Information Sciences 292 (2015) 75–94

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Extended self-reproducible Discrete Event System Specification (DEVS) formalism using hidden inheritance

Sangjoon Park [a], Seong-Moo Yoo [b,*]

[a] Department of Computer Information Engineering, Kunsan National University, Kunsan Gunsan-si, JeonBuk 573-701, Republic of Korea
[b] Electrical and Computer Engineering Department, The University of Alabama in Huntsville, Huntsville, AL 35899, USA

ARTICLE INFO

Article history:
Received 14 November 2013
Received in revised form 5 August 2014
Accepted 18 August 2014
Available online 16 September 2014

Keywords:
Discrete event system specification
Modeling
Formalism
Inheritance
Self-reproducible

ABSTRACT

In system modeling and simulation, a reproducible parent object can transfer its properties to a child object, allowing the inherited child to represent the characteristics of its parent. In this paper, we propose the use of an extended Self-Reproducible Discrete Event System Specifications (SR-DEVS) modeling formalism, which is characterized by a coupled (equivalent) parent–child hereditary relationship. Yet, contrary to other existing schemes, we consider the *hidden* mechanism of the inheritance process. When modeling a self-growing overall system in which a component object reproduces an offspring object, the hereditary formalism elaboration from one generation to the next normally displays the system architecture and modeling characteristics. Thus, a parent component object passes its structural properties to its child components so that the overall growing system expresses a continuous self-identity and similarity. However, in the presence of a hidden inheritance mechanism, an inherited asset may be concealed or its proper function may not be outwardly clear, even after its inheritance by the child object. In this work, we investigate a case study that applies the proposed model to the simulation of a social evolution model, using social behaviors related to online shopping.

1. Introduction

The overall framework entities of a system simulation consist of a real-world source system, system model and simulator. The system model derived from the real source system has specifications to express the modeled architecture and system behavior [7,15]. The model's structure and behavior can be clearly constructed using rich system data; therefore, system modeling is an important phase in determining how well the extracted system model reflects the properties of the real source system. After system modeling is complete, the system simulator, which is constructed based on model specifications, implements the simulation and can provide analyzed simulation results. In filling the system agent role, the simulator not only obeys the model specifications but also reveals the simulation behavior. Through computer technique developments, computer simulations are being widely used to analyze more complex source systems and to predict difficult and changeable real systems. However, simulation modeling needs a regular style to be effective, which supports the use of a formal, specified model of rules when constructing a valid simulation model, as opposed to using a more irregular approach. Hence, the formalism of simulation modeling provides for the design of a rule model that formally sets the model's specifications.

* Corresponding author.
E-mail addresses: lubimia@kunsan.ac.kr (S. Park), yoos@uah.edu (S.-M. Yoo).

http://dx.doi.org/10.1016/j.ins.2014.08.069
0020-0255/

To attain formalism in simulation modeling and computer simulation, Discrete Event System Specification (DEVS) was introduced in [15] as a well defined modeling formalism and is now continuously extended to describe applied complex system models. Using this methodology, DEVS formalism can maintain the validity of simulation modeling. *Atomic DEVS* and *coupled DEVS* were introduced as the classical forms of DEVS formalism but were succeeded by DEVS models that represented more complex system models [2–4,8,10,11,14,16,17]. *Parallel DEVS* was also introduced to handle simultaneously scheduled events and resolve transition collisions [4,5,13,14]. In classical DEVS formalism, through the *select* function, the model attempts to process the collision behavior between internal and external events. However, the *select* function is limited to adjusting two or more transition events, so the *confluent* transition function in *parallel DEVS* is employed to control the collision behavior. *Dynamic structure DEVS* allows for the possibility of dynamic structural change within the model [1,2], such as the addition or deletion of a model component or the alteration of a component connection. *Real time DEVS* (RT-DEVS) is extended DEVS formalism in which the real time advance function replaces the virtual time advance function for the simulation's time activity [6,8,18]. In RT-DEVS, a driver model function is introduced to offer an interface between the simulation model and real source objects [6]. Finally, the *cell-DEVS* model is used to combine Cellular Automata with DEVS formalism [17].

The DEVS-based applications as well as formalism extensions have been introduced in various simulations [12,19–23]. Sung and Kim [12] have proposed a collaborative modeling method to develop effective modeling and simulation software, using a war game DEVS simulator to exemplify and support their approach. Huang et al. [19] have proposed a railway simulation library (LIBROS-II) using the DEVS formalism to increase the performance of rail simulation. Further, the authors in [20,21] have incorporated the applications of DEVS into wireless network simulations while Wang et al. [22] have proposed a simulation model of emergency evacuation that is supported by Building Information Modeling (BIM) of Computer-aided Design (CAD) software. It aims to produce useful emergency plans and rapid evacuation times through evacuation modeling and simulation with a *Cell-DEVS* model. Additionally, Olamide and Kaba [23] have presented a model-based verification and validation technique to ensure the accuracy and correctness of simulation results. They have also provided a case study example of GSM telecommunication systems to highlight the capabilities of the verification framework.

Self-Reproducible DEVS (SR-DEVS) is also proposed to ensure the fitness of complex modeling and requires a more adequate modeling formalism for the reproducible system structure [9]. In reproducible systems, self-reproduction can be represented in a structural system with an inheritance function between the parent and child components. This inheritance function dictates that a child component receives a property from its parent component while reproducing. Here, we can consider the elaboration of the inheritance function when a child component takes its parent's characteristic. In the inheritance process, the hidden property of the inherited asset in a child component can be evaluated for various genetic characteristics.

Thus, in this paper, we propose the use of extended SR-DEVS formalism using hidden inheritance properties (contrary to existing schemes) to show that modeling formalism can represent more complicated inheritance mechanisms. Also, while the system structure is partially changeable through the process of succession, the structural properties can be continuously maintained with inherited assets. As a result, the inheritance function can support the organizational continuity found in social functional systems, evolutionary or biological structural systems and other technical computing system simulations.

In this paper, we also investigate a case study that applies our proposed model to a social evolution model based on online social networks. The social evolution model is used to find the social change model based on biological evolution [26], with various researches being provided to describe the social change mechanism [27,30]. Based on the proposed model, we attempt to find the technical analysis model for social evolution. Also, we examine online social commerce and the verification methodology of evolution variation in consideration of the social model.

This paper is organized as follows. In Section 2, we provide an overview of the existing DEVS model formalism. In Section 3, we propose an extended SR-DEVS formalism and examine the proposed model to a social evolution model. Finally, we conclude this paper in Section 4.

2. Devs formalism

2.1. Classic DEVS formalism

DEVS formalism [9,16] is a representative modeling methodology used to describe various computing models and to analyze the validity of simulations and modeling. Initially, atomic DEVS and coupled DEVS were introduced as forms of classical DEVS formalism. Atomic DEVS defines the specifications of dynamic system behavior for single-system modeling. It has a single input and output port by which event values come in and out of a component model on the time segment. An atomic DEVS model, M, is defined by the following structure:

$$M = (X, Y, S, \delta_{\text{int}}, \delta_{ext}, \lambda, ta)$$

where

X is a set of input events,
Y is a set of output events,
S is a set of states,

δ_{int}: $S \to S$ is the internal transition function,
δ_{ext}: $Z \times X \to S$ is the external transition function,
where $Z = \{(s,e)|s \in S,\ 0 \leqslant e \leqslant ta(s)\}$ is the set of total states of the model and
e is the time elapsed since last transition,
$\lambda : S \to Y$ is the output function,
$ta : S \to R^{+}_{0,\infty}$ is the time advance function.

