LDEF Formalism for Agent-Based Model Development

Jang Won Bae and Il-Chul Moon

Abstract—**As agent-based models (ABMs) are applied to various domains, the efficiency of model development has become an important issue in its applications. The current practice is that many models are developed from scratch, while they could have been built by reusing existing models. Moreover, when models need reconfiguration, they often need to be rebuilt significantly. These problems reduce the development efficiency and ultimately damage the efficacy of ABM. This paper partially resolves the challenges of model reusability from the systems engineering approach. Specifically, we propose a formalism-based ABM development and demonstrate its potential to promote model reuses. Our formalism, named large-scale, dynamic, extensible, and flexible (LDEF) formalism, encourages the building of a larger model by the composition of modularly developed components. Also, LDEF is tailored to the ABM contexts to represent the agent's action procedure and support the dynamic changes of their interactions. This paper shows that LDEF improves the model reusability in ABM development through its practical examples and theoretical discussions.**

Index Terms—**Agent-based model (ABM) formalism, efficient ABM development, formalism-based model development, model reusability.**

I. Introduction

AGENT-based models (ABMs) have been successfully applied to understand, analyze, and predict system behaviors in various domains, such as sociology [1], biology [2], management [3], economics [4], military science [5], urban-growth [6], and logistics [7]. This paper proposes an ABM formalism called large-scale, dynamic, extensible, and flexible (LDEF) formalism. Each concept of LDEF is introduced as follows.

1) *Large-Scale:* LDEF supports the development of a model containing a large number of components.
2) *Dynamic:* LDEF enables the change of model structures during its simulation execution.
3) *Extensible:* LDEF supports building a model by adding another model to an extant model (i.e., incremental modeling).
4) *Flexible:* LDEF supports building a model by changing one component in an extant model with another component (i.e., flexible modeling).

Manuscript received December 26, 2014; revised April 6, 2015; accepted June 13, 2015. Date of publication August 12, 2015; date of current version May 13, 2016. This work was supported by the the Man-Made Disaster Prevention Research Center, National Emergency Management Agency of Korea under Grant "Research and Development of Modeling and Simulating the Rescues, the Transfer, and the Treatment of Disaster Victims" [nema-md-2013-36]. This paper was recommended by Associate Editor L. Cao.

The authors are with the Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Korea (e-mail: repute@kaist.ac.kr; icmoon@kaist.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2015.2461178

In particular, this paper focuses on large-scale and flexible ABM development supported by LDEF. To this end, LDEF promotes model reusability by a modular and hierarchical manner and reflects the details of the ABM contexts including agent-oriented behaviors and dynamic interaction changes to its model specifications.

The success of ABM has led to the development of numerous and complex ABMs [8]. However, in contrast to this prosperity, most ABM developments have been conducted in an inefficient way. Still, many models are independently developed and rarely maintained. Additionally, it is difficult to adopt models from other modelers because there are no standard theories governing interface and interoperation between ABMs. From this reality, we hypothesize that there is room to improve efficiency in the development and the reutilization of the ABMs.

ABM holds a typical constitution where the ABM consists of agents, environments, and their interactions [9]. This stereotype structure denotes that there are many possibilities for reusing the existing ABMs in another ABM development. Such model reuses would reduce time and cost in the ABM development and its maintenance and, furthermore, enable the application of collaborative modeling approaches to ABMs [10].

While modelers have known these obvious advantages, there are several obstacles in the reuse of practical models. One of the significant obstacles is the question of validity and correctness in reutilizing the models [11], [12]. If modelers utilize existing models to build their new models, they should require assurance that the models will work as they expected. Another obstacle is, although the trust in the behaviors of the reused models is implicitly assumed, a lack of modeling methods that theoretically and technically support model reuses in ABM development. One feasible solution to overcome these hurdles is to apply formal methods to the ABM development as the observed advantages in the discrete event modeling domains [13].

Formal methods, or formalisms, have been used to unambiguously specify system behaviors, and this trait enabled the applications to requirement analysis, development, and verification of systems [14]. Moreover, this unambiguity makes the formal methods to be considered as methods supporting model reusability. For instance, Chu and Yang [15] addressed that model reuse is stimulated by adopting formalisms in model development.

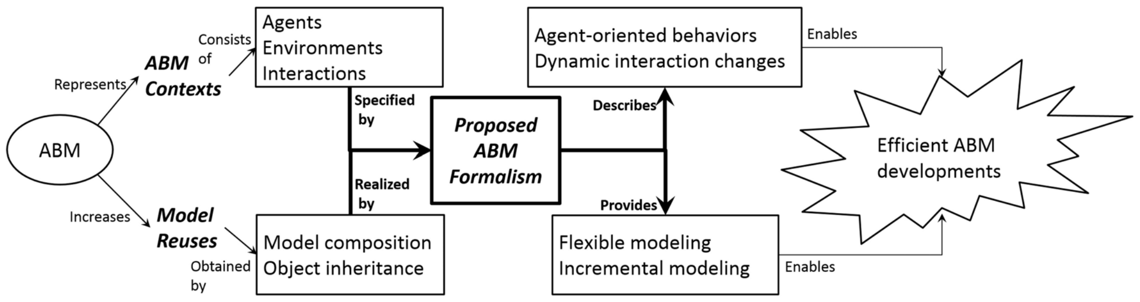


Fig. 1. Efficient ABM development through formalism-based representation of ABM contexts and increasing model reuses.

In a similar trend, ABM researchers have partially used formal methods from two major perspectives: 1) systems engineering and 2) artificial intelligence (AI). While the system engineering field focuses on the structured modeling methods to increase model reuses, the AI field concentrates on the details of ABM contexts including agent, environment, and their interactions. However, there is no agreement within the community of ABMs on a universal formalism to describe ABM.

An efficient ABM development is closely associated with encouraging model reuses and explicitly representing ABM contexts at the same time. To realize those conditions, formalism needs to play an important and dual role. The first role is fitting into the right semantic structure to embody the generally acceptable ABM contexts. The second role is providing the syntax to promote model reuses by supporting model compositions and object inheritances. These two roles together allow modelers to describe the detailed ABM contexts and to exploit efficient modeling methods, such as flexible and incremental modeling. At the focal point of this expectation, formalism would be a bridge between the explicit ABM context and the efficient ABM development (see Fig. 1).

This pper also presents theoretical and practical explanations to show the validity and the efficiency of LDEF. The validity of LDEF is examined by the system morphism approach [13], [16] to compare and contrast the existing and established formalism. To demonstrate the efficiency of LDEF, we measured the model reusability using two examples and discussed how the model reusability is secured by LDEF formalism.

II. Related Work

Various formal methods have been invented and applied to ABM developments from systems engineering and AI fields. While the systems engineering field has focused on how to efficiently construct the ABM structure, the AI field has been concerned with delineating the ABM contexts. This section reviews the strengths and the weaknesses of those formalisms.

A. ABM Formalisms From Systems Engineering Field

Systems engineers are interested in developing a general modeling method, so they consider a system as a black box with input/output ports. Based on this concept, they focus on how to understand, analyze, and predict the overall system behaviors. To support these considerations, the concept of model composition and decomposition is devised, and it is well represented in discrete event system specification (DEVS) formalism [13].

Due to the generality of the DEVS, it has been applied to develop several ABM applications [17]–[19], invent formal methods for ABM development [20], [21], and construct the cognitive process of an agent [22]. In addition, Duboz *et al.* [23] describe agents and multiagent systems using the semantics of dynamic structure DEVS (DS DEVS) [24]. We reviewed some of these formalisms by comparing their elements with the ABM contexts (see Table I). They seem to represent most of the contents by exploiting their modeling components. In particular, their specifications about the agent interactions are apposite to describe large-scale ABMs. Among them, cell-DEVS [25] seems to be one of the most comprehensive formalisms for ABM.

Nonetheless, these formalisms have not been widely used in ABM domains. We suspect that one reason is because there is a significant barrier to adopting the DEVS-oriented formalisms in ABM developments. For instance, the agent should autonomously perceive, decide, and act upon the outside stimuli. DEVS and its extensions represent these behaviors with the terms they have used, such as state transitions and coupling relations. However, their terms are not explicitly associated with the ABM terms so that ABM developers would experience the difficulty of understanding and utilizing these formalisms.

B. ABM Formalisms From AI Field

Contrary to the systems engineering field, formalisms in the AI field have focused on the detailed descriptions of the ABM contexts. These formalisms are often used to develop prescriptive models that generate model behaviors through their inferences. Their prescriptive property results in their ability to be adaptable to the ABMs in terms of their taxonomic wording. For example, they named their elements as actions, observations, utility, and so on, which are naturally understood by ABM modelers, unlike DEVS's taxonomy.

The classic formalisms, such as partially observable Markov decision process [26], belief–desire–intention [27], and game theory [28], were more concerned about the agent behaviors rather than the environment and the interaction features. However, several recent formalisms make up for these limitations and, moreover, consider additional features of ABM. One example of the additional features is dynamic interaction

TABLE I
MAPPING THE GENERAL TERMINOLOGY IN THE ABM CONTEXTS WITH ELEMENTS IN DEVS-ORIENTED FORMALISMS
(ITALIC AND BOLD FIGURES ARE THE GENERAL TERMINOLOGY IN THE ABM CONTEXTS)

Categories of tuples in formalisms		DEVS oriented formalisms		
		DEVS Formalism	Mobile-DEVS formalism	Cell-DEVS formalism
Agent	Input or ***Perception***	Input event set	Input event set, Structure change events set	Input event set, Input event values
	Output or ***Action***	Output event set	Output event set	Output event set
	State	State set	State set	State set and values, Delay, Neighborhood size
	State transition or ***Decision***	External transition function, Internal transition function	External transition function, Internal transition function	External transition function, Internal transition function, Local computation function
	Output function or ***Action***	Output function, Time advance function	Output function, Time advance function	Output function, Transport delay, State's duration function
Environment				Cell space, Border cells
Multi-Agent Coupling Structure or ***Interactions***	Component or ***Population***	Component model set	Activated model set, Model activation function	Neighborhood set
	Coupling Relations or ***Neighborhood, Boundary***	External input couping relation, External output coupling relation, Internal coupling relation	Structure state set, Structure transition function, External input coupling relation, External output coupling relation, Internal coupling relation, Change coupling relation	Input coupling list, Output coupling list, Translation function

TABLE II
FORMAL LANGUAGES AND EXPRESSIVE POWER OF THE RECENT FORMALISMS APPLIED FROM THE AI FIELD; 'O' INDICATES A CONSIDERED ELEMENT IN FORMALISM, AND 'X' INDICATES A NOT-CONSIDERED ELEMENT IN FORMALISM

Literature	Formal specification	Expressive power for ABM contexts (O: considered element, X: not-considered element)			
		Agent	Environment	Interaction	Dynamic changes
Luck and D'Inverno [36]	Z Specification	O	X	O	X
Silva and Paulino [37]		O	X	O	X
Niazi and Hussain [38]		O	X	X	X
Fisher and Wooldridge [39]	Temporal belief logic	O	X	O	X
Conrad *et al.* [40]	Evolving temporal logic	O	X	O	Partially (behaviors)
Zhu [41]	Temporal logic	O	O	O	X
Ricci *et al.* [31]	Process algebra	O	O	O	O
Brazier *et al.* [42]	DESIRE	O	O	O	X
Lomazova [43]	Nested petri net	O	X	O	X
Bauer *et al.* [44]	AgentUML	O	X	O	X
Bentahar *et al.* [45]	Network structure	X	X	O	Partially (interactions)
Chatfield *et al.* [46]	Multi formalism	O	X	O	X
Marzougui *et al.* [47]	Agent petri net	O	X	O	X
Wu *et al.* [48]	ADAM	O	X	O	X

changes, which indicates that the relationships between agents and environments change over time. Moreover, the dynamic interaction changes have become noteworthy in ABM because such dynamics are closely bound to the self-organization and the emergence concepts, which provide invaluable insights for ABM [29], [30].

Among the recent formalisms in Table II, Ricci *et al.* [31] show the broadest coverage for the ABM contexts including the dynamic interaction changes. In their previous works [31]–[35], they suggested the concept of agent-coordination contexts for the infrastructure of the interactions between agents and environments, which is implemented by process algebra.

