling and simulation. In: *Proceedings of the 2007 spring simulation multi-conference*, San Diego, CA, March 25–29, 2007, pp.307–315.
20. Park SC, Kwon Y, Seong K, et al. Simulation framework for small scale engagement. *Comput Ind Eng* 2010; 59: 463–472.
21. Kim TG. Lecture note for discrete event system modeling and simulation (EE612), http://smslab.kaist.ac.kr/Course/EE612/2013/ (accessed November 2013).
22. Ho YC. Introduction to special issue on dynamics of discrete event systems. *Proc IEEE* 1989; 77(1): 3–6.
23. Page EH and Smith R. Introduction to military training simulation: a guide for discrete event simulationists. In: *Proceeding of the 1988 winter simulation conference*, Washington, DC, December 13–16, 1998, pp.53–60.
24. Zeigler BP, Praehofer H and Kim TG. *Theory of modeling and simulation*. 2nd ed. Academic Press, 2001.
25. Zeigler BP and Vahie S. DEVS formalism and methodology: unity of conception/diversity of application. In: *Proceedings of the 1993 winter simulation conference*, Los Angeles, CA, December 12–15, 1993, pp.573–579.
26. Zeigler BP, Fulton D, Nutaro J, et al. M&S enabled testing of distributed systems: Beyond interoperability to combat effectiveness assessment. In: *The 9th annual modeling and simulation workshop*, December 8–11, 2003.
27. Hong GP and Kim TG. A framework for verifying discrete event models within a DEVS-based system development methodology. *Trans Soc Comput Simul* 1996; 13(1): 19–34.
28. Palaniappan S, Sawhney A and Sarjoughian H. Application of the DEVS framework in construction simulation. In: *Proceedings of the 2006 winter simulation conference*, Monterey, CA, December 3–6, 2006, pp.2077–2086.
29. Kwon SJ, Seo KM, Kim BS, et al. Effectiveness analysis of anti-torpedo warfare simulation for evaluating mix strategies of decoys and jammers. *Advanced methods, techniques, and applications in modeling and simulation*. Springer, 2012.
30. Seo KM, Sung CH and Kim TG. Realization of the DEVS formalism in matlab/simulink. In: *Proceedings of the grand challenges in modeling and simulation*, Edinburgh, Scotland, June 16–19, 2008, pp.251–256.
31. Risco-Martın JL, Mittal S, Zeigler BP, et al. From UML state charts to DEVS state machines using XML. In: *Proceedings of the workshop on multi-paradigm modeling: concepts and tools*, Nashville, TN, September 30–October 5, 2007.
32. Borland S and Vangheluwe H. Transforming statecharts to DEVS. In: *Proceedings of the summer computer simulation conference*, Montreal, Canada, July 20–24, 2003, pp.154–159.