The basic system model is affected by event X that is given by modeling environments. When the basic system is in a state s without an external event, it will stay in the state s within the time $ta(s)$. If the elapsed time becomes $e = ta(s)$, the system outputs the value by $\lambda(s)$, and changes to a state $\delta_{\text{int}}(s)$. Hence, if no external event occurs before the time expiration of state s, the system generates an output event by the output function, and then immediately its state is shifted to next state by the internal transition function. If an external event $x \in X$ arrives at elapsed time $e \leqslant ta(s)$, and the system is in a state (s, e), the system changes to a state $\delta_{ext}(s, e, x)$ and new $ta(s)$ is determined. That is, if an external event arrives before the time exhaustion to next internal transition, the external transition function leads the next state of system which is determined by the current state, input event and the elapsed time. Here, the time base t is a real number. Also, the output function Λ of the dynamic system [9,15] is given by

$$\Lambda(s,e) = \begin{cases} \lambda(s) & \text{if } e = ta(s) \text{ and } \omega(t) = \varnothing \\ \lambda(s) \text{ or } \varnothing & \text{if } e = ta(s) \text{ and } \omega(t) \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases}$$

where $\omega(t)$ is the time segment.

A single system can be a component model in a composed structural system that includes a component group. A component model in a composed system has input and output ports to communicate with other component models.

Coupled DEVS is also a classic form of DEVS that it is composed of atomic systems. A coupled model can be joined as a component system to construct a more hierarchically complicated structural system. Also, coupled models can be employed to make a distributed modular system. For inside components, a coupled model has three interconnections (couplings): an external input coupling, an internal coupling and an external output coupling. Fig. 1 shows an example of these three couplings. The external input coupling is the interconnection between the external coupled model and the input of the inside component. The internal coupling provides the interconnection for the output of a component and the input of another component while the external output coupling connects the output of the component and the output of the coupled model.

The coupled DEVS, DM, including sub components, is defined as:

$$DM = (X, Y, D, \{cM_d | d \in D\}, EIC, EOC, IC, SELECT)$$

where
$X = \{(p, v)|p \in IPorts, v \in X_p\}$ is a set of input ports p and values v,
$Y = \{(p, v)|p \in OPorts, v \in Y_p\}$ is a set of output ports p and values v,
D is a set of DEVS component names,
$cM_d = (X_d, Y_d, S, \delta_{\text{int}}, \delta_{ext}, \lambda, ta)$ is a component DEVS model with
$X_d = \{(p, v)|p \in IPorts_d, v \in X_p\}$ and $Y_d = \{(p, v)|p \in OPorts_d, v \in Y_p\}$,
$EIC \subseteq \{((DM, inp_M), (d, inp_d))|inp_M \in IPorts, inp_d \in IPorts_d\}$ is the external input coupling to external inputs and component inputs,
$EOC \subseteq \{((d, oup_d), (DM, oup_M))|oup_M \in OPorts, oup_d \in OPorts_d\}$ is the external output coupling to component outputs and external outputs,

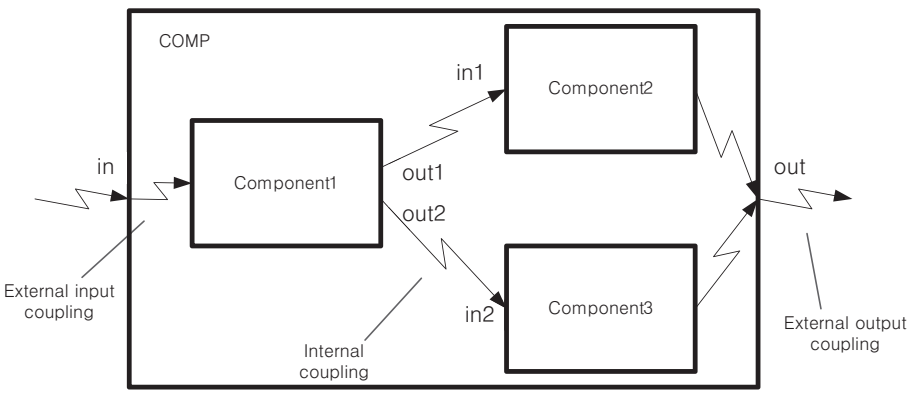


Fig. 1. Couplings of coupled *DEVS* system model.

$IC \subseteq \{((f, oup_f), (s, inp_s)) | f \in D, oup_f \in OPorts_d, s \in D, inp_s \in IPorts_d\}$ is the internal coupling to component outputs and inputs, and
SELECT: $2^D - \varnothing \to D$ is the selection function.

Example 1. Refer to Fig. 1. The input of COMP, a whole system, is connected into the input of component1 so that the external input coupling is $EIC = \{(COMP, in), (component1, in)\}$. The internal coupling to component1, component2 and component3 is $IC = \{((component1, out1), (component2, in1)), ((component1, out2), (component3, in2))\}$. The output of COMP is linked into the outputs of component2 and component3 that the external output coupling is $EOC = \{((component2, out), (COMP, out)), ((component3, out), (COMP, out))\}$.

2.2. SR-DEVS model

SR-DEVS model [9] provides a mechanism that a single DEVS parent reproduces child components showing structural similarity with the same or similar behavior function. A parent DEVS component once reproduces a child DEVS component, and both become a cluster as family. A parent model, P_Ψ, has a structure as follows;

$$P_\Psi = (X, Y, S, C, \delta_{\text{int}}, \delta_{ext}, \lambda, ta)$$

where Ψ is a name of a parent model.

$$C = (P_\psi, \{R_\psi | R_\psi \in^* P_\psi\}, INH, CON)$$

where
C is a *connector* of P_Ψ as the reproduction relationship to parent and child components,
R_Ψ is a set of child components ($R_\Psi = \{r_1, r_2, \ldots, r_n | 1 \leqslant n \leqslant \infty\}$),
ϵ^* means the inheritance relation that a child *DEVS* component is derived from a parent *DEVS* model,
INH is a set of inheritance function from a parent model to child components,
$CON = \{((P_\psi, out_\psi)(R_{\psi'}, in)), ((R_{\psi'}, out), (P_\psi, in_\psi)) | out_\psi \text{ and } out \in OutCon, in_\psi \text{ and } in \in InCon\}$ *CON* is a connection set of parent and child models
inCon is a set of input connection, and *OutCon* is a set of output connection.

Hence, a child component model is obtained as:

$$r_\psi = (IP, OP, S_\psi, C_\psi, \delta_{\psi-\text{int}}, \delta_{\psi-ext}, \lambda_\psi, ta_\psi)$$

where
$IP = \{\alpha_1, \alpha_2, \ldots, \alpha_l\}$ is an input set,
$OP = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is an output set,
$S_\Psi = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ is a state set,
C_Ψ is a *connector*,
$\delta_{\Psi\text{-}int}(\epsilon\ ^*\delta_{\text{int}})$ is the internal transition function,
$\delta_{\Psi\text{-}ext}(\epsilon\ ^*\delta_{ext})$ is the external transition function,
$\lambda_{\psi'}(\in {}^*\lambda_\psi)$ is the output function.

Note that if a parent component model has no child component, it is the same as classic atomic DEVS model. Assume that a component model is given by $P_a = (X_a, Y_a, S, C_a, \delta_{\text{int}}, \delta_{ext}, \lambda, ta)$, and it has no child component ($R_a = \varnothing$). Then, INH_a and *CON* cannot be accomplished ($INH_a = \varnothing$, $CON_a = \varnothing$), and by $C_a = \varnothing\ P_a$ is a single DEVS model. However, if a parent component has at least one child component, both components have the inheritance relationship and child components show the same or similar system behaviors to their parent model.