In contrast to the detailed descriptions about the ABM contexts, formalisms in the AI field have less considerations about an efficient ABM development. More specifically, their specifications are inappropriate for developing an organized ABM, which is the main drawback of these formalisms when developing complex ABMs.

To sum up, formalisms from the systems engineering field encourage model reuse and support dynamic changes (see Table I), but their scope of reusable components in ABM development would be limited because of their limitations on describing the ABM contexts. Meanwhile, formalisms from the AI field have advantages for representing the ABM contexts, so they would hold a wider scope of reusable components in ABM developments. However, they would be limited to facilitating model compositions in the model development and to representing their structural changes (see Table II).

While those formalisms have been improved by enhancing their own strengths, to promote an efficient ABM development, these advantages should be synthesized into a single formal method. Hence, this paper proposes a new formal method for ABM called LDEF formalism. To embrace the two benefits, the LDEF borrows modular and hierarchical modeling from DEVS, and it reflects the general ABM contexts and the dynamic interactions on its formal specifications.

III. LDEF Formalism

This section introduces the concept and the constitution of LDEF formalism and shows how LDEF captures the ABM contexts using its elements.

A. Concept of LDEF Formalism

LDEF formalism is suggested to adopt systems engineering techniques for ABM development. The systems engineering techniques mainly focused on efficient model developments utilizing structured modeling methods. Such considerations have become indispensable in the current ABMs where numerous agents and environments exist.

To embody the efficient ABM developments, LDEF introduces modular and hierarchical modeling concept from the DEVS formalism. For a modular form, components in the LDEF are designed with interfaces, and all interactions between other models are performed only through the interfaces. From the systematic view, such interfaces would include inputs and outputs of a model.

Also, for a hierarchical form, LDEF distinguishes the ABM constitution into structural parts and behavioral parts. Eventually, a model would be decomposed by many structural and behavioral parts (i.e., model decomposition) and be constructed hierarchically by combining these components (i.e., model composition).

LDEF also reflects the ABM contexts in its formal specifications. Fig. 2 represents the standardized structure of ABM: the ABM holds agents, environments, and their interactions. LDEF considers the ABM contexts as actions and environment elements (e.g., behavioral elements); agents, environments, multiagents, and multienvironments (e.g., structural elements) (see Fig. 2).

As the behavioral elements in LDEF, we considered that ABM contains two kinds of objects: 1) agent and 2) environment. Conceptually, the only difference between them is that the environment has no autonomous features; in other words, the agent perceives an outer event, decides its action policy with its own status, and acts out the decided action onto other objects. However, the environment only reacts to outside stimuli with a simple automaton. To formally separate these objects, LDEF has two behavioral elements for the behaviors of the agent and the environment.

The structural elements in LDEF illustrate interactions among their components in ABM. Due to the two kinds of objects, the interactions are also distinguished into three types: 1) between agents; 2) between environments; and 3) between agents and environments. These interactions are hierarchically positioned according to the ABM contexts (see Fig. 2). These separated interactions facilitate the associated models to be reused as coarse-grained components in other ABM developments. Moreover, through the dynamic interaction changes, the structural elements would delineate the self-organization, the emergence, and a flexible agent, whose actions are changed over time in ABMs.

B. Elements in LDEF Formalism

LDEF has two kinds of elements for specifying the behavioral and the structural parts of ABM. The behavioral elements specify the object behaviors in ABM, while the structural elements describe how these objects are interconnected and how their relationships are changed.

1) Behavioral Elements in LDEF Formalism: The objects in ABM are separated into agents and environments so that LDEF contains two behavioral elements. The behavioral element for the agent is called the action model (ACT). The action model illustrates a general procedure of an agent's action, so it shows how the agent perceives, decides, and acts. In order to explicitly specify this action procedure, we separate the states of the action model into three types.

1) *States of Situation Awareness (S_{aw}):* States about the external information of the agent.
2) *States of Condition (S_{cond}):* States describing conditions for the agent's decision-making.
3) *States of Action (S_{act}):* States indicating which action the agent will execute.

Based on these states, four functions are defined in the action model: 1) perceive and 2) decide functions for the state transition functions; 3) act function for the output function; and 4) time advance function for the next acting time of the agent. Three states and their associated functions are based on the skeleton of a simple reflex agent [9] and the human cognition process [49]. Such explicit descriptions about agent behaviors bring the difference with the atomic model in DEVS. The mathematical representation of the action model is as follows:

$$\mathrm{ACT} = <X, Y, S_{\mathrm{aw}}, S_{\mathrm{cond}}, S_{\mathrm{act}}, P, D, A, \mathrm{ta}>$$

where X = a set of input events; Y = a set of output events; S_{aw} = a set of situation awareness states; S_{cond} = a set of decision condition states; S_{act} = a set of action states; $P : X \times Q \rightarrow S_{\mathrm{aw}} \times S_{\mathrm{cond}}$, a function for agent's perception, where $Q = \{(s, e) | s \in S_{\mathrm{aw}}, 0 \leq e \leq \mathrm{ta}(S_{\mathrm{act}})\}$; $D : S_{\mathrm{aw}} \times S_{\mathrm{cond}} \times S_{\mathrm{act}} \rightarrow S_{\mathrm{act}} \times S_{\mathrm{cond}}$, a function for agent's decision process; $A : S_{\mathrm{act}} \rightarrow Y$, a function for agent's action; $\mathrm{ta} : S_{\mathrm{act}} \rightarrow R^{+}$, a function for time advance.

The behavioral part for the environment, called environment element model (EEM), represents a reactive behavior that only responds to the external events. Hence, LDEF adopted formal specifications of the atomic model in DEVS for the EEM

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

where X = a set of input events; Y = a set of output events; S = a set of states; $\delta_{\mathrm{ext}} : Q \times X \rightarrow S$, a function for external state transition, where $Q = \{(s, e) \mid s \in S, 0 \leq e \leq \mathrm{ta}(s)\}$; $\delta_{\mathrm{int}} : S \rightarrow S$, a function for internal state transition; $\lambda : S \rightarrow Y$, a function for output; $\mathrm{ta} : S \rightarrow R^{+}$, a function for time advance.

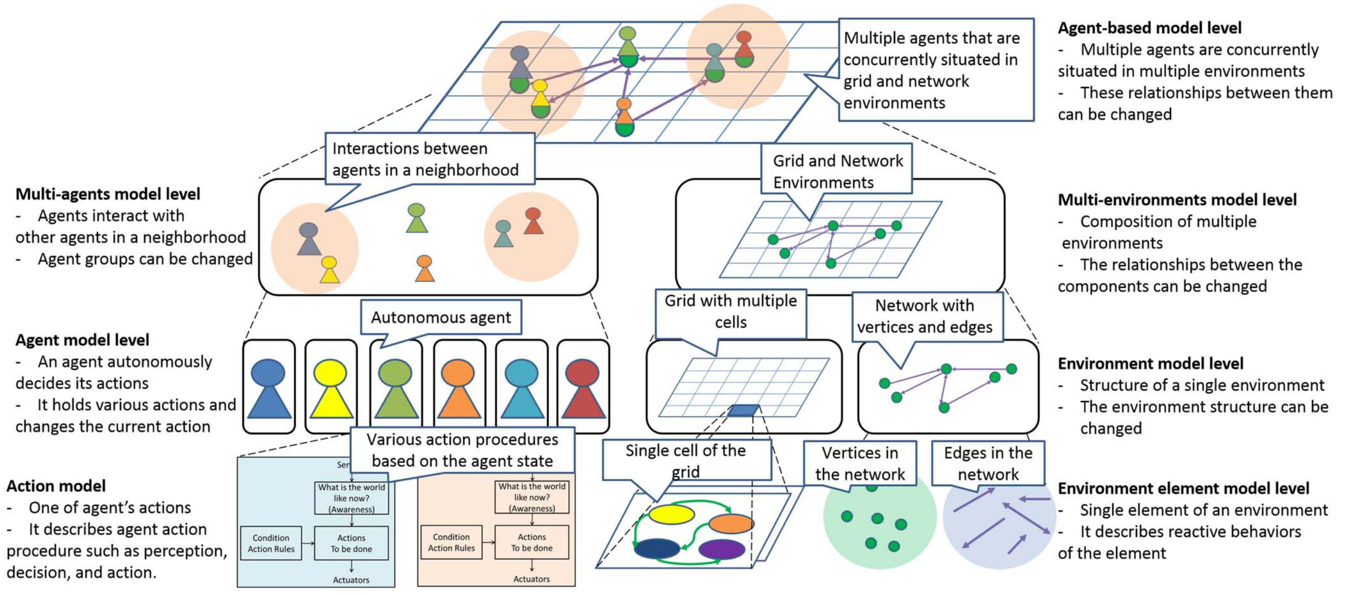

Agent-based model level
- Multiple agents are concurrently situated in multiple environments
- These relationships between them can be changed
Multiple agents that are concurrently situated in grid and network environments
Interactions between agents in a neighborhood
Grid and Network Environments
Multi-agents model level
- Agents interact with other agents in a neighborhood
- Agent groups can be changed
Multi-environments model level
- Composition of multiple environments
- The relationships between the components can be changed
Autonomous agent
Grid with multiple cells
Network with vertices and edges
Agent model level
- An agent autonomously decides its actions
- It holds various actions and changes the current action
Environment model level
- Structure of a single environment
- The environment structure can be changed
Various action procedures based on the agent state
Single cell of the grid
Vertices in the network
Edges in the network
Action model
- One of agent's actions
- It describes agent action procedure such as perception, decision, and action.
Environment element model level
- Single element of an environment
- It describes reactive behaviors of the element
What is the world like now? (Awareness)
Condition Action Rules
Actions To be done
Actuators

Fig. 2. Conceptual illustration of the ABM contexts; the ABM contexts are hierarchically formed, and each component consists of its subordinates.

2) Structural Elements in LDEF Formalism: While the interactions of the classic ABM were regarded as couplings between agents and environments only [50], [51], LDEF considers the interactions as the collaborations between the objects in the ABM including dynamic interaction changes.

The structural element in LDEF, named dynamic structural model (DSM), is an abstract model that formally resolves the above issues. The DSM specifies its component models (M) and how they are interconnected, which is represented as the state-based coupling structure (sCS). For the dynamic interactions, we supplemented coupling states (S_{c}) and a state transition function (δ_{CS}) to the DSM. The state transition function is only triggered by the events from its component models (see δ_{CS} in the below specification).

The sCS describes the relations between coupling states (S_{CS}) and coupling structures (CS). Each coupling structure contains its own component models (M_{s}), coupling relations ($\mathrm{EIC}_{\mathrm{s}}, \mathrm{EOC}_{\mathrm{s}}, \mathrm{IC}_{\mathrm{s}}$), and a tie-break function ($\mathrm{SELECT}_{\mathrm{s}}$). Therefore, by changing the coupling states of the DSM, its coupling structure is also changed by the coupling relation associated with the updated coupling state. The mathematical representation of the DSM is as follows:

$$\mathrm{DSM} = <X, Y, M, S_{\mathrm{CS}}, \mathrm{sCS}, \delta_{\mathrm{CS}}>$$

where X = a set of input events; Y = a set of output events; M = a set of component models; S_{CS} = a set of coupling structure states; $\mathrm{sCS} \subseteq S_{\mathrm{CS}} \times \mathrm{CS}$, a set of relations for state-based coupling structure; where $\mathrm{CS} = \{M_{\mathrm{s}}, \mathrm{EICs}, \mathrm{EOCs}, \mathrm{IC}_{\mathrm{s}}, \mathrm{SELECT}_{\mathrm{s}}\}$ and $s \in S_{\mathrm{CS}}$, a set of coupling relations, where $M_s \subseteq M$, $\mathrm{EIC}_{\mathrm{s}} \subseteq X \times \cup_{m \in M_s} m.X$, where $m.X$ is input events of a component model $m \in M_s$, external input coupling relations; $\mathrm{EOC}_{\mathrm{s}} \subseteq \cup_{m \in M_s} m.Y \times Y$, where $m.Y$ is output events of m, external output coupling relations; $\mathrm{IC}_{\mathrm{s}} \subseteq \cup_{m \in M_s} m.Y \times \cup_{n \in M_s} n.X$, where $m.Y$ and $n.X$ are input/output events of different component models ($m \neq n$), internal coupling relations; $\mathrm{SELECT}_{\mathrm{s}} : 2^{M_s} - \emptyset \rightarrow M_s$, a tie-breaking function among components M_s; $\delta_{\mathrm{CS}} : \cup_{m \in M_s} m.Y \times S_{\mathrm{CS}} \rightarrow S_{\mathrm{CS}}$, a function of coupling structure state transition.