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

33. Barros FJ. The dynamic structure discrete event system specification formalism. *Trans Soc Comput Simul Int* 1996; 13(1): 35–46.
34. Chow ACH. Parallel DEVS: a parallel, hierarchical, modular modeling formalism and its distributed simulator. *Trans Soc Comput Simul Int* 1996; 13(2): 55–68.
35. Hong JS, Song HS, Kim TG, et al. A real-time discrete event system specification formalism for seamless real-time software development. *Discrete Event Dyn Syst* 1997; 7: 355–375.
36. Sung CH and Kim TG. Collaborative modeling process for development of domain-specific discrete event simulation systems. *IEEE Trans Syst Man Cybern Part C Appl Rev* 2012; 42: 532–546.
37. Kim TG, Sung CH, Hong SY, et al. DEVSim++ toolset for defense modeling and simulation and interoperation. *J Defense Model Simul* 2011; 8(3): 129–142.
38. Arinoza Center for Integrative Modeling and Simulation. DEVSJAVA 2.7, http://acims.asu.edu/software/devsjava (accessed November 2013).
39. Wainer G. CD++: a toolkit to develop DEVS models. *Software Pract Experience* 2002; 32: 1261–1306.
40. Robinson T. ODIN – an underwater warfare simulation environment. In: *Proceedings of the 2001 winter simulation conference*, Arlington, VA, December 9–12, 2001, pp.672–679.
41. Canney S, Best J and Cramp A. *Virtual maritime system architecture description document issue 2.0*. Virtual Maritime System Document, 2002.
42. Kercbner RM and Hughes RG. TAC BRAWLER: an application of engagement simulation modeling to simulator visual system display for air combat maneuvering. In: *Proceedings of the second symposium*, April 25–28, 1983, pp.599–606.
43. Fong G. Adapting COTS games for military simulation. In: *Proceedings of the 2004 ACM SIGGRAPH international conference on virtual reality continuum and its applications in industry*, Singapore, June 15–18, 2004, pp.269–272.
44. Sung CH, Hong SY and Kim TG. Layered approach to development of OO war game models using DEVS framework. In: *Proceedings of the summer computer simulation conference*, Philadelphia, PA, July 24–28, 2005, pp.65–70.
45. Song HS and Kim TG. DEVS diagram revised: a structured approach for DEVS modeling. In: *Proceedings of European simulation conference*, Hasselt, Belgium, October 25–27, 2010, pp.94–101.
46. Bakos G. *Submarine approach and attack tactics – simulation and analysis*. Masters Dissertation, Naval Postgraduate School, 1995.
47. Deniz F, Alpdemir M, Kara A, et al. Supporting dynamic simulations with simulation modeling architecture (SiMA): a discrete event system specification-based modeling and simulation framework. *Simulation: Trans Soc Comput Simul Int* 2012; 88: 707–730.
48. Kim TG and Park SB. The DEVS formalism: hierarchical modular systems specification in S++. In: *Proceedings of the 1992 European simulation multi-conference*, York, UK, June 1–3, 1992, pp.152–156.
49. Biggerstaff TJ and Richter C. Reusability framework, assessment, and directions. In: *Software reusability*. New York: ACM Press, 1989.
50. Choi YI and Kim TG. Reusability measure of DEVS simulation models in DEVSim++ environment. In: *Proceedings of the AeroSense 97 conference on photonic quantum computing*, Orlando, FL, April 20–25, 1997, pp.244–255.

Author biographies

Kyung-Min Seo received his PhD in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2014. Currently, he is a research engineer at Daewoo Shipbuilding & Marine Engineering (DSME) Co., Ltd. His research interests include combat modeling and simulation, discrete event system, system of systems, and effectiveness analysis.