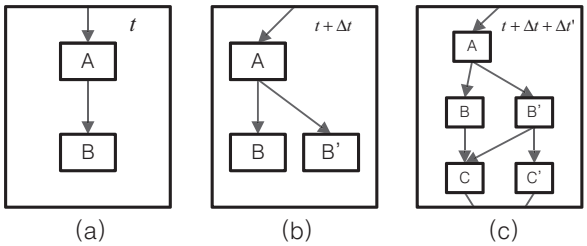


Fig. 2. A system reproduction.

2.3. More explanations of SR-DEVS modeling

In this subsection, we explain SR-DEVS modeling in greater detail, using Figs. 2 and 3, before proposing the extended model.

In simulation modeling environments, we may need a model with the same or similar components (or coupled model) to construct a distributed or hierarchical structure model. In such a functional structure, a member component can reproduce a child component so that a reproduction relationship is established. The reproduction relationship among components permits similarities in component operation, in that a child DEVS component can show the same or similar functional configuration as its parent DEVS component. We call this reproduction relationship the behavior inheritance. Through inherited behavior, a child DEVS component receives behavior from its parent DEVS component. Furthermore, structural inheritance as well as behavior inheritance is possible in the reproduction of a coupled model. The system architecture may change the

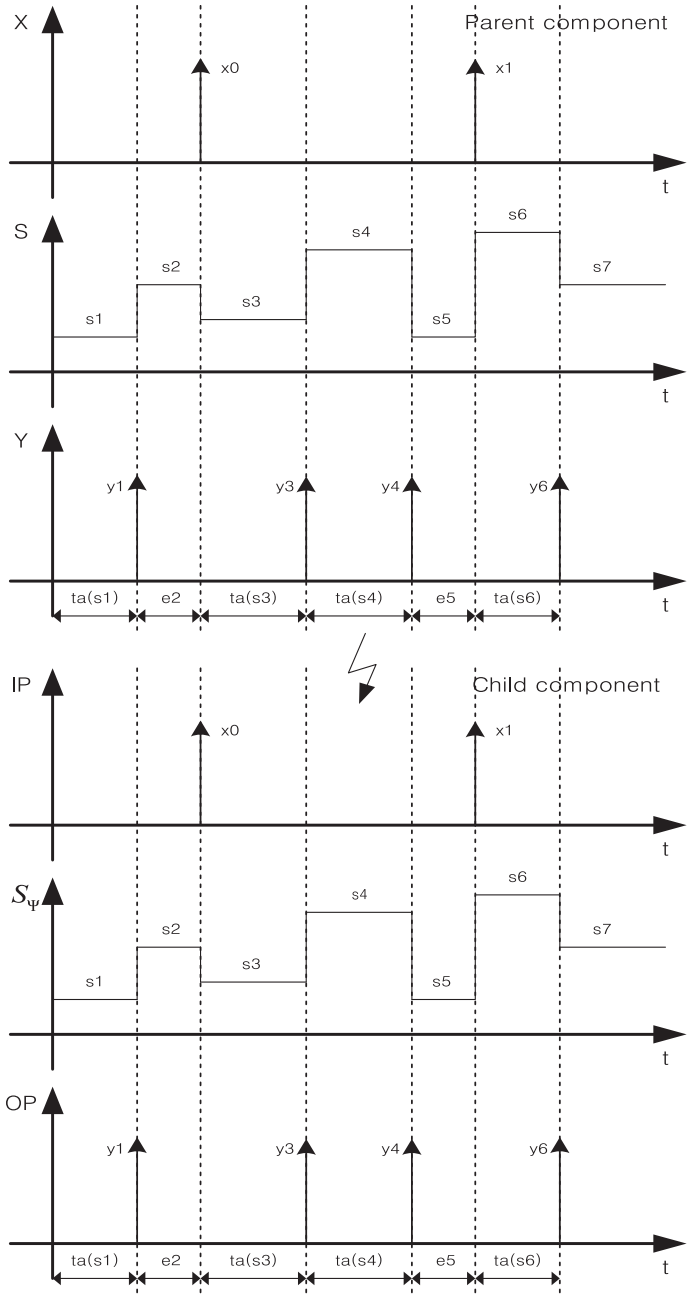


Fig. 3. Inherited behavior of *SR-DEVS* mode.

time flow of the simulation as a coupled component receives structural properties from a parent coupled component during the reproduction process.

Example 2. Fig. 2 shows a system reproduction process. Component A reproduces component B at time t. From the component A, the component B receives not only the set of inherited behavior but the structural configuration. In time $t + \Delta t$, component A reproduces a new component B'. It is possible that the child components B and B' may resemble or be the same each other. Otherwise, the component B' can show entirely different behavior configuration to component B. In Fig. 2(c), components B and B' reproduce their child components C and C' during $\Delta t'$. The child component C and C' may also resemble or not. Hence, we need an inheritance mechanism for self-reproducible system which has the property of structural growth or evolution system.

Example 3. The inherited behavior sequences of a child component model are illustrated in Fig. 3. The upper (bottom) figure shows system behavior of a parent (child) model's transition sequences. Here, $\delta_{\text{int}}(s1) \rightarrow s2$ is the internal state transition without an external event. When external input event $x0$ is arrived at the end of elapsed time $e2$ and current state $s2$, next state is determined as $\delta_{ext}(s2, e2, x0) \rightarrow s3$. After a child component is reproduced, it shows the same transition behavior as that of parent component if it receives all functions from the parent model. Like the parent component, when no external event occurs in state $s4$, the internal state transition of the child component is $\delta_{\Psi\text{-int}}(s4) \rightarrow s5$. With external input $x1$, elapsed time $e5$ and current state $s5$, the new state is represented to $\delta_{ext}(s5, e5, x1) \rightarrow s6$.

Note that the inheritance pattern has two categories: integrity and fragment. By integrity a child component receives all inheritance assets of parent model so that the child component shows the same system behavior. By fragment, a child model can only represent partial behavior of parent model. It is possible that child components are more or less different each other by the inheritance variation. In SR-DEVS model a child DEVS component model can have plural parent DEVS models to compose various inheritance behaviors. When a child DEVS component has some parent DEVS models, its reproduction relationship (connector) is represented as follows:

$$C_P = (P, \{R_\Psi | R_\Psi \in {}^*P\}, INH, CON)$$

where P is a set of parent models $P = \{p_{\Psi 1}, p_{\Psi 2}, p_{\Psi 3}, \cdots, p_{\Psi n}\}$.

Also, a *coupled SR-DEVS* model, CP_m, is given by

$$CP_m = (X_m, Y_m, D, \{cM_d | d \in D\}, C_{cp}, EIC, EOC, IC)$$

where

D is a set of sub-components,

C_{cp} is the connector of coupled model.

Here, a coupled SR-DEVS model can deliver the overall structured model to the child coupled model. Inside components are basically imparted to the child coupled model during the model's reproduction process. In particular, a child coupled model resembles not only the system behavior of its parent coupled model but also the structural properties that define the composed identity. In essence, a parent coupled component can include normal atomic components without the reproduction function. If a parent coupled component has inside components that have reproduction ability, a child coupled component can be reproduced while a parent component changes its structure.

The SR-DEVS model is a reproduction formalism methodology that shows inheritance characteristics (*integrity* and *fragment*) [9]. In the reproduction process, a more elaborate inheritance form can be requested for the functional configuration. In the next section, we propose an extended SR-DEVS model using hidden inheritance properties.