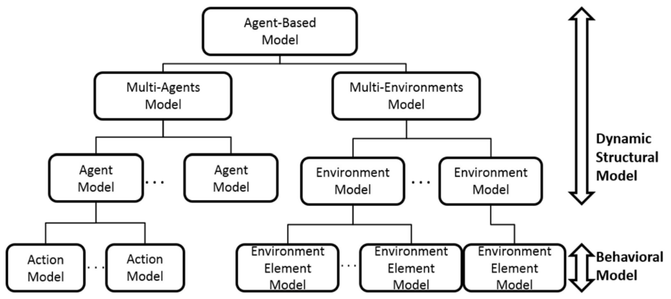


Fig. 3. Applying LDEF elements to describe the ABM hierarchical structure: leaf nodes correspond to the behavioral elements, and non-leaf nodes correspond to the structural elements.

C. Describing ABM Contexts Using LDEF

When we develop an ABM using LDEF, the LDEF elements represent components in the ABM contexts: corresponding to the ABM contexts in Fig. 2, the behavioral and the structural elements in LDEF are hierarchically composed to represent an ABM (see Fig. 3). In the hierarchical structure, an ABM is described as a composition of agent-side and environment-side models. Each component model is also constructed by a recursive model composition of the subordinated components.

Nodes in the hierarchical tree correspond to the elements in LDEF: the leaf nodes correspond to the action model and the EEM, and the nonleaf nodes associate with the DSMs.

Also, the edges represent hierarchical composition relations. For example, action models are components of an agent model. By these mappings, LDEF resolves the considerations depicted in Fig. 2. For example, the agents in Fig. 2, which change their action polices according to their own conditions, are embodied as the DSM that contains multiple action models (e.g., agent model in Fig. 3).

Based on the hierarchical tree in Fig. 3, the semantics of LDEF capture the significant concepts of ABM, such as emergence and collectives. In [52], emergence is defined as a system-level behavior from adaptive traits of individual agents, and collectives include that the interactions among its agents are strong, while the external interactions are weak. A trait corresponds to an action model, so the adaptive traits are described by structural changes of an agent model in Fig. 3. Also, collectives are illustrated by a multiagents model in Fig. 3, and its characteristics are depicted by structural changes of the multiagents model. Eventually, by gathering such collectives, emergence of an ABM would be realized during simulation execution.

Besides, by the virtue of the modular form of the LDEF models, subtrees in the hierarchical tree can be reused, which enables the coarse-grained model reuses. For example, by composing the existing multiagents model and newly developed multienvironments model, the new ABM is created with lower costs. These advantages would be illustrated through practical examples in the next section.

D. Comparing Formalisms Using System Morphism

Before the practical examples, this section presents theoretical considerations about LDEF formalism. LDEF is a kind of DEVS extension that supplements the agent-oriented features with its formal specifications. That fact, however, does not back up the validity and the correctness of LDEF. Therefore, this section shows the theoretical basis of LDEF by comparing LDEF with other formalisms using the concept of system morphism.

1) System Morphism: Morphism is generally used to capture similarity between two objects. The morphism concept is extended to a system morphism to explore the relations between two systems [53]. In the system morphism, if inputs, outputs, and states of one system are associated with those of another system by several mapping functions, then two systems are said to be in the homomorphic relationship (see Fig. 4), which means that the behaviors of one system are preserved in another system. Moreover, it is proven that the system morphism is also applied to compare the behavior equivalence among formalisms [13].

By the level of system specifications, the system morphism focuses on different modeling features. For example, when the inputs, outputs, and state transitions of two systems are specified, the system morphism is about the preservation of state transition and output functions between the two systems; on the other hand, when we have the knowledge about the components of two systems and how the components are interconnected, the system morphism focuses on the preservation of the local state transitions, output behaviors, and coupling

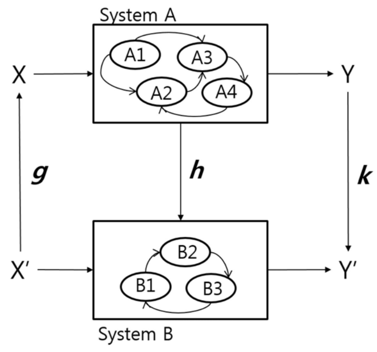


Fig. 4. Concept of system morphism: System *A* is homomorphic to System *B* by the mapping functions (i.e., *g*, *h*, and *k*).

relations between the two systems. Further explanation about such conditions are in [13].

When applying system morphisms to compare LDEF with other formalisms, all of the above conditions should be considered because LDEF describes the former as the behavioral models and the latter as the structural models. The next section, for example, reveals that there is a homomorphic relationship between LDEF and DEVS using the system morphism.

2) Homomorphism Between LDEF and DEVS: Both LDEF and DEVS illustrate a system with behavioral and structural elements. Hence, to show that LDEF is homomorphic to DEVS, it should be proved that system morphisms between the associated elements exist.

For the behavioral elements, LDEF has an action model, and DEVS has an atomic model. While the atomic model has one state set (S_{atomic}), state transition ($\delta_{\text{atomic\_ext}}$ and $\delta_{\text{atomic\_int}}$) functions, and an output function (λ_{atomic}), the action model has three types of states ($S_{\text{action\_aw}}$, $S_{\text{action\_cond}}$, and $S_{\text{action\_act}}$), the associated state transition functions (P_{action} and D_{action}) and output function (A_{action}).

We define a mapping function, h_c, which associates the states of the atomic model with the Cartesian product of the three states of the action model

$$h_c : S_{\text{action}} \rightarrow S_{\text{atomic}}$$

where $S_{\text{action}} = S_{\text{action\_aw}} \times S_{\text{action\_cond}} \times S_{\text{action\_act}}$.

Also, the mapping function should be devised to ensure that the state transitions and outputs in the action model are preserved in the atomic model with the following constraints (see the following equations). Then, we can say that the action model is homomorphic to the atomic model

$$h_c(P_{\text{action}}(S_{\text{action}}, e, x)) = \delta_{\text{atomic\_ext}}(h_c(S_{\text{action}}), e, x)$$

where $x \in X, 0 \leq e \leq \text{ta}(S_{\text{action}})$

$$h_c(D_{\text{action}}(S_{\text{action}})) = \delta_{\text{atomic\_int}}(h_c(S_{\text{action}}))$$
$$A_{\text{action}}(S_{\text{action}}) = \lambda_{\text{atomic}}(h_c(S_{\text{action}})).$$

Considering structural elements, LDEF has a DSM, and DEVS has a coupled model. In contrast to the coupled model, the DSM changes its coupling structures during simulation execution because it contains coupling structure states

($S_{\text{DS\_CS}}$), a state transition function ($\delta_{\text{DS\_CS}}$), and state-based coupling structures (sCS_{DS}). Also, while the state of the coupled model (S_{coupled}) is represented as the Cartesian product of its components, the state of the DSM is expressed as the composition of its coupling structure state and the states of its components. Hence, system morphism between the two models is defined as the following four-tuple <{coord(s)}, {$h_{m'}(s)$}, {$g_{m'}(s)$}, {$k_{m'}(s)$}>:

$$\text{coord}(s) = M_{\text{DS}}(s) \rightarrow M_{\text{coupled}}, s \in S_{\text{DS\_CS}}$$

where $M_{\text{DS}}(s)$ is a subset of M_{DS} associated with a state s in sCS_{DS}

$$h_{m'}(s) = S_{\text{DS}(m')}(s) \rightarrow S_{\text{coupled}}, m' \in M_{\text{coupled}}$$

where $S_{\text{DS}(m')}(s) = s \times \{\times_{m \in M_{\text{DS}(s,m')}} m.S\}$ and $M_{\text{DS}(s,m')} = \text{coord}^{-1}(s, m')$ is a subset of M_{DS} related with m' by coord(s), and $S_{\text{coupled}} = \times_{m \in M_{\text{coupled}}} m.S$

$$g_{m'}(s) = \times_{m \in M_{\text{DS}(s,m')}} m.X \rightarrow m'.X$$
$$k_{m'}(s) = \times_{m \in M_{\text{DS}(s,m')}} m.Y \rightarrow m'.Y.$$

Components in the DSMs are changed by their coupling structure state, so mapping functions for the two models should take this coupling structure state into account. Hence, system morphism between the two models consists of sets of the mapping functions considering state $s \in S_{\text{DS\_CS}}$. Based on this notion, components, state transitions, inputs, and outputs of the two models are associated by the above mapping functions in the system morphism (i.e., {coord(s)}, {$h_{m'}(s)$}, {$k_{m'}(s)$}, {$g_{m'}(s)$}). Furthermore, the system morphism should satisfy the following constraints so that the local state transitions, output behaviors, and coupling relations of the DSM are preserved in the coupled model:

$$\begin{aligned} h_{m'}\Big(\delta_{\text{DS\_CS}}\Big(\cup_{m \in M_{\text{DS}(s,m')}} m.Y \times S_{\text{CS}}\Big) \\ \times\Big\{\times_{m \in M_{\text{DS}(s,m')}} \Delta_{\text{DS}(m)}(m.S, m.X)\Big\}\Big) \\ = \Delta_{\text{coupled}(m')}\Big(h_{m'}\Big(S_{\text{DS\_CS}} \times \Big\{\times_{m \in M_{\text{DS}(s,m')}} m.S\Big\}\Big) \\ g_{m'}\Big(\times_{m \in M_{\text{DS}(s,m')}} m.X\Big)\Big) \end{aligned}$$

where $\Delta_{\text{DS}(m)}(m.S, m.X)$ and $\Delta_{\text{coupled}(m')}(m.S, m.X)$ are the compositions of state transition functions

$$k_{m'}\Big(\times_{m \in M_{\text{DS}(s,m')}} \beta_m(m.S)\Big) = \beta_{m'}\Big(h_{m'}\Big(s \times \Big\{\times_{m \in M_{\text{DS}(m')}} m.S\Big\}\Big)\Big)$$

where $\beta_m(m.S)$ is a function that generates the corresponding output from the component m according to its state

$$\begin{aligned} g_{m'}\Big(\times_{m \in M_{\text{DS}(s,m')}} Z_m\Big(\times_{i \in I_m} \rho_i(i.S)\Big)\Big) \\ = Z_{m'}\Big(\times_{i' \in I_{m'}} k_{i'}\Big(\times_{i \in M_{\text{DS}(s,i')}} \rho_i(i.S)\Big)\Big) \end{aligned}$$

where I_m indicates influencing components to the component m, Z_m indicates interface mappings of events to the component m, and ρ_m is an output value from m according to its state.

Because behavioral and structural parts in LDEF are preserved in DEVS using system morphisms, we can argue that LDEF is homomorphic to DEVS so that the correctness and the validity of LDEF behaviors are guaranteed by those of DEVS.

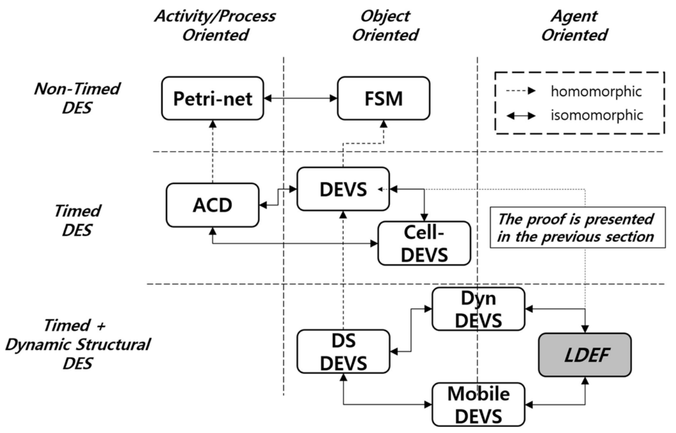


Fig. 5. Homomorphic map among LDEF and DES formalisms from the two perspectives: world views in DES (horizontal) and the time and DS in DES (vertical).