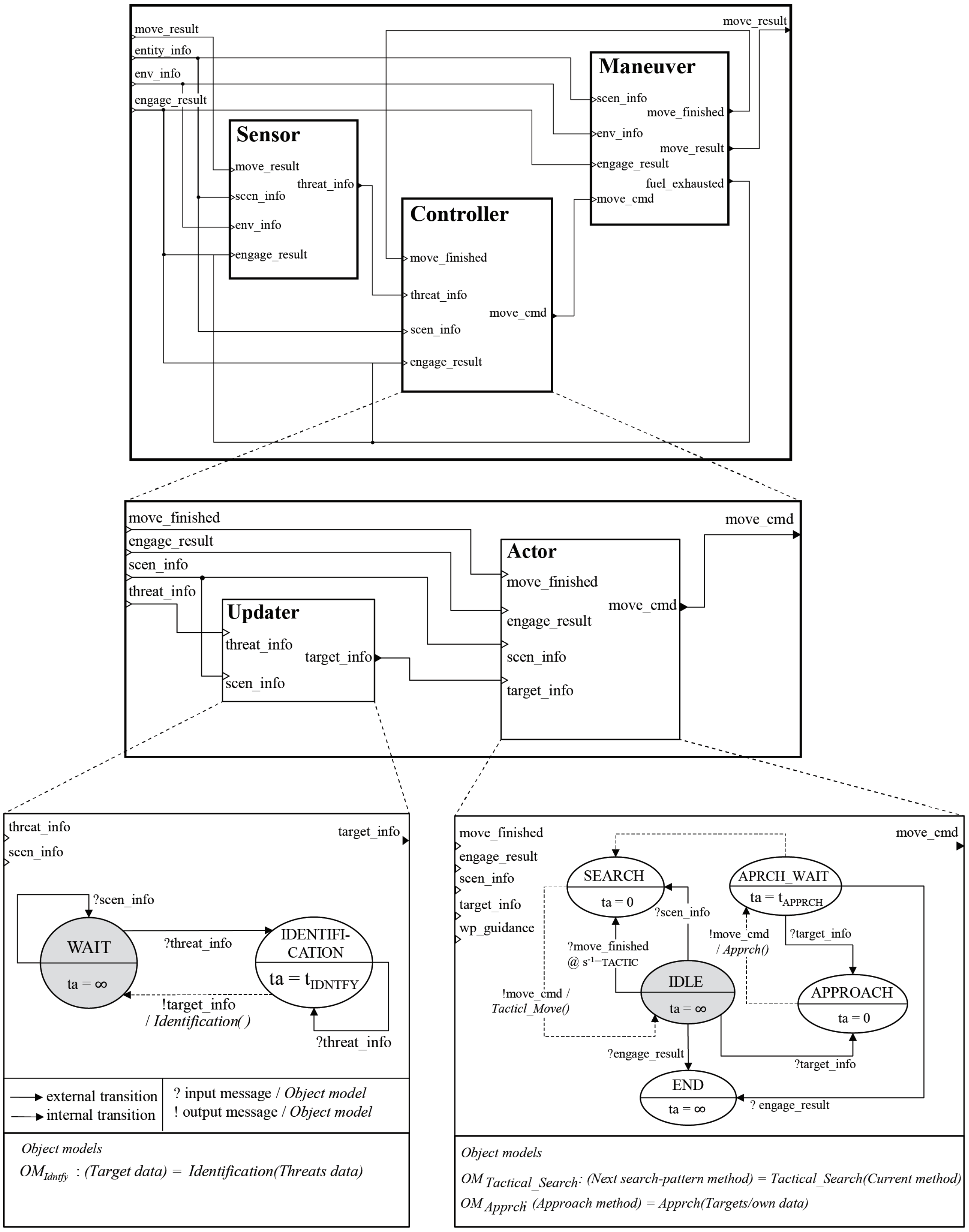
Changbeom Choi received his BS in computer engineering from Kyung Hee University and his MS in computer science from KAIST in 2005 and 2007, respectively. He is currently a PhD candidate at the Department of Electrical Engineering, at KAIST. His research interests include discrete event systems modeling/simulation, verification, validation, and accreditation (VV&A), and Agent based simulation.

Jung Hoon Kim received the BS degree in Oceanography from Seoul National University and the MS degree in Physical Oceanography from Seoul National University, in 1998 and 2001, respectively. Currently, he is a senior researcher at Agency for Defense Development (ADD). His research interests include underwater warfare modeling and simulation for effectiveness analysis.