3. Extended SR-DEVS model

3.1. Hidden inheritance

An inheritance function allows the properties of a parent component to be passed on to a child component while preserving the generational characteristics on time flow. We introduce a hidden mechanism that is adopted in the reproduction process. When a child component receives an inheritance property from its parent component, the hidden mechanism is involved in the inheritance. Hence, after an inheritance asset of the parent is transferred to the child, the hidden mechanism is instantaneously implemented to determine the expression of the inheritance asset. Through this hidden inheritance, the property can be either represented or hidden by the system behavior. If a child DEVS component receives a normal inherited behavior, it expresses this property externally. However, if it obtains a hidden inheritance behavior, it does not outwardly reveal its hidden characteristic, although the inherited property has not disappeared. This is called hidden inheritance, as it permits a child DEVS component to receive an inherited behavior from its parent component, but it cannot show that behavior. Instead, the child component simply possesses the inherited property without expressing the property's behavior. For the hidden mechanism, the inheritance function of the SR-DEVS model is represented as follows:

$$INH^* = (r_\Psi, Ih, Oh, Sh, \delta_{h-\text{int}}, \delta_{h-ext}, \lambda h, hs, ta)$$

where

r_Ψ is a child DEVS component reproduced from its DEVS parent component,
Ih is a set of inheritance functions of input values $X = \{x_1, \ldots, x_m^*, \ldots, x_{m'}\}$,
ih_m^* is an inheritance function to $x_m^*(\in X)$ for the hidden inheritance,
Oh is a set of inheritance functions of output values $Y = \{y_1, \ldots, y_n^*, \ldots, y_{n'}\}$,
oh_n^* is an inheritance function to $y_n^*(\in Y)$ for the hidden property,
Sh is a set of inheritance function of state set which can include a hidden inheritance,
$\delta_{h\text{-}int}$ is a set of inheritance functions of internal transition which can include a hidden inheritance (δih_i^*),
$\delta_{h\text{-}ext}$ is a set of inheritance functions of external transition which can include a hidden inheritance (δxh_j^*),
λh is a set of inheritance function of output including a hidden inheritance, and hs is the hidden strength of inheritance property for the atomic SR-DEVS component.

The elements of Ih, Oh, $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of INH^* are listed in Table 1.

Here, a hidden mechanism has hidden strength. The hidden strength affects the determination of the hidden characteristic to the inheritance property. Also, an inheritance property can have the hidden mechanism. If an inherited property has no hidden characteristic, its hidden strength is $\varnothing$. Hence, if all inherited properties do not have any hidden mechanisms, the child receives all inherited properties normally, and it shows the received asset as it is. The hidden strength might be a constant value, a threshold value, or described as the strength function with a proper format. If the hidden strength is a Boolean value, the inheritance property has unconditionally hidden when the hidden strength is 1. If the hidden strength is expressed by the strength function, this function becomes the determination function to the hidden inheritance. The function provides a decision value for the hidden inheritance.

The inheritance properties can share certain hidden strength values as

$$hs = [w_1, \ldots, w_m, \ldots, w_{m'}],$$

or as

$$hs = \begin{bmatrix} f_1(v) & \cdots & f_k(v') & \cdots \\ \cdots & & \cdots & \\ & \cdots & n_m(q) & \cdots \\ \cdots & o_{n'}(w) & \cdots & o_{n''}(w') \end{bmatrix}.$$

After the inheritance to a property is fulfilled, immediately the hidden property is determined by the hidden strength. Note that the hidden inheritance is only applied to current child component, not other components. A positive property in a parent component can be a hidden property in a child component while inheriting the property. Otherwise, a hidden property of parent component remains as unchanged property in a child component. Also, a child component can receive several hidden properties from a parent component.

Table 1
Elements of hidden inheritance.

	Ih	Oh
INH^*	$ih_1, \ldots, ih_m^*, \ldots, ih_{m'}$	$oh_1, \ldots, oh_n^*, \ldots oh_{n'}$
$INH^{[*]}$	$ih_1, \ldots, [ih]_m^*, \ldots, ih_{m'}$	$oh_1, \ldots, [oh]_n^*, \ldots oh_{n'}$
$INH_c^{[*]}$	$ih_{s1'}, ih_{s2'}, [ih]_{s3'}^*, \ldots, ih_{m'}$	$oh_{s1'}, oh_{s2'}, [oh]_{s3'}^*, \ldots oh_{n'}$
INH_{nc}	$ih_{s1''}, ih_{s2''}, ih_{s3''}, \ldots, ih_{m''}$	$oh_{s1''}, oh_{s2''}, oh_{s3''}, \ldots oh_{n''}$
$INH^{(*)}$	$ih_1, \ldots, (ih)_m^*, \ldots, ih_{m'}$	$oh_1, \ldots, (oh)_n^*, \ldots oh_{n'}$
$INH_c^{(*)}$	$ih_{s1'}, ih_{s2'}, (ih)_{s3'}^*, \ldots, ih_{m'}$	$oh_{s1'}, oh_{s2'}, (oh)_{s3'}^*, \ldots oh_{n'}$
$INH_{nc}^{[*]}$	$ih_{s1''}, ih_{s2''}, [ih]_{s3''}^*, \ldots, ih_{m''}$	$oh_{s1''}, oh_{s2''}, [oh]_{s3''}^*, \ldots oh_{n''}$
	$\delta_{h\text{-int}}$	$\delta_{h\text{-}ext}$
INH^*	$\delta ih_1, \ldots, \delta ih_i^*, \ldots, \delta ih_{i'}$	$\delta xh_1, \ldots, \delta xh_j^*, \ldots, \delta xh_{j'}$
$INH^{[*]}$	$\delta ih_1, \ldots, [\delta ih]_i^*, \ldots, \delta ih_{i'}$	$\delta xh_1, \ldots, [\delta xh]_j^*, \ldots, \delta xh_{j'}$
$INH_c^{[*]}$	$\delta ih_{s1'}, \delta ih_{s2'}, [\delta ih]_{s3'}^*, \ldots, \delta ih_{i'}$	$\delta xh_{s1'}, \delta xh_{s2'}, [\delta xh]_{s3'}^*, \ldots, \delta xh_{j'}$
INH_{nc}	$\delta ih_{s1''}, \delta ih_{s2''}, \delta ih_{s3''}, \ldots, \delta ih_{i''}$	$\delta xh_{s1''}, \delta xh_{s2''}, \delta xh_{s3''}, \ldots, \delta xh_{j''}$
$INH^{(*)}$	$\delta ih_1, \ldots, (\delta ih)_i^*, \ldots, \delta ih_{i'}$	$\delta xh_1, \ldots, (\delta xh)_j^*, \ldots, \delta xh_{j'}$
$INH_c^{(*)}$	$\delta ih_{s1'}, \delta ih_{s2'}, (\delta ih)_{s3'}^*, \ldots, \delta ih_{i'}$	$\delta xh_{s1'}, \delta xh_{s2'}, (\delta xh)_{s3'}^*, \ldots, \delta xh_{j'}$
$INH_{nc}^{[*]}$	$\delta ih_{s1''}, \delta ih_{s2''}, [\delta ih]_{s3''}^*, \ldots, \delta ih_{i''}$	$\delta xh_{s1'}, \delta xh_{s2'}, [\delta xh]_{s3'}^*, \ldots, \delta xh_{j'}$