3) Homomorphic Map Among DES Formalisms: In the previous section, we showed that the behaviors from LDEF are preserved in DEVS by their homomorphic relationship. These homomorphism relationships are used not only to support the validity of a new formalism, but also to explore the theoretical position of it. Since the syntactic features in the LDEF are based on discrete event systems (DESs), we investigate homomorphism relationships between LDEF and other DES formalisms.

Formalisms in DES are classified by: 1) world views and 2) time and DSs. Formal specifications on DES are formed differently according to from what viewpoints the structures of DES will be seen, such as activity/process-oriented, object-oriented, and agent-oriented views. Also, there are various formalisms by considering the time and DS features in DES. For example, petri-net has a process-oriented view and is based on nontimed DES, but DEVS has an object-oriented view and is based on timed DES. From those perspectives, we plotted famous formalisms by their homomorphic relationship (see Fig. 5).

Fig. 5 illustrates isomorphic relationships: 1) between petri-net and finite state machine; 2) among activity cycle diagram [54], DEVS, and cell-DEVS; and 3) among DS DEVS [24], Dynamic DEVS (dynDEVS) [21], Moblie DEVS [20], and LDEF. Also, the figure illustrates homomorphic relationships among nontimed DES, timed DES, and timed and dynamic structural DES formalisms.

Also, Fig. 5 represents the theoretical position of LDEF, which holds an agent-oriented view and is based on the timed and dynamic structural DES. LDEF is isomorphic to DS DEVS, DynDEVS, and Mobile DEVS, which means all of them are compatible with each other. However, LDEF is located at the middle of agent-oriented view while others are in the borderline. This contextual difference often provides the extra usages to specific fields. For example, the similar relationships are also observed between DEVS and cell-DEVS. Despite their isomorphic properties, they have been applied to the different domains because the cell-DEVS better connotes the explicit semantics about the cellular-automata compared to DEVS.

In the same context, LDEF explicitly reflects the ABM contexts rather than other isomorphic formalisms. These formalisms often reveal unclear parts in the ABM development because they have deficient semantics in terms of the ABM contexts. On the other hand, LDEF provides the explicit representations about the ABM contexts so that it can close those semantic gaps. Hence, LDEF augments the accessibility of ABM developers who are not familiar with the terms used in the systems engineering fields and, simultaneously, provide the ABM developers with an efficient ABM developing method using modeling techniques from the systems engineering fields.

IV. Examples: Model Reuses in LDEF-Based Model Developments

LDEF formalism has been applied to the model developments of several agent-based systems. This section introduces evacuation agent-based model (EABM) and ambulance agent-based model (AABM) as the example models. With these examples, we illustrate the practical efficiency provided by LDEF.

A. Behavioral Similarity Between EABM and AABM

EABM describes the massive evacuations of citizens through a road network in the Gangnam region due to bombing attacks from a hostile side. Let us briefly introduce an evacuation scenario in EABM. While agents move around in the Gangnam region, the bombs attack the region. These threats destroy parts of the road network and force the agents to evacuate from the region. In the evacuation situation, agents gather to the evacuation spots and thus create bottlenecks in the road network. EABM generates these phenomena by calibrating the agents and bombing factors to analyze the evacuation dynamics under various conditions [55].

AABM depicts the transportation of victims with ambulances from disaster scenes to available hospitals. For a better understanding of its behaviors, we briefly present a scenario in AABM. The scenario assumed that a big fire occurs in the manufacturing quarter near Cheonan and causes numerous casualties. To rescue the victims, a central management unit, called emergency management, dispatches ambulances to transport them to available hospitals in Cheonan. Because each victim has different injuries, there can be numerous rescue plans for various objectives, such as minimizing transportation time and maximizing survival rates. Hence, AABM is used to analyze and optimize the rescue plans on various conditions. The AABM structure is similar to its earlier version [56], yet the present one involves more considerations, such as the hospital agent and ambulance behaviors.

Considering the scenarios of the two models, we found out that AABM has similar behaviors as EABM. For example, evacuation agents in EABM escape from the Gangnam region through the road network and some of them are immovable due to the bombing attacks. Similarly, ambulance agents in AABM transport victims to hospitals through the road network in the target city, and some victim agents are immovable according to their health states. Such similarities permit the components in EABM to be reused in AABM development. Moreover, the role of the road network in AABM is identical to that in EABM because both calculate the shortest path between two points on the network structure.

Also, behavioral similarities can be observed even in the same model. For example, ambulances and hospitals in AABM check the patient status and treat the patient to boost his survivability. Even though the level of treatments from ambulances and hospitals may be different according to the medical devices that they hold, it is certain that they have similar behaviors.

Such similarities between EABM and AABM become the pivot points of model reuse. As they have more similar parts, the model reusability would increase. Hence, we reused the component models from EABM in the development of the similar parts in AABM. The level of model reuse is dependent on the level of their similarity. For example, the road network model in EABM shows identical behavior to the road network model in AABM, so they are compatible in the two models. On the other hand, the job action model in EABM has similar parts as the movement action model in AABM so that the job action model is partially reused.

We present screen shots of simulating EABM and AABM on the LDEF simulation environment in Fig. 6. Fig. 6 illustrates that agents in the EABM either evacuate from the bombs or become immovable because of the bomb damage, and ambulance agents in the AABM approach victim agents to transport them to hospital agents. The screen shots illustrate that the similar behavior components (e.g., evacuation and ambulance agents) are moving along to the road network with their own objectives (e.g., escaping from the bombing regions and approaching the victim agents).

B. Reusing EABM Components in AABM Development

With the behavioral resemblance between EABM and AABM, the EABM components are reusable in the AABM development. To recognize the corresponding parts in EABM, there must be a modeling tool that explicitly represents agent behaviors, such as LDEF formalism. Therefore, we found the reusable components in EABM through its structure.

The structure of EABM forms a hierarchical structure as the general ABM (see Fig. 7). In Fig. 7, nonleaf nodes represent DSMs and leaf nodes represent behavior models (i.e., action model for agent models and EEM for environment models). The top-most model, i.e., EABM, contains multievacuation agents model and multienvironments model, and each involves evacuation agent model, as well as road network and bombardment model, respectively. The evacuation agent holds three action models and changes its behavior patterns by activating one of the action models according to their current state. The road network model and the bombardment model describe the network structure consisting of roads and junctions in the Gangnam region, and bombing attacks and damages to the Gangnam region, respectively.

Fig. 8 shows the internal structure of EABM: where components of EABM are positioned, how the components are interconnected each other, and which components are dynamically changed when bombing attacks begin with dotted

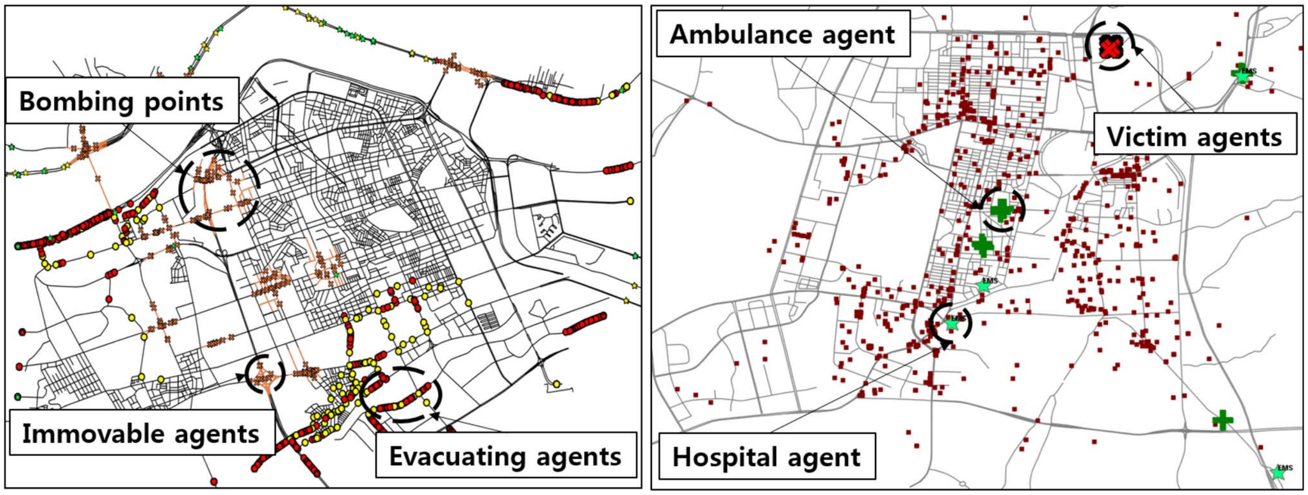


Fig. 6. Screen captures of simulating evacuation agent-based model (EABM) in Gangnam region (left) and ambulance agent-based model (AABM) in Cheonan city (right).

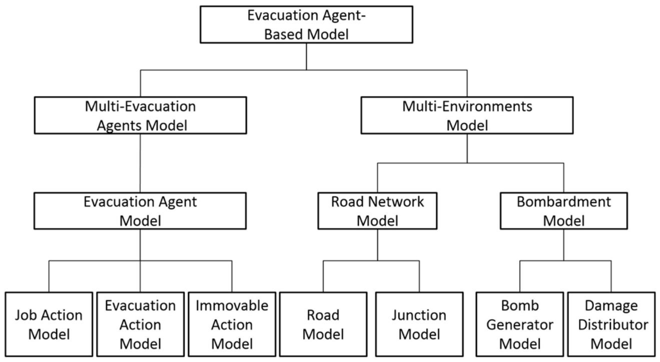


Fig. 7. Model structure of EABM developed by LDEF formalism.

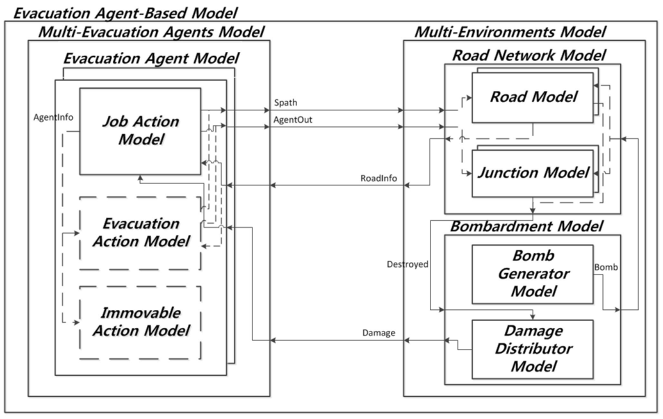


Fig. 8. Coupling structure in EABM: dotted coupling relations and models represent the dynamically changed components.

$$Evacuation\ Agent\ Model\ (EAM) =< X, Y, M, S_{cs}, \delta_{cs}, sCS >$$
$$X = \{RoadInfo, Damage\}$$
$$Y = \{Spath, AgentOut\}$$
$$M = \{JAct, EAct, IAct\}$$
$$S_{cs} = \{Normal, Evacuation, Damage\}$$
$$\delta_{cs}: JAct.AgentInfo \times Normal \longrightarrow Evacuation, if\ the\ agent\ can\ evacuate$$
$$JAct.AgentInfo \times Normal \longrightarrow Damage, otherwise$$
$$sCS = \{(Normal, CS_N), (Evacuation, CS_E), (Damage, CS_D)\}, \text{where}$$
$$CS_N = \{M_N, EIC_N, EOC_N, IC_N, SELECT_N\}:$$
$$M_N = \{JAct\}$$
$$EIC_N = \{(EAM.RoadInfo, JAct.RoadInfo), (EAM.Damage, JAct.Damage)\}$$
$$EOC_N = \{(JAct.Spath, EAM.Spath), (JAct.AgentOut, EAM.AgentOut)\}$$
$$SELECT_N = \{JAct\}$$
$$CS_E = \{M_E, EIC_E, EOC_E, IC_E, SELECT_E\}:$$
$$M_E = \{JAct, EAct\}$$
$$EIC_N = \{(EAM.RoadInfo, EAct.RoadInfo), (EAM.Damage, EAct.Damage)\}$$
$$EOC_N = \{(EAct.Spath, EAM.Spath), (EAct.AgentOut, EAM.AgentOut)\}$$
$$IC_N = \{(JAct.AgnetInfo, EAct.AgentInfo)\}$$
$$SELECT_N = \{JAct, EAct\}$$
$$CS_D = \{M_D, EIC_D, EOC_D, IC_D, SELECT_D\}:$$
$$M_D = \{JAct, IAct\}$$
$$IC_N = \{(JAct.AgnetInfo, IAct.AgentInfo)\}$$
$$SELECT_N = \{JAct, IAct\}$$

Fig. 9. LDEF formal specification of the evacuation agent model in EABM.

lines. Note that for explanation of the internal model structures of examples, this paper adopts DEVS diagrams, but the examples are developed by LDEF formalism. Hence, they should be analyzed using LDEF specifications.