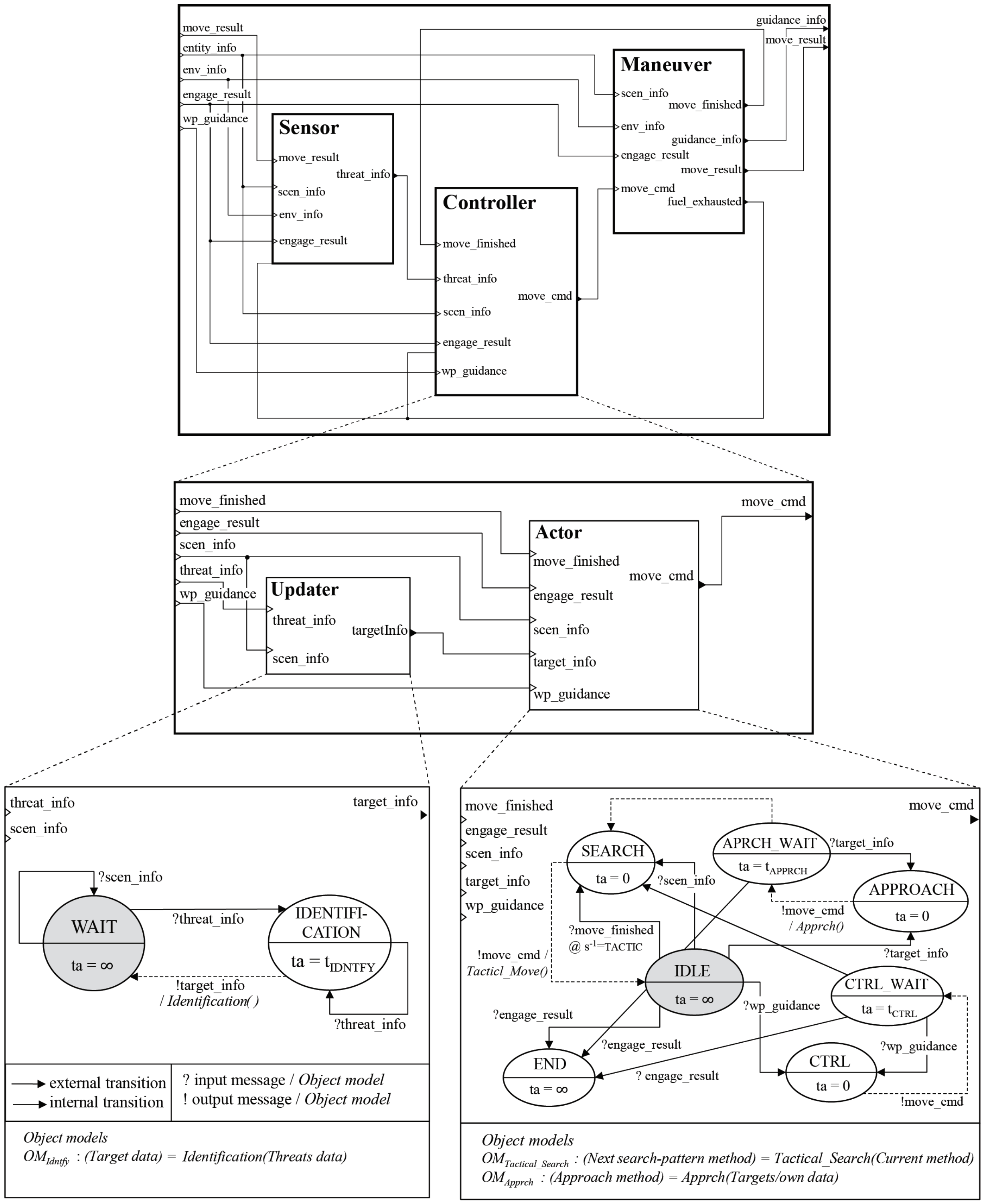
Tag Gon Kim received his PhD in computer engineering with specialization in systems modeling and simulation from University of Arizona, Tucson, AZ, 1988. He was an Assistant Professor at Electrical and Computer Engineering, University of Kansas, Lawrence, Kansas, USA from 1989 to 1991. He joined at Electrical Engineering Department, KAIST, Daejeon, Korea in fall, 1991 as has been a Full Professor at 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 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 modeling and simulation in Korea. He was/is a consultant for defense M&S technology at various Korea government organizations, including Ministry of Defense, Defense Agency for Technology and Quality (DTAQ), Korea Institute for Defense Analysis (KIDA), and Agency for Defense Development (ADD). He is a Fellow of SCS and a Senior Member of IEEE.

Downloaded from sim.sagepub.com at KOREA ADV INST OF SCI & TECH on July 16, 2014

Appendix A: DEVS modeling of weapon model (basic type)



Appendix B: DEVS modeling of weapon model (advanced type)



View publication stats