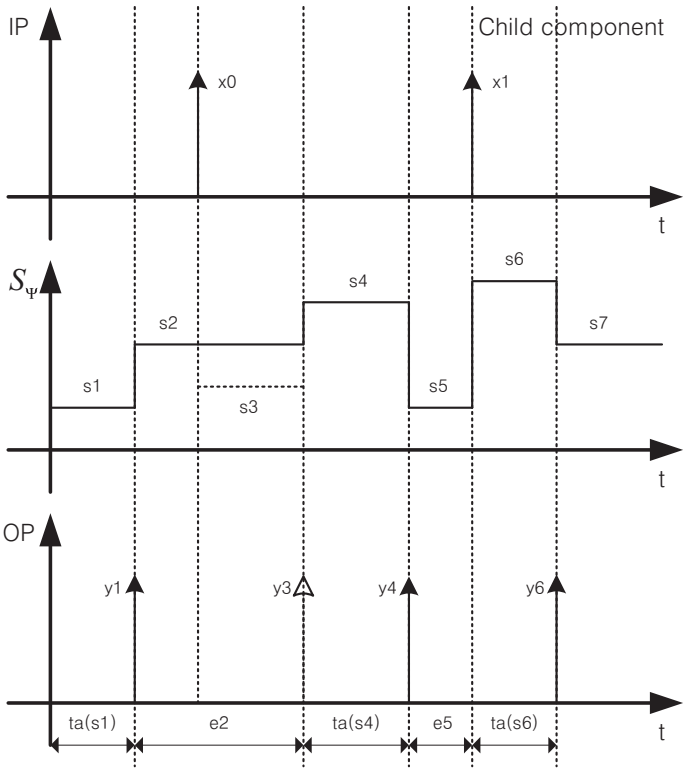


Fig. 4. Hidden inheritance.

Example 4. Fig. 4 shows an example of the hidden inheritance of the child component in Fig. 3. In the child component an external input event $x0$ is arrived to activate an external transition function, but by the hidden inheritance it does not show the same behavior of its parent component and the system state remains at $s2$. The output function to the external function is not worked to produce output value $y3$. System state $s2$ becomes the next system state $s4$.

Hidden inheritance consists of two different types: *complete hidden property* and *incomplete hidden property*. With a complete hidden property, the inherited asset is thoroughly hidden, but, with an incomplete hidden property, the asset is unstable and only partially hidden. Here, hidden strength influences whether the property exists as a complete or incomplete hidden property. If the hidden strength presents a decision value to the hidden inheritance, the type of completion mode would be selected by the decision value. The two hidden property types are illustrated in Fig. 5. Here, a parent DEVS component reproduces its child DEVS component, which receives the parent's properties, including structural as well as behavioral inheritance. After its own reproduction, the child component can then reproduce the next child component.

(1) *Complete hidden property*: With a complete hidden property, related input and output values and transition functions of a hidden behavior are concealed. Hence, a child DEVS component does not show any system behaviors related to the complete hidden property. However, the child component still contains the concealed asset though it cannot represent its behavior. Also, it is possible for the child component to pass on its concealed asset to the next child component. In doing so, a complete hidden property inherited from a parent component can be changed into a positive property during reproduction, which may allow the inherited behavior to be displayed normally.

The inheritance function including the complete hidden property is represented as: $[INH^{[*]} = (r_\Psi, Ih, Oh, Sh, \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h, hs, ta)]$, where the inheritance element functions for the complete hidden property are represented as $[ih]^*_m$, $[oh]^*_n$, $[\delta ih]^*_i$ and $[\delta xh]^*_j$. The elements of *Ih, Oh,* $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of $INH^{[*]}$ are listed in Table 1.

Fig. 5(a) shows a complete hidden property. In the figure, the inherited function $s3'$ of the first child component shows complete hidden inheritance style. Though the child component has the inherited function $s3'$ from the parent function $s3$, the child component does not show the behavior to $s3'$. However, the next child component can show active behavior to $s3''$ when it obtains the functional behavior $s3''$ from its parent component. In this case, as normal inherited function, the inheritance of $s3''$ does not show the complete hidden property.

Hence, in Fig. 5(a) the inheritance function of the first child component is given by $INH_c^{[*]} = (r_{fc}, Ih, Oh, Sh', \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h', hs, ta)$. The elements of *Ih, Oh,* $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of $INH_c^{[*]}$ are listed in Table 1. Also, the inheritance function of the next child component is $INH_{nc} = (r_{nc}, Ih, Oh, Sh'', \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h'', hs, ta)$. The elements of *Ih, Oh,* $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of INH_{nc} are listed in

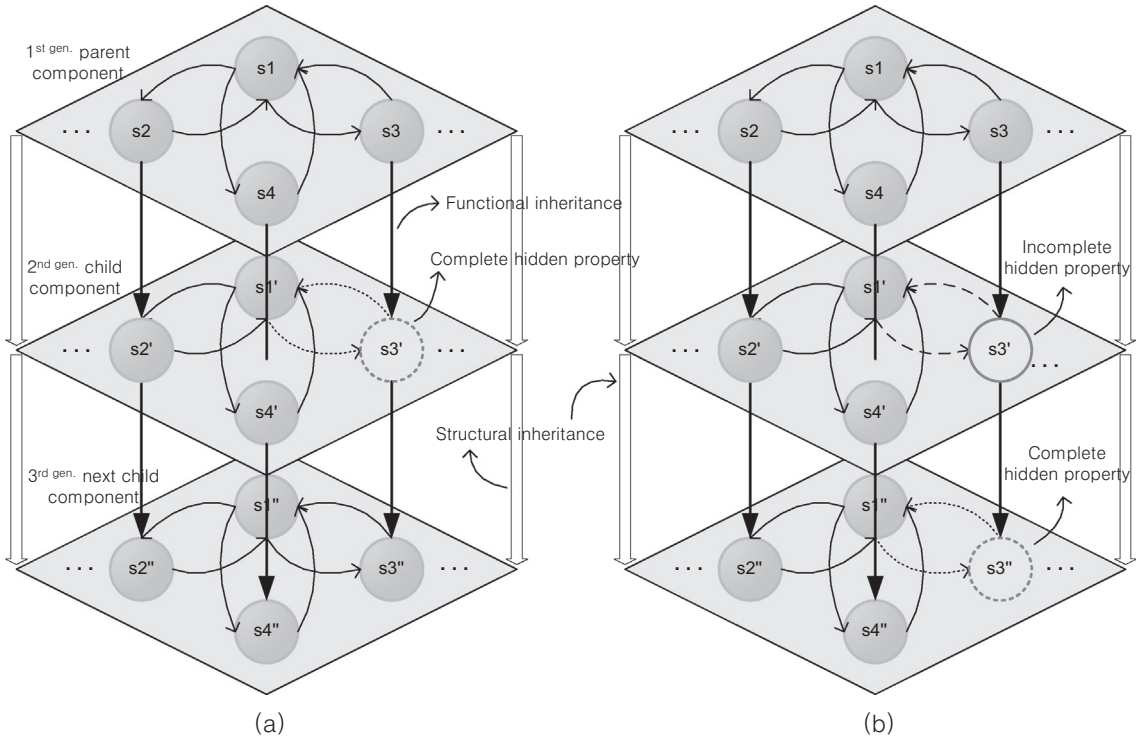


Fig. 5. Hidden inheritance type. (a) Complete hidden property. (b) Incomplete hidden property.

Table 1. The next child component is able to represent the same inherited behavior to the first parent component, when the inheritance is implemented as the *integrity* type.