For example, when an agent moves around or evacuates, job action model or evacuation action model asks for the shortest path to road network model using Spath event. The road network model then calculates the shortest path and returns a road list of the shortest path on RoadInfo event. When an agent arrives at the end of a road in the road list, the agent signals to the corresponding road to update its current velocity on AgentOut event. On the environmental side, the damages from the bombing attack are propagated to agents using Damage event so that the agents change their actions according to their damages. The model specification of the evacuation agent model is presented in Fig. 9, and the further information is described in [55].

By the explicit ABM representations from LDEF, we could check the similar components in EABM and reuse them in AABM development. Fig. 10 shows the reused components in AABM development: 1) reused models from EABM with light-grayed model and 2) reused models from AABM with dark-grayed models; Fig. 11 depicts that coupling relations associated with the reused components were also reused in the AABM development, which is supported by the modular property from LDEF. For example, the road network model in EABM and its coupling relations were reused as they are, and parts of the job action model and the immovable action

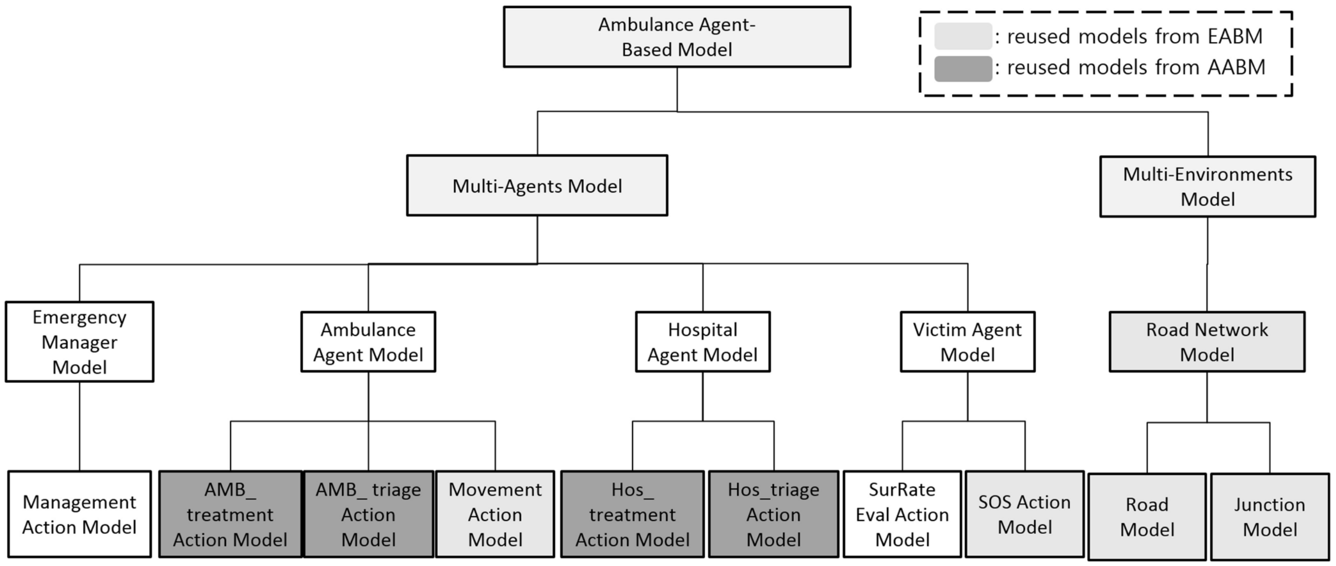


Fig. 10. Reused models in AABM: light-grayed models are the reused models from the EABM, and dark-grayed models are the reused models from AABM.

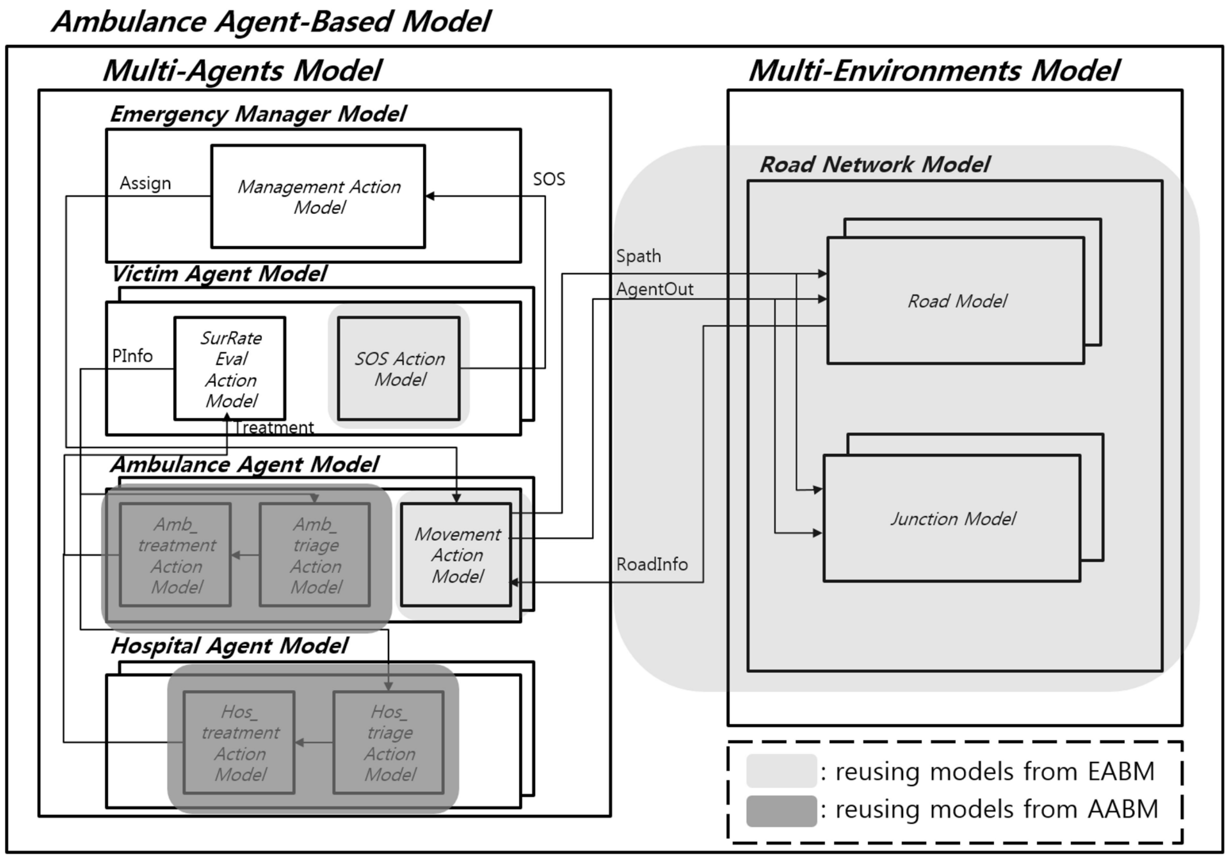


Fig. 11. Reused coupling relations and component models in AABM: light-grayed parts are the reused components from EABM, and dark-grayed parts are the reused components from AABM.

model in EABM were reused in the development of the ambulance agent model and the victim agent model in AABM (see light-grayed parts in Fig. 11). Also, the triage and the treatment action parts in the ambulance model in AABM were reused in the development of the hospital model in the AABM (see dark-grayed parts in Fig. 11).

C. Quantitative Evaluation on Model Reusability

The previous section presents the mapping relations between the existing models from EABM (or AABM) and the newly developed models from AABM, but it is not apparent how much of the efficiency of model reusability is provided from LDEF. Therefore, we quantitatively measured

$$\begin{aligned}
&Job\ action\ = < X, Y, S_{aw}, S_{cond}, S_{act}, P, D, A, ta >\\
&X = \{RoadInfo, Damage\}\\
&Y = \{Spath, AgentOut\}\\
&S_{aw} = \{RoadVelocity\}\\
&S_{cond} = \{EndRoad, Arrived, Damaged, Ready\}\\
&S_{act} = \{Wait, Move, Decide, Notify\}\\
&P{:}\, RoadInfo \times (s,e) \rightarrow RoadVelocity \times Ready\\
&\quad Damage \times (s,e) \rightarrow s \times Damaged\\
&D{:}\, RoadVelocity \times s \times Move \rightarrow Notify \times EndRoad\\
&\quad RoadVelocity \times Ready \times Move \rightarrow Wait \times Arrived\\
&\quad RoadVelocity \times Ready \times Move \rightarrow Move \times Ready\\
&\quad RoadVelocity \times s \times Wait \rightarrow Move \times Ready\\
&\quad s \times Damaged \times s \rightarrow Wait \times Damaged\\
&\quad s \times EndRoad \times s \rightarrow Decide \times EndRoad\\
&A{:}\, Decide \rightarrow Spath\\
&\quad Notify \rightarrow AgentOut\\
&ta{:}\, Decide \rightarrow T_{decide}\\
&\quad Notify \rightarrow 0
\end{aligned}$$

$$\begin{aligned}
&Movement\ action\ = < X, Y, S_{aw}, S_{cond}, S_{act}, P, D, A, ta >\\
&X = \{\boldsymbol{RoadInfo}, PatientInfo\}\\
&Y = \{\boldsymbol{Spath, AgentOut}\}\\
&S_{aw} = \{\boldsymbol{RoadVelocity}, Destination\}\\
&S_{cond} = \{\boldsymbol{EndRoad}, ToPatient, ToHospital, \boldsymbol{Ready}\}\\
&S_{act} = \{\boldsymbol{Wait, Move, Decide, Notify}\}\\
&P{:}\, \boldsymbol{RoadInfo \times (s,e) \rightarrow RoadVelocity \times s}\\
&\quad PatientInfo \times (Ready, e) \rightarrow Destination \times ToPatient\\
&D{:}\, \boldsymbol{RoadVelocity \times s \times Move \rightarrow Notify \times EndRoad}\\
&\quad RoadVelocity \times ToPatient \times Move \rightarrow Decide \times ToHospital\\
&\quad RoadVelocity \times ToHospital \times Move \rightarrow Wait \times Ready\\
&\quad \boldsymbol{RoadVelocity \times s \times Wait \rightarrow Move \times Ready}\\
&\quad Destination \times ToPatient \times s \rightarrow Decide \times ToPatient\\
&\quad s \times ToHospital \times s \rightarrow Decide \times ToPatient\\
&\quad \boldsymbol{s \times EndRoad \times s \rightarrow Decide \times EndRoad}\\
&A{:}\, \boldsymbol{Decide \rightarrow Spath}\\
&\quad \boldsymbol{Notify \rightarrow AgentOut}\\
&ta{:}\, \boldsymbol{Decide \rightarrow T_{decide}}\\
&\quad \boldsymbol{Notify \rightarrow 0}
\end{aligned}$$

$\alpha_{Movement}$	η_X	η_Y	η_{Saw}	η_{Scond}	η_{Sact}	η_P	η_D	η_A	η_{ta}
11.11	0.5	1	0.5	0.5	1	0.5	0.43	1	1

Fig. 12. Reused elements of the job action model in EABM in the development of the movement action model in AABM (bold-figured elements are the reused elements) and their weights for the reuse proportions.

the model reusability during the AABM development by defining the proportion of reuses for each component of the EABM.

The proportion of reuses indicates how much portion of an existing model is reused during another model development. In particular, we note that this measure is calculated only at the model specification level. The proportion of reuses from model n to model m ($P_{m,n}$) is defined as the following equation. Moreover, according to the value of η_e, the level of model reusability is classified as follows: fully reused ($\eta_e = 1$), partially reused ($0 < \eta_e < 1$), and not reused ($\eta_e = 0$).

$$\text{Proportion of reused elements from model } n \text{ to develop model } m (P_{m,n}) = \alpha_m \Sigma_{e\in E_m} \eta_e,$$

where E_m: a set of nonempty elements in m, $\alpha_m = (100/|E_m|)$: a weight that is equivalently distributed to each element in m, $\eta_e = (|m.e \cap n.e|/|m.e|)$, $m.e \in E_m$ and $n.e \in E_n$: a weight for the reuse proportion of an element e in model m.

For a better understanding, let us provide an example of evaluating the proportion of reuses from the job action model in EABM to the movement action model in AABM ($P_{\text{Movement,Job}}$). Since the LDEF action model consists of nine-tuple and elements in the movement action are nonempty, the weight for each element in the movement action (α_{Movement}) is as follows:

$$\alpha_{\text{Movement}} = \frac{100}{9} = 11.11.$$

Then, weights for the reuse portion of all elements, η, are separately evaluated. Fig. 12 shows reuse elements from job action model in the development of movement action model. For example, according to Fig. 12, one input (RoadInfo) of the job action model is reused in the input set (X) of the movement action model, and the weight for the reuse proportion of the input (η_X) is calculated through the following manner:

$$\begin{aligned}\eta_X &= \frac{|\{\text{RoadInfo, PatientInfo}\} \cap \{\text{RoadInfo, Damage}\}|}{|\{\text{RoadInfo, PatientInfo}\}|}\\ &= \frac{1}{2} = 0.5.\end{aligned}$$

Similarly, the weights for other elements are evaluated (the evaluated values are presented in the bottom of Fig. 12). By substituting these weights to the above equation, $P_{\text{Movement,Job}}$ is evaluated as follows:

$$\begin{aligned}P_{\text{Movement,Job}} &= \alpha_{\text{Movement}} \Sigma_{e\in E_{\text{Movement}}} \eta_e\\ &= 11.11 \times \left\{4 \times 1 + 4 \times \frac{1}{2} + 1 \times \frac{3}{7}\right\}\\ &= 71.44\%.\end{aligned}$$

The proportion of reuses for the development of each component in AABM is evaluated, and then by averaging all the evaluated values, the model reusability of AABM is quantitatively represented as 40.80% (see Table III). The fully reused components, with 100% proportion of reuse, possess the identical behavior as the reused model, such as road network, Hos_triage action, road, and junction. Despite the behavioral equivalence, the behavioral details are controlled by their model parameters, which are initially assigned by model users. For example, although the two triage models check the health status of victims, the Hos_triage action model might show better accuracy due to its advanced equipment. Also, the partially reused components explore the incremental and flexible modeling manners. For example, the SOS action model in Table III is developed by adding a help request behavior to the immovable action model in EABM, which exemplifies incremental modeling and is gauged at a relatively low proportion of model reuse (i.e., 33.33%).

TABLE III

PROPORTION OF REUSES FOR COMPONENT MODELS IN AABM: REUSED COMPONENTS ARE FROM EABM OR AABM, AND THEIR ELEMENTS ARE FULLY (I.E., WITHOUT ANY MODIFICATIONS) OR PARTIALLY (I.E., ADDING OR REMOVING ELEMENTS) UTILIZED

Newly developed component in AABM (*m*)	Reused component in EEBM or AABM (*n*)	Elements in the newly developed component		% of reuse ($P_{m,n}$)
		Reused elements (reused size/total size)		
		Fully, or $\eta = 1$	Partially, or $0 < \eta < 1$	
Ambulance Agent-Based	Evacuation Agent-Based	$S_{cs}(1/1), \delta_{cs}(1/1)$	none	50.00%
Multi-Agents	Multi-Evacuation Agents	$Y(2/2), S_{cs}(1/1), \delta_{cs}(1/1)$	$X(1/2)$	58.33%
Multi-Environments	Multi-Environments	$X(2/2), S_{cs}(1/1), \delta_{cs}(1/1)$	$Y(1/2), M(1/2)$	66.67%
Emergency Manager	–	none	none	0.00%
Ambulance Agent	–	none	none	0.00%
Hospital Agent	–	none	none	0.00%
Victim Agent	–	none	none	0.00%
Road Network	Road Network	$X(2/2), Y(1/1), S_{cs}(1/1), \delta_{cs}(1/1), M(2/2), sCS(5/5)$	none	100.00%
Management Agent	–	none	none	0.00%
Amb_treatment Action	–	none	none	0.00%
Amb_triage Action	–	none	none	0.00%
Movement Action	Job Action	$Y(2/2), S_{act}(4/4), A(2/2), ta(2/2)$	$X(1/2), S_{aw}(1/2), S_{cond}(2/4), P(1/2), D(3/7)$	71.44%
Hos_treatment Action	Amb_treatment Action (AABM)	$X(1/1), S_{aw}(1/1), P(1/1)$	$Y(1/3), S_{cond}(1/3), S_{act}(1/4), D(1/3), A(1/3), ta(1/3)$	54.63%
Hos_triage Action	Amb_triage Action (AABM)	$X(1/1), S_{aw}(2/2), S_{cond}(3/3), S_{act}(3/3), P(2/2), D(3/3), A(3/3), ta(3/3)$	none	100.00%
SurRate Eval Action	–	none	none	0.00%
SOS Action	Immovable Action	$S_{cond}(2/2)$	$S_{act}(1/2), A(1/2), D(2/4), ta(1/2)$	33.33%
Road	Road	$X(2/2), Y(1/1), S(4/4), \delta_{ext}(2/2), \lambda(1/2), ta(2/2)$	none	100.00%
Junction	Junction	$X(2/2), S(2/2), \delta_{ext}(2/2), ta(1/1)$	none	100.00%
Average of $P_{m,n}$				40.80%

V. DISCUSSION ON LDEF FORMALISM

The theoretical basis and the practical efficiency of LDEF formalism are presented in the previous sections. One remaining issue is how the model reusability can be facilitated by LDEF formalism. The model reusability from formalism was referred through the concept of two-way model reusability [57]. Fig. 13 represents the two-way model reusability: from formal method and from object-oriented paradigm (OOP). From the perspective of formalism, model reuse is embodied via model compositions. On the other hand, from the perspective of the OOP, model reuse is realized by class inheritance.

Specifically, the model reusability from formalism is enabled by its syntactic and semantic features. The syntax of the formalism enables the model composition, so we are able to reuse models by coupling an existing model with a new model to build a larger, yet new model, which is stated in the two-way model reusability. On the other hand, the semantics of the formalism determines the scope of the reusability of the model. In the traditional DEVS, the internal transition and the external transition jointly modeled the perception and the decision together. Therefore, it was difficult to reuse those parts in another ABM. However, LDEF separates the meanings of the perception and the decision behavior to be reused in the future. From this perspective, dynDEVS formalisms, such as DS DEVS, Mobile DEVS, and dynDEVS, and LDEF would be similar in terms of the expression, but the LDEF formalism focuses on the semantics in ABM compared to the dynDEVS formalism and their families. Lastly, the reusability from the semantics is different from the model composition with the syntax because the realization of the semantic reuses are done through the inheritance of the object-oriented implementation of models (see examples in Fig. 13).

Hence, this section argues about how the syntactic and semantic features of LDEF enhances the model reusability in the ABM development. Also, the limitations of LDEF are discussed for further improvement in the ABM development using LDEF.

A. Model Reusability from LDEF Formalism

1) Relationships Between Model Reusability and Syntactic Features in LDEF: Model composition indicates how the multiple components are assembled to a single one, and, on the contrary, model decomposition means how a model is broken into multiple components. For example, Fig. 14 illustrates that model *AB* is decomposed into model *A* and *B* (i.e., model decomposition) and is also a component of model *ABC* (i.e., model composition).

These concepts help us to see a complex system as multiple interconnected components, so they are quite suitable for ABM development because ABM behaviors are represented by the interactions among their component behaviors, rather than their explicit descriptions.

To apply these concepts in the ABM development, component models should satisfy two conditions: modular components and closure under coupling. Modular models mean their inside and outside are strictly separated, and closure under coupling [13] guarantees that composition of models results in another component model. These two conditions could be satisfied under LDEF formalism: LDEF explicitly distinguishes

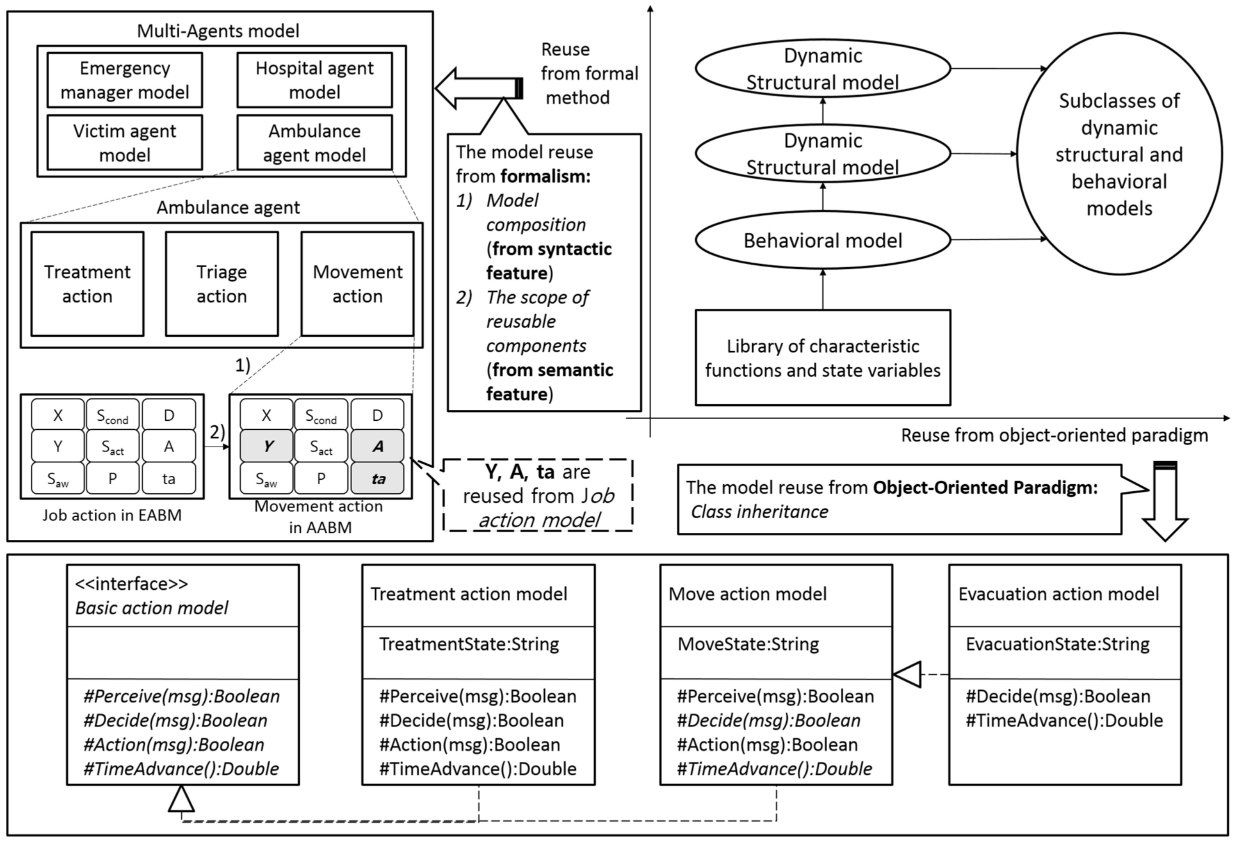


Fig. 13. Two-way model reusability from formal method and object-oriented paradigm. In particular: 1) syntax features in the formal method supports model reuse by model composition and 2) semantic features in the formal method affect the scope of reusable components.

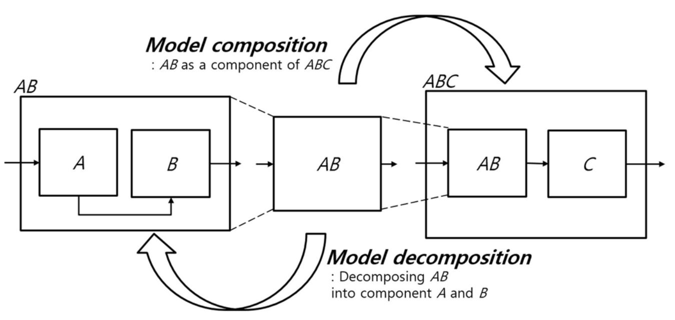

Model composition
: AB as a component of ABC
AB
A
B
AB
ABC
AB
C
Model decomposition
: Decomposing AB into component A and B

Fig. 14. Examples of model composition and decomposition: Model *AB* is decomposed by Models *A* and *B*, and also is a component of Model *ABC*.