(2) *Incomplete hidden property:* With an incomplete hidden property, partially inherited input and output values and transition functions can be related to reveal some behaviors in the hidden environment. When an inherited behavior shows an incomplete hidden property in the inheritance process, a partial function to the inherited behavior is activated to represent some of its actions. A child DEVS component can show unstable or irregular inherited behaviors for the incomplete hidden property even though its parent expresses normal behavior. When a parent DEVS component reproduces multiple child components that receive an inherited function from an incomplete hidden property, it is unusual for the received function to manifest unequally in these child components. Also, a child DEVS component that has an incomplete hidden behavior can reproduce a child component in which the incomplete hidden behavior can be changed into a normal or complete hidden behavior during reproduction.

The inheritance function that includes incomplete hidden property is given by $INH^{(*)} = (r_{\Psi}, Ih, Oh, Sh, \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h, hs, ta)$, where the inheritance element functions of the incomplete hidden property are represented as $(ih)^*_m$, $(oh)^*_n$, $(\delta ih)^*_i$ and $(\delta xh)^*_j$. By the hidden strength, the hidden type of a child can be the incomplete hidden type. The elements of *Ih, Oh*, $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of $INH^{(*)}$ are listed in Table 1.

Fig. 5(b) shows an incomplete hidden property. When a parent component makes a child component, the inherited behavior $s3'$ shows the incomplete hidden style. As mentioned above, even though another child component may get the functional behavior $s3'$ from the parent component as the same incomplete inheritance pattern, it can show different conduct. The inheritance behavior of the first child component is given by $INH^{(*)}_c = (r_{fc}, Ih, Oh, Sh', \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h', hs, ta)$. The elements of *Ih, Oh*, $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of $INH^{(*)}_c$ are listed in Table 1.

The next child component receives inherited behavior from the first child component that the behavior $s3''$ is in the complete hidden property. The inheritance function of the next child component is given by $INH^{[*]}_{nc} = (r_{nc}, Ih, Oh, Sh'', \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h'', hs, ta)$. The elements of *Ih, Oh*, $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of $INH^{[*]}_{nc}$ are listed in Table 1.

Therefore, there exist three composition cases of hidden type management caused by the hidden strength. An inheritance group may have only either complete or incomplete hidden type, or the group may possess child components showing both hidden types.

3.2. Multiple parent inheritance with a hidden property

The hidden property of an inherited behavior can be delivered to child components in different ways during the inheritance process. In special cases, a child DEVS component with multiple parent DEVS components receives a greater number of compositional inheritance characteristics. Further, in local cluster morphism, when specific parent components try to transmit an inheritance behavior to a child component, the child component selects only a subset of the parents' behavioral properties. Meanwhile, in *integrity* inheritance, the inheritance asset of selected parents remains intact in the child component. But, with a hidden property, some inherited behaviors become inactive, and the appearance of the inherited behavior is limited so that it appears as if the inheritance is of the *fragment* type.

We also consider hidden properties within the integrity type of global cluster morphism. In global cluster morphism, even though a child DEVS component receives all of the inheritance assets from certain parent DEVS components, it appears as if the child only partially receives these inheritance assets because of a hidden property. Hence, the child DEVS component will display inherited behavior from the specific parent components but not from all parent components, making this format similar to local cluster morphism of the integrity type. However, if the child component can only partially represent the behaviors of *all* parent components, then this is akin to the fragment type of inheritance. The child component still contains all of the inheritance assets from its parent components, but the expression of these assets is limited by hidden behaviors. Notably, the next child component can outwardly express all of the functional behaviors through integrity inheritance.

Fig. 6 illustrates the multiple DEVS parent inheritance with the hidden property. Assume that by the *integrity* type the first child component gets the functional inheritance from three parent components (*α-component*, *β-component* and *γ-component*), and each parent gives all self-inheritance assets to the next child component. Then, the next child component makes its child component with another component. Once through the cluster morphism, α-, β- and γ-components as the parent components give their all functional behavior to the first child component. If the succession is normally implemented without any variation, the child component should successfully form the inherited behavior to three parent components. However, as shown in the figure, the first child component only represents the inherited behavior of the α-component normally. In this case the inheritance asset of the β- and the γ-components is locked to hide active behavior. Here, during the inheritance, to the behavior of the β-component it follows the complete inheritance property, and the incomplete inheritance property is applied to the behaviors of the γ-component.

The inheritance function to the first child component is given by $INH_{mp\_fc} = (r_{fc}, Ih, Oh, Sh', \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h', hs, ta)$. In this case, the hidden strength affects the hidden inheritance of β- and γ-components. The elements of *Ih, Oh*, $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of INH_{mp_fc} are listed in Table 2. Then, the first child component makes the next child component by the *integrity* type without the hidden property. The hidden behavior of β- and γ-components in the first child component is released to appear in the next child component, i.e., the hidden property of the first child component is finished in the next child component. The inheritance function of the next child component is given by $INH_{mp\_nc} = (r_{nc}, Ih, Oh, Sh'', \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h'', hs, ta)$. The elements of *Ih, Oh*, $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of INH_{mp_nc} are listed in Table 2. Here, all inheritance properties of next child component show their behaviors to proper characteristics because the results of hidden strength stop the hidden inheritance.

In terms of a system component, we consider the continuance of the hidden characteristics of an inherited behavior. Variations within the system state can affect the hidden behavior; for example, a system component that holds a hidden behavior may undergo a state transition, which can change the hidden property to a normal mode of expression. Therefore, in this

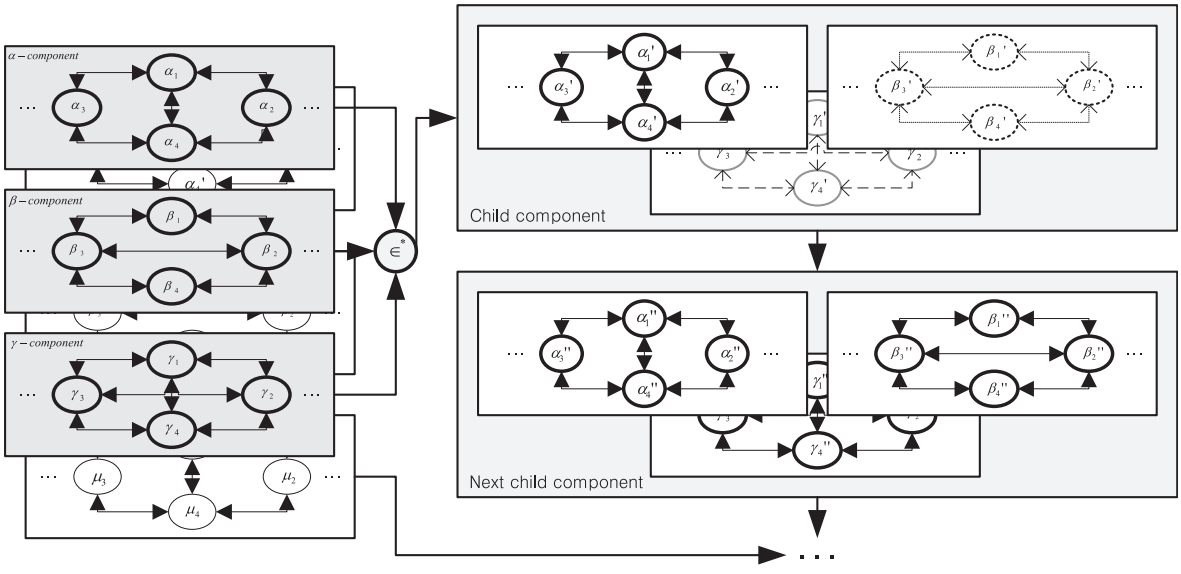


Fig. 6. Multiple parent *DEVS* inheritance.

Table 2
Elements of multiple parent inheritance with hidden property.

	Ih	*Oh*
INH_{mp_fc}	$ih_{\alpha'_1}, \ldots, ih_{\alpha'_i}, \ldots, ih_{\alpha'_{i'}}$, $[ih]^*_{\beta'_1}, \ldots, [ih]^*_{\beta'_m}, \ldots, [ih]^*_{\beta'_{m'}}$, $(ih)^*_{\gamma'_1}, \ldots, (ih)^*_{\gamma'_n}, \ldots, (ih)^*_{\gamma'_{n'}}$	$oh_{\alpha'_1}, \ldots, oh_{\alpha'_i}, \ldots, oh_{\alpha'_{i'}}$, $[oh]^*_{\beta'_1}, \ldots, [oh]^*_{\beta'_m}, \ldots, [oh]^*_{\beta'_{m'}}$, $(oh)^*_{\gamma'_1}, \ldots, (oh)^*_{\gamma'_n}, \ldots, (oh)^*_{\gamma'_{n'}}$
INH_{mp_nc}	$ih_{\alpha''_1}, \ldots, ih_{\alpha''_i}, \ldots, ih_{\alpha''_{i'}}$, $ih_{\beta''_1}, \ldots, ih_{\beta''_m}, \ldots, ih_{\beta''_{m'}}$, $ih_{\gamma''_1}, \ldots, ih_{\gamma''_n}, \ldots, ih_{\gamma''_{n'}}$	$oh_{\alpha''_1}, \ldots, oh_{\alpha''_i}, \ldots, oh_{\alpha''_{i'}}$, $oh_{\beta''_1}, \ldots, oh_{\beta''_m}, \ldots, oh_{\beta''_{m'}}$, $oh_{\gamma''_1}, \ldots, oh_{\gamma''_n}, \ldots, oh_{\gamma''_{n'}}$
INH^*	$(ih_1, \ldots, ih^*_m, \ldots, ih_{m'})$, $(\varnothing, \ldots, \widetilde{itr_m}, \ldots, \varnothing)$	$(oh_1, \ldots, oh^*_n, \ldots oh_{n'})$, $(\varnothing, \ldots, \widetilde{otr_n}, \ldots, \varnothing)$
INH'_{mp_fc}	$(ih_{\alpha'_1}, \ldots, ih_{\alpha'_i}, \ldots, ih_{\alpha'_{i'}}$, $[ih]^*_{\beta'_1}, \ldots, [ih]^*_{\beta'_m}, \ldots, [ih]^*_{\beta'_{m'}}$, $(ih)^*_{\gamma'_1}, \ldots, (ih)^*_{\gamma'_n}, \ldots, (ih)^*_{\gamma'_{n'}})$, $([itr]^{\sim}_{\beta'_1}, \ldots, [itr]^{\sim}_{\beta'_m}, \ldots, [itr]^{\sim}_{\beta'_{m'}})$ $= i\beta'^{\sim} = \varnothing$, $((itr)^{\sim}_{\gamma'_1}, \ldots, (itr)^{\sim}_{\gamma'_n}, \ldots, (itr)^{\sim}_{\gamma'_{n'}})$ $= i\gamma'^{\sim} = \varnothing$	$(oh_{\alpha'_1}, \ldots, oh_{\alpha'_i}, \ldots, oh_{\alpha'_{i'}}$, $[oh]^*_{\beta'_1}, \ldots, [oh]^*_{\beta'_m}, \ldots, [oh]^*_{\beta'_{m'}}$, $(oh)^*_{\gamma'_1}, \ldots, (oh)^*_{\gamma'_n}, \ldots, (oh)^*_{\gamma'_{n'}})$, $(o\beta'^{\sim} = \varnothing$, $o\gamma'^{\sim} = \varnothing)$
	$\delta_{h\text{-}int}$	$\delta_{h\text{-}ext}$
INH_{mp_fc}	$\delta ih_{\alpha'_1}, \ldots, \delta ih_{\alpha'_i}, \ldots, \delta ih_{\alpha'_{i'}}$, $[\delta ih]^*_{\beta'_1}, \ldots, [\delta ih]^*_{\beta'_m}, \ldots, [\delta ih]^*_{\beta'_{m'}}$, $(\delta ih)^*_{\gamma'_1}, \ldots, (\delta ih)^*_{\gamma'_n}, \ldots, (\delta ih)^*_{\gamma'_{n'}}$	$\delta xh_{\alpha'_1}, \ldots, \delta xh_{\alpha'_i}, \ldots, \delta xh_{\alpha'_{i'}}$, $[\delta xh]^*_{\beta'_1}, \ldots, [\delta xh]^*_{\beta'_m}, \ldots, [\delta xh]^*_{\beta'_{m'}}$, $(\delta xh)^*_{\gamma'_1}, \ldots, (\delta xh)^*_{\gamma'_n}, \ldots, (\delta xh)^*_{\gamma'_{n'}}$
INH_{mp_nc}	$\{\delta ih_{\alpha''_1}, \ldots, \delta ih_{\alpha''_i}, \ldots, \delta ih_{\alpha''_{i'}}$, $\delta ih_{\beta''_1}, \ldots, \delta ih_{\beta''_m}, \ldots, \delta ih_{\beta''_{m'}}$, $\delta ih_{\gamma''_1}, \ldots, \delta ih_{\gamma''_n}, \ldots, \delta ih_{\gamma''_{n'}}$	$\{\delta xh_{\alpha''_1}, \ldots, \delta xh_{\alpha''_i}, \ldots, \delta xh_{\alpha''_{i'}}$, $\delta xh_{\beta''_1}, \ldots, \delta xh_{\beta''_m}, \ldots, \delta xh_{\beta''_{m'}}$, $\delta xh_{\gamma''_1}, \ldots, \delta xh_{\gamma''_n}, \ldots, \delta xh_{\gamma''_{n'}}$
INH^*	$(\delta ih_1, \ldots, \delta ih^*_i, \ldots, \delta ih_{i'})$, $(\varnothing, \ldots, \delta \widetilde{itr_i}, \ldots, \varnothing)$	$(\delta xh_1, \ldots, \delta xh^*_j, \ldots, \delta xh_{i'})$, $(\varnothing, \ldots, \delta \widetilde{xtr_j}, \ldots, \varnothing)$
INH'_{mp_fc}	$(\delta ih_{\alpha'_1}, \ldots, \delta ih_{\alpha'_i}, \ldots, \delta ih_{\alpha'_{i'}}$, $[\delta ih]^*_{\beta'_1}, \ldots, [\delta ih]^*_{\beta'_m}, \ldots, [\delta ih]^*_{\beta'_{m'}}$, $(\delta ih)^*_{\gamma'_1}, \ldots, (\delta ih)^*_{\gamma'_n}, \ldots, (\delta ih)^*_{\gamma'_{n'}})$, $(\delta i\beta'^{\sim} = \varnothing, \delta i\gamma'^{\sim} = \varnothing)$	$(\delta xh_{\alpha'_1}, \ldots, \delta xh_{\alpha'_i}, \ldots, \delta xh_{\alpha'_{i'}}$, $[\delta xh]^*_{\beta'_1}, \ldots, [\delta xh]^*_{\beta'_m}, \ldots, [\delta xh]^*_{\beta'_{m'}}$, $(\delta xh)^*_{\gamma'_1}, \ldots, (\delta xh)^*_{\gamma'_n}, \ldots, (\delta xh)^*_{\gamma'_{n'}})$, $(\delta x\beta'^{\sim} = \varnothing, \delta x\gamma'^{\sim} = \varnothing)$

work, we describe the continuous as well as interrupted concealment of the hidden functional behavior. If a child component has a hidden behavior that remains purely in the existence of the child component, it guarantees continuous concealment, in which the state of a hidden behavior would not be changed to expose the concealed action. However, it is possible that the continuation of a hidden behavior is interrupted when behavior activity reaches the interrupted concealment. If a child component receives a hidden behavior with interrupted concealment and an event occurs to activate the interruption function, the hidden property of the behavior would be removed and it would appear as a normal behavior in the child component.

In the SR-DEVS component, the interrupted concealment is delivered in the inheritance function $INH^* = (r_\Psi, Ih, Oh, Sh, \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h, hs, ta)$, where the interruption element functions are represented as $\widetilde{itr_m}$, $\widetilde{otr_n}$, $\delta \widetilde{itr_i}$ and $\delta \widetilde{xtr_j}$. The elements of *Ih*, *Oh*, $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of INH^* are listed in Table 2.

The interrupted concealment in the inheritance function is connected to the hidden function. If a child component shows no interruption function to any hidden behavior, the child component contains un-changed, hidden behavior. Assume that inheritance function is given in the first child component in Fig. 6.

When the hidden behavior of β- and γ-components follows the continuous concealment, the interruption function is not provided that the inheritance function is the same as the above function by $INH'_{mp\_fc} = (r_{fc}, Ih, Oh, Sh', \delta_{h\text{-int}}, \delta_{h\text{-}ext}, \lambda h', hs, ta)$. The elements of *Ih*, *Oh*, $\delta_{h\text{-int}}$, and $\delta_{h\text{-}ext}$ of INH_{mp_fc} are listed in Table 2.

However, if the hidden behavior adopts the interrupted concealment, the interruption functions are inserted to apply the state change of behavior from the hidden mode to active behavior mode. Hence, to offer the interruption functions to the hidden behavior of β- and γ-component of the first child component in Fig. 6, the inheritance function is the same as INH'_{mp_fc} except that all "$(= \varnothing)$" are removed from all elements.

On the inheritance process, the child component may receive the hidden behavior having interruption functions by the inheritance function. The hidden property of a behavior is released by a triggered interruption function while simulating the component behavior.

3.3. Hidden inheritance of coupled SR-DEVS

In a coupled SR-DEVS model of the integrity type, a child coupled component receives all assets from its parent coupled component so that the parent properties dictate the entire structural form as well as the functional inheritance. After the

reproduction of a coupled system, internal elementary DEVS components implement their proper behaviors and can communicate with each other. Let us consider the hidden inheritance of the coupled SR-DEVS model. To determine the hidden inheritance, the connector C_{cp} for the inheritance relationship of the coupled DEVS model is given as follows:

$$C_{cp} = \left(P_{cp}, \{R_\Psi | R_\Psi \in *P\}, INH^*_{cp}, CON\right)$$

where P_{cp} is a parent coupled DEVS model, and INH^*_{cp} is the inheritance function including a set of hidden functions.

Here, the inheritance function INH^*_{cp} has a structure as follows:

$$INH^*_{cp} = (r_{cp}, Ih, Oh, d_M, \delta EI, \delta EO, \delta IC, chs)$$

where

r_{cp} is a child coupled DEVS model,

$d_M = \{d_{m1}, \ldots, d^*_{mn}, \ldots, d_{mn'}\}$ is a set of inheritance functions of sub-element DEVS components,

d^*_{mn} is an inheritance function to the hidden inheritance,

$\delta EI = \{\delta ei_1, \ldots, \delta ei^*_i, \ldots, \delta ei_{i'}\}$ is a set of inheritance functions of external input coupling connections that include a hidden function (δei^*_i),

$\delta EO = \left\{\delta eo_1, \ldots, \delta eo^*_j, \ldots, \delta eo_{j'}\right\}$ is a set of inheritance functions of external output coupling connections that include a hidden function $\left(\delta eo^*_j\right)$,

$\delta IC = \{\delta ic_1, \ldots, \delta ic^*_k, \ldots, \delta eic_{k'}\}$ is a set of inheritance functions of internal coupling connections that include a hidden function (δic^*_k), and *chs* is the component's hidden strength to determine the hidden inheritance of DEVS element.

Note that a parent coupled SR-DEVS component model becomes a general coupled DEVS model if the former does not have any child coupled component ($C_{cp} = \varnothing$). A coupled SR-DEVS model can hold several elementary DEVS components, which have the coupling connections to specify structural characteristics. The inheritance of three coupling connections and their inside elementary components are used to construct the structure of a child coupled component. In the coupled SR-DEVS model, only *integrity* inheritance is allowed so that a child coupled component inherits all behavior and structural properties from its parent coupled component.

However, hidden inheritance affects the expression of a child coupled structure. In this paper, we consider a hierarchical hidden mechanism for the coupled SR-DEVS model. The hidden mechanism of an atomic component has its own elementary hidden strength (see Section 3.1) while the coupled structure possesses the component's hidden strength for the hidden inheritance. Thus, if the component's hidden strength affects the component in the coupled structure, the child coupled structure would receive the component as a hidden property. If this hierarchical hidden strength of the coupled structure is $\varnothing$, it does not have any hidden characteristic to its components. Also, even though a component in the coupled structure will not mirror the entire hidden component in terms of the appearance value of the component's hidden strength, it may internally contain this elementary hidden strength.

The component's hidden strength might be a decision value, a threshold value, or a determination function to decide the hidden inheritance. Hence, when the hidden inheritance is determined by the component's hidden strength, the component at work will be hidden in the child coupled structure. Also, atomic components of a coupled structure can have the same component's hidden strength, or they may acquire different hidden strengths in the inheritance process. As mentioned earlier, although the child coupled structure cannot outwardly show the hidden component's properties, it has still inherited the component's assets. Therefore, it is possible for the child coupled structure to show different characteristics from its parent coupled structure through hierarchical hidden inheritance. In this way, a child coupled structure that has hidden properties may appear different from another child coupled structure that is produced by the same parent component.

Example 5. Fig. 7 illustrates a hidden inheritance of coupled SR-DEVS component. The parent coupled component (κ-*component*) reproduces two child coupled components (*G*-component and *H*-component) in Fig. 7(a). By the component hidden strength, the *G*-component has μ' $(\mu'_1, \ldots, \mu'_m, \ldots, \mu'_{m'})$ elements as hidden assets, and the *H*-component has λ' $(\lambda'_1, \ldots, \lambda'_k, \ldots, \lambda'_{k'})$ and ω' $(\omega'_1, \ldots, \omega'_n, \ldots, \omega'_{n'})$ elements as hidden assets. Here, the component hidden strength might be Boolean array values to each atomic component. Even though *G*-component and *H*-component have the same parent component and hold the same inheritance, they would show different coupled component movement. If an element component gets into hidden mode in the inheritance process, its overall behavior cannot be activated as if it were not included in the coupled component.

The inheritance function of *G*-component is given by $INH^*_{\kappa-component} = (r_{G\text{-}component}, Ih, Oh, d_{G\text{-}component}, \delta EI, \delta EO, \delta IC, chs)$. Also, the inheritance function of *H*-component is given by $INH^*_{\kappa-component} = (r_{H\text{-}component}, Ih, Oh, d_{H\text{-}component}, \delta EI, \delta EO, \delta IC, chs)$. The elements of both *G*-component and *H*-component are listed in Table 3.

Fig. 7(b) illustrates some examples of inheritance configuration to the coupled component. Assume that the *G*′-component is a next child coupled component from the *G*-component of Fig. 7(a), and reproduces its next child coupled component (*G*″-component). By the reproduction of inside DEVS element, the architecture of *G*′-component is varied to extend