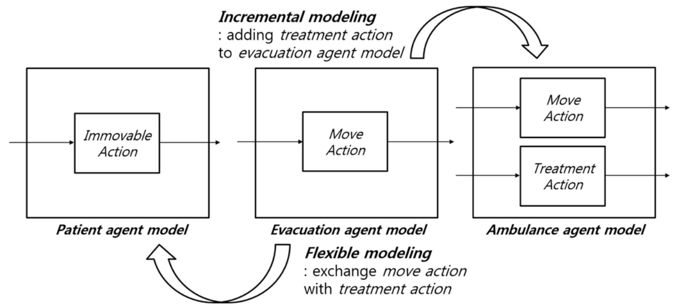


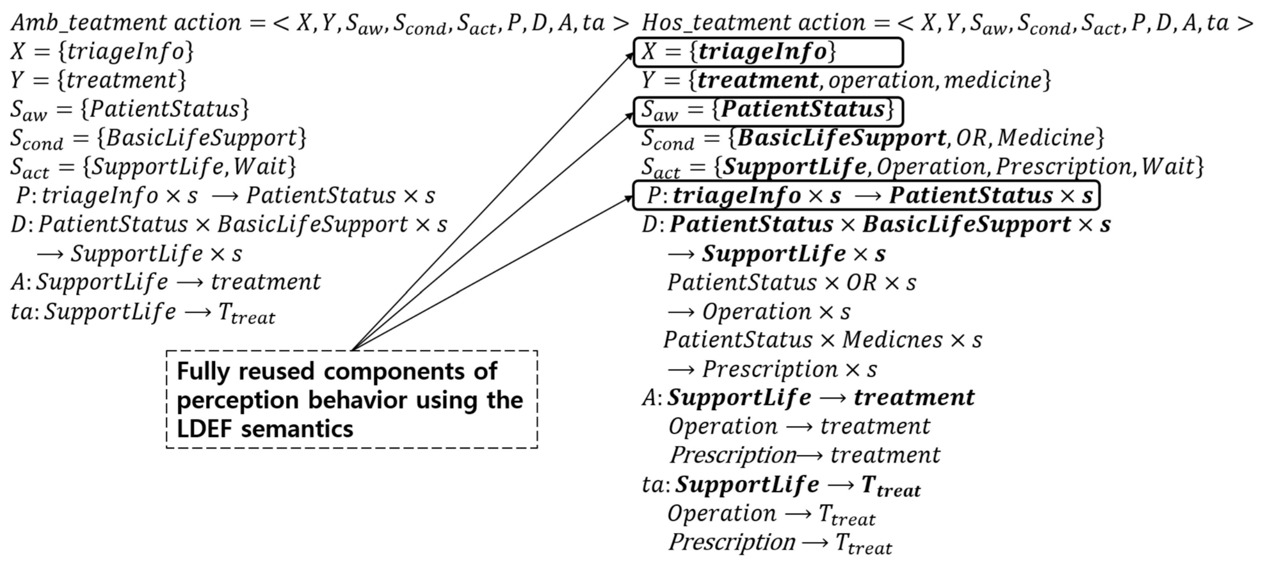
Fig. 15. Examples of incremental and flexible modeling using modular components.

the model interface and states using its mathematical representations and supports closure under coupling among DSMs using DEVS properties.

With modular and reusable components, modelers facilitate an efficient model development by following modeling techniques: incremental modeling and flexible modeling. Incremental modeling means that a new model is developed by adding extra component models to the existing model, and flexible modeling indicates that a new model is created by exchanging component models in the existing model with other component models. Fig. 15 exemplifies incremental modeling (e.g., adding treatment action to evacuation agent for the development of ambulance agent) and flexible modeling (e.g., exchanging move action model from evacuation agent with immovable action for the development of patient action model).

Furthermore, we consider an ABM framework that contains various components in existing ABMs and supports the development of new ABM by reusing the components considering its model structure, which is based on the system entity structure/model base framework [58].

2) Relationships Between Model Reusability and Semantic Features in LDEF: It is possible to misunderstand that the model reusability from formalisms are determined by their syntactic features. The homomorphic map (see Fig. 5) illustrates that there are many syntactically isomorphic formalisms. Do these isomorphic formalisms provide the same amount of



$$\begin{aligned}
&Amb\_teatment\ action = < X, Y, S_{aw}, S_{cond}, S_{act}, P, D, A, ta >\\
&X = \{triageInfo\}\\
&Y = \{treatment\}\\
&S_{aw} = \{PatientStatus\}\\
&S_{cond} = \{BasicLifeSupport\}\\
&S_{act} = \{SupportLife, Wait\}\\
&P: triageInfo \times s \longrightarrow PatientStatus \times s\\
&D: PatientStatus \times BasicLifeSupport \times s\\
&\quad \longrightarrow SupportLife \times s\\
&A: SupportLife \longrightarrow treatment\\
&ta: SupportLife \longrightarrow T_{treat}
\end{aligned}$$

$$\begin{aligned}
&Hos\_teatment\ action = < X, Y, S_{aw}, S_{cond}, S_{act}, P, D, A, ta >\\
&\boxed{X = \{\boldsymbol{triageInfo}\}}\\
&Y = \{\boldsymbol{treatment}, operation, medicine\}\\
&\boxed{S_{aw} = \{\boldsymbol{PatientStatus}\}}\\
&S_{cond} = \{\boldsymbol{BasicLifeSupport}, OR, Medicine\}\\
&S_{act} = \{\boldsymbol{SupportLife}, Operation, Prescription, Wait\}\\
&\boxed{P: \boldsymbol{triageInfo \times s \longrightarrow PatientStatus \times s}}\\
&D: \boldsymbol{PatientStatus \times BasicLifeSupport \times s}\\
&\quad \boldsymbol{\longrightarrow SupportLife \times s}\\
&\quad PatientStatus \times OR \times s\\
&\quad \longrightarrow Operation \times s\\
&\quad PatientStatus \times Medicnes \times s\\
&\quad \longrightarrow Prescription \times s\\
&A: \boldsymbol{SupportLife \longrightarrow treatment}\\
&\quad Operation \longrightarrow treatment\\
&\quad Prescription \longrightarrow treatment\\
&ta: \boldsymbol{SupportLife \longrightarrow T_{treat}}\\
&\quad Operation \longrightarrow T_{treat}\\
&\quad Prescription \longrightarrow T_{treat}
\end{aligned}$$

Fully reused components of perception behavior using the LDEF semantics

Fig. 16. Fully reused elements for the perception behavior of the Amb_treatment action in the development of the Hos_treatment action, which is facilitated by the ABM-oriented semantics in LDEF.

model reusability? They probably do not because their semantic features also affect the model reusability from the different points.

Model reusability is considered at various levels: a small portion of code, a component, and a complete model [11]. Then, how do we determine the level of the model reusability? Semantics in formalisms would be one of the answers. The semantics of LDEF formalism, for instance, represent the ABM contexts: The action model in LDEF explicitly represents the procedure of agent behaviors (e.g., perception, decision, and action) in its formal specifications (i.e., three types of states—S_{aw}, S_{cond}, and S_{act}) and the associated functions (P, D, and A). These explicit representations increase the scope of reusable components in the ABM development.

The AABM development presents an exact example of increasing model reusability by the LDEF semantics. Fig. 16 illustrates two kinds of treatment action models in AABM: one is activated by ambulance agents and the other is performed in hospital agents. These action models equivalently perceive patient information (triageInfo), but their decisions and actions might be different because hospital agents contain more advanced medical equipment and human resources. Based on this notion, the treatment action model in hospital agents could reuse the perception components (i.e., X, S_{aw}, and P) of the other treatment action model and vice versa. Moreover, such reuses are fully supported by the LDEF semantics.

What if these models are developed by other isomorphic formalisms such as DS DEVS or Mobile DEVS? Because their syntactical properties are equivalent to the LDEF, they can describe AABM just as LDEF can. However, their semantics are based on systems engineering terms, and this dissimilarity restricts the scope of model reusability. Specifically, these formalisms would describe all behaviors of treatment action models as a mixture of their states and state transition functions, and such blended representations would confine the reuses of a portion of the behaviors.

It should be noted that such limitations of the isomorphic formalisms do not indicate that they are inferior to LDEF. Their difference is caused by the specificity of LDEF formalism in ABM developments. Semantics of those isomorphic formalisms focus on the high-level abstraction of the general systems so that they would provide more model reusability in the development of general systems. However, in the ABM development, LDEF offers more possibility of model reuse because its semantics deals with the ABM contexts.

B. Limitations of LDEF Formalism

This paper argues that LDEF is efficient in the ABM development. Nonetheless, from the various points, an LDEF-based approach holds several limitations as well. This section introduces several limitations for the further improvement.

LDEF formalism improves model reusability in the ABM development, and this improved model reusability is definitively efficient in the view of the model scalability. However, the efficiency in the model development should be considered as part of a big picture including system requirements, objectives, and costs. In this sense, the concept of value-driven systems engineering would need to be considered.

Value-driven systems engineering is an engineering strategy using microeconomics to transform systems engineering for multidisciplinary design optimization [59], [60]. Hence, the value-driven systems engineering would be applied to evaluate a system, so it helps to measure efficiency of its model development. Moreover, the concept of value-driven systems engineering would be utilized as the important criteria for efficient model compositions in the ABM framework discussed in the above section.

The DSM in LDEF seems to be similar to the coupled model of Mobile DEVS [20] from the perspective that both change coupling structure by defining states in the coupled model. However, there are different points between them: for example, the DSM does not separate particular events for triggering structural changes (which is called structure-change event in the Mobile DEVS). Instead, the structure changes of DSM are caused by internal output events, which reduces event routines for structural changes. However, this advantage strongly depends on the structure of applications.

LDEF defines that structure changes are dependent only on internal model outputs, and this mechanism is based on the following assumption: an event is forwarded through the current coupling structures, and if coupling structures are changed by an event, the new coupling structures are considered at the next event. This LDEF assumption is appropriate to represent the ABM contexts. Structural changes are related to embody self-organization and emergences in ABMs, and behavioral components would lead such collective behaviors. Therefore, the structural changes should be originated from the behavioral components, in particular, their output events. However, we note that this assumption is not always reasonable, depending on application cases.

VI. CONCLUSION

ABM is widely applied to various domains, but its development has been conducted without reusing the extant models. Meanwhile, systems engineers have facilitated model reuses by the virtue of formal methods that support modular and hierarchical modeling. Hence, this paper proposed LDEF formalism to promote model reuses in the ABM development by its modular and hierarchical manner and to explicitly represent the ABM contexts in its model specifications at the same time. In particular, this paper presents that model reusability supported by LDEF helps to promote large-scale and flexible ABM modeling.

To support the modular and hierarchical modeling, LDEF distinguishes the behavioral and structural parts of the ABM and develops them as individual components. Also, to reflect the ABM contexts, LDEF embodies the action procedure of an agent and the dynamic interaction changes in the model specifications.

This paper presents the theoretical basis and practical efficiency of LDEF. To show the theoretical basis, LDEF is compared with other formalisms using the concept of system morphism, which shows the behavioral equivalence between LDEF and other DES formalisms. The example models illustrate the practical efficiency of LDEF from the perspectives of model reusability. In particular, the example models show that the model reusability in the ABM development is improved by the virtue of the syntactic and semantic features in LDEF. We expect that LDEF would encourage model reuses in ABM developments to reduce the costs of future developments.

REFERENCES

[1] Y. Jiang and J. Jiang, “Diffusion in social networks: A multiagent perspective,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 198–213, Feb. 2015.

[2] S. M. Daboor, “Application of bacterial biomass as a potential heavy metal bio-removal agent,” *Afr. J. Microbiol. Res.*, vol. 8, no. 22, pp. 2229–2237, May 2014.

[3] I. G. Unda *et al.*, “Management of electric vehicle battery charging in distribution networks with multi-agent systems,” *Electr. Power Syst. Res.*, vol. 110, pp. 172–179, May 2014.

[4] D. Taghawi-Nejad, “Modelling the economy as an agent-based process: ABCE, a modelling platform and formal language for ACE,” *J. Artif. Soc. Soc. Simulat.*, vol. 16, no. 3, p. 1, Jun. 2013.

[5] R. Hayes and R. Hayes, “Agent-based simulation of mass shootings: Determining how to limit the scale of a tragedy,” *J. Artif. Soc. Soc. Simulat.*, vol. 17, no. 2, p. 5, Mar. 2014.

[6] Y. Xie and S. Fan, “Multi-city sustainable regional urban growth simulation—MSRUGS: A case study along the mid-section of Silk Road of China,” *Stoch. Environ. Res. Risk Assess.*, vol. 28, no. 4, pp. 829–841, May 2014.

[7] R. Das and S. Hanaoka, “An agent-based model for resource allocation during relief distribution,” *J. Humanit. Logist. Supply Chain Manage.*, vol. 4, no. 2, pp. 265–285, 2014.

[8] B. Heath, R. Hill, and F. Ciarallo, “A survey of agent-based modeling practices (January 1998 to July 2008),” *J. Artif. Soc. Soc. Simulat.*, vol. 12, no. 4, p. 9, Oct. 2009.

[9] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2009.

[10] C. Sung and T. G. Kim, “Collaborative modeling process for development of domain-specific discrete event simulation systems,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 532–546, Jul. 2012.

[11] S. Robinson, R. E. Nance, R. J. Paul, M. Pidd, and S. J. Taylor, “Simulation model reuse: Definitions, benefits and obstacles,” *Simulat. Model. Pract. Theory*, vol. 12, nos. 7–8, pp. 479–494, Nov. 2004.

[12] E. T. Saulnier and B. J. Bortscheller, “Simulation model reusability,” *IEEE Commun. Mag.*, vol. 32, no. 3, pp. 64–69, Mar. 1994.

[13] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation*, 2nd ed. San Diego, CA, USA: Academic Press, 2000.

[14] J. F. Monin and M. G. Hinchey, *Understanding Formal Methods*. London, U.K.: Springer, 2003.

[15] W. C. Chu and H. Yang, “A formal method to software integration in reuse,” in *Proc. 20th Conf. Comput. Softw. Appl.*, Seoul, Korea, 1996, p. 343–348.

[16] H. Yamada and S. Amoroso, “Structural and behavioral equivalences of tessellation automata,” *Inf. Control*, vol. 18, no. 1, pp. 1–31, Feb. 1971.

[17] H. S. Sarjoughian, B. P. Zeigler, and S. B. Hall, “A layered modeling and simulation architecture for agent-based system development,” *Proc. IEEE*, vol. 89, no. 2, pp. 201–213, Feb. 2001.

[18] M. Akplogan, G. Quesnel, A. Joannon, and R. Martin-Clouaire, “Towards a deliberative agent system based on DEVS formalism for application in agriculture,” in *Proc. Summer Comput. Simulat. Conf. (SCSC)*, Ottawa, ON, Canada, 2010, pp. 250–257.

[19] B. Zhang, W. K. Chan, and S. V. Ukkusuri, “Agent-based discrete-event hybrid space modeling approach for transportation evacuation simulation,” in *Proc. Winter Simulat. Conf. (WSC)*, Phoenix, AZ, USA, 2011, pp. 199–209.

[20] J.-H. Kim and T. G. Kim, “DEVS-based framework for modeling/simulation of mobile agent systems,” *Simulation*, vol. 76, no. 6, pp. 345–357, Jun. 2001.

[21] A. M. Uhrmacher, “Dynamic structures in modeling and simulation: A reflective approach,” *ACM Trans. Model. Comput. Simu.*, vol. 11, no. 2, pp. 206–232, Apr. 2001.

[22] M. Zhang, “Constructing a cognitive agent model using DEVS framework for multi-agent simulation,” in *Proc. 15th Eur. Agent Syst. Summer School (EASSS)*, London, U.K., 2013, pp. 1–5.

[23] R. Duboz, D. Versmisse, G. Quesnel, A. Muzy, and E. Ramat, “Specification of dynamic structure discrete event multiagent systems,” *Simulat. Series*, vol. 38, no. 2, pp. 103–114, 2006.

[24] F. J. Barros, “The dynamic structure discrete event system specification formalism,” *Trans. Soc. Comput. Simulat. Int.*, vol. 13, no. 1, pp. 35–46, Mar. 1996.

[25] A. Troccoli and G. Wainer, “Implementing parallel cell-DEVS,” in *Proc. 36th Annu. Symp. Simulat.*, Orlando, FL, USA, 2003, pp. 273–280.

[26] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artif. Intell.*, vol. 101, nos. 1–2, pp. 99–134, May 1998.

[27] M. Bratman, *Intention, Plans, and Practical Reason*, 2nd ed. Cambridge, MA, USA: Harvard Univ. Press, 1987.

[28] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton, NJ, USA: Princeton Univ. Press, 2007.

[29] T. Homer-Dixon *et al.*, “A complex systems approach to the study of ideology: Cognitive-affective structures and the dynamics of belief systems,” *J. Soc. Polit. Psychol.*, vol. 1, no. 1, pp. 337–363, Dec. 2013.

[30] Y. Bar-Yam, *Dynamics of Complex Systems*. Reading, MA, USA: Addison-Wesley, 1997.

[31] A. Ricci, M. Viroli, and M. Piunti, "Formalising the environment in MAS programming: A formal model for artifact-based environments," in *Proc. 7th Int. Workshop Program. Multi-Agent Syst. (ProMAS)*, Budapest, Hungary, 2009, pp. 133–150.

[32] A. Omicini, A. Ricci, and M. Viroli, "Formal specification and enactment of security policies through agent coordination contexts," *Electron. Notes Theor. Comput. Sci.*, vol. 85, no. 3, pp. 17–36, Aug. 2003.

[33] A. Omicini, A. Ricci, and M. Viroli, "Agent coordination contexts for the formal specification and enactment of coordination and security policies," *Sci. Comput. Program.*, vol. 63, no. 1, pp. 88–107, Nov. 2006.

[34] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the A&A meta-model for multi-agent systems," *Auton. Agents Multi-Agent Syst.*, vol. 17, no. 3, pp. 432–456, Dec. 2008.

[35] A. Ricci, E. Denti, and M. Piunti, "A platform for developing SOA/WS applications as open and heterogeneous multi-agent systems," *Multiagent Grid Syst.*, vol. 6, no. 2, pp. 105–132, Apr. 2010.

[36] M. Luck and M. D'Inverno, "A formal framework for agency and autonomy," in *Proc. Int. Conf. Multiagent Syst.*, Menlo Park, CA, USA, 1995, pp. 254–260.

[37] D. Silva and L. Paulino, "A formal model for the fifth discipline," *J. Artif. Soc. Soc. Simulat.*, vol. 8, no. 3, p. 6, Jun. 2005.

[38] M. A. Niazi and A. Hussain, "A novel agent-based simulation framework for sensing in complex adaptive environments," *IEEE Sensors J.*, vol. 11, no. 2, pp. 404–412, Feb. 2011.

[39] M. Fisher and M. Wooldridge, "On the formal specification and verification of multi-agent systems," *Int. J. Cooperative Inf. Syst.*, vol. 6, no. 1, pp. 37–66, 1997.

[40] S. Conrad, G. Saake, and C. Türker, "Towards an agent-oriented framework for specification of information systems," in *Formal Models of Agents*. Berlin, Germany: Springer, 1999, pp. 57–73.

[41] H. Zhu, "SLABS: A formal specification language for agent-based systems," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 11, no. 5, pp. 529–558, Oct. 2001.

[42] F. M. T. Brazier, B. M. Dunin-Keplicz, N. R. Jennings, and J. Treur, "DESIRE: Modelling multi-agent systems in a compositional formal framework," *Int. J. Cooperative Inf. Syst.*, vol. 6, no. 1, pp. 67–94, 1997.

[43] I. A. Lomazova, "Nested Petri nets—A formalism for specification and verification of multi-agent distributed systems," *Fundamenta informaticae*, vol. 43, nos. 1–4, pp. 195–214, Jan. 2000.

[44] B. Bauer, J. P. Müller, and J. Odell, "Agent UML: A formalism for specifying multiagent software systems," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 11, no. 3, pp. 207–230, Jun. 2001.

[45] J. Bentahar, B. Moulin, and B. Chaib-draa, "Commitment and argument network: A new formalism for agent communication," in *Advances in Agent Communication*. Berlin, Germany: Springer, 2004, pp. 146–165.

[46] D. C. Chatfield, J. C. Hayya, and T. P. Harrison, "A multi-formalism architecture for agent-based, order-centric supply chain simulation," *Simulat. Model. Pract. Theory*, vol. 15, no. 2, pp. 153–174, Feb. 2007.

[47] B. Marzougui, K. Hassine, and K. Barkaoui, "A new formalism for modeling a multi agent systems: Agent Petri nets," *J. Softw. Eng. Appl.*, vol. 3, no. 12, pp. 1118–1124, Dec. 2010.

[48] L. Wu, D. Wu, J. Chen, and W. Li, "An agent formal model for autonomous decision-making," in *Proc. 4th Int. Conf. Multimedia Inf. Netw. Security (MINES)*, Nanjing, China, 2012, pp. 943–946.

[49] D. A. Norman, "Cognitive engineering," in *Human Cognitive Abilities: A Survey of Factor-Analytic Studies*. New York, NY, USA: Cambridge Univ. Press, 1993, ch. 3, p. 89.

[50] P. Maes, "Artificial life meets entertainment: Lifelike autonomous agents," *Commun. ACM*, vol. 38, no. 11, pp. 108–114, Nov. 1995.

[51] B. Hayes-Roth, "An architecture for adaptive intelligent systems," *Artif. Intell.*, vol. 72, nos. 1–2, pp. 329–365, Jan. 1995.

[52] V. Grimm and S. F. Railsback, "A conceptual framework for designing individual-based models," in *Individual-Based Modeling and Ecology*. Princeton, NJ, USA: Princeton Univ. Press, 2013, ch. 5, pp. 68–113.

[53] C. Luh and B. P. Zeigler, "Abstracting event-based control models for high autonomy systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 23, no. 1, pp. 42–54, Jan./Feb. 1993.

[54] J. Shi, "A conceptual activity cycle-based simulation modeling method," in *Proc. 29th Conf. Winter Simulat.*, Huntington Beach, CA, USA, 1997, pp. 1127–1133.

[55] J. W. Bae, S. Lee, J. H. Kim, and I.-C. Moon, "Simulation-based analyses on massive evacuation from metropolis during bombardment," *Simulat., Trans. Soc. Model. Simulat. Int.*, vol. 90, no. 11, pp. 1244–1267, Nov. 2014.

[56] S. Lee, J. W. Bae, J. H. Hong, and I.-C. Moon, "Simulation experiment of routing strategy for evacuees and disaster responders," in *Proc. Spring Simulat. Multi-Conf.*, Pasadena, CA, USA, 2014, Art. ID 2.

[57] Y. Choi and T. G. Kim, "Reusability measure of DEVS simulation models in DEVSim++ environment," in *Proc. AeroSense*, Orlando, FL, USA, 1997, pp. 244–255.

[58] T. G. Kim, C. Lee, E. R. Christensen, and B. P. Zeigler, "System entity structuring and model base management," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 20, no. 5, pp. 1013–1024, Sep./Oct. 1990.

[59] P. D. Collopy and P. M. Hollingsworth, "Value-driven design," *J. Aircraft*, vol. 48, no. 3, pp. 749–759, May 2011.

[60] J. Cheung *et al.*, "Application of value-driven design to commercial aeroengine systems," *J. Aircraft*, vol. 49, no. 3, pp. 688–702, May/Jun. 2012.

Jang Won Bae received the M.S.E. degree from the School of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2009, where he is currently pursuing the Ph.D. degree with the Department of Industrial and Systems Engineering.

His current research interests include discrete event system modeling and simulation, agent-based modeling and simulation, interoperation of simulations and complex adaptive systems analyses.

Il-Chul Moon received the Ph.D. degree from the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, in 2008.

He is currently an Assistant Professor with the Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea. His research interests include the overlapping area of computer science, management, sociology, and operations research, and also military command and control analysis, counterterrorism analysis, intelligence analysis, and disaster management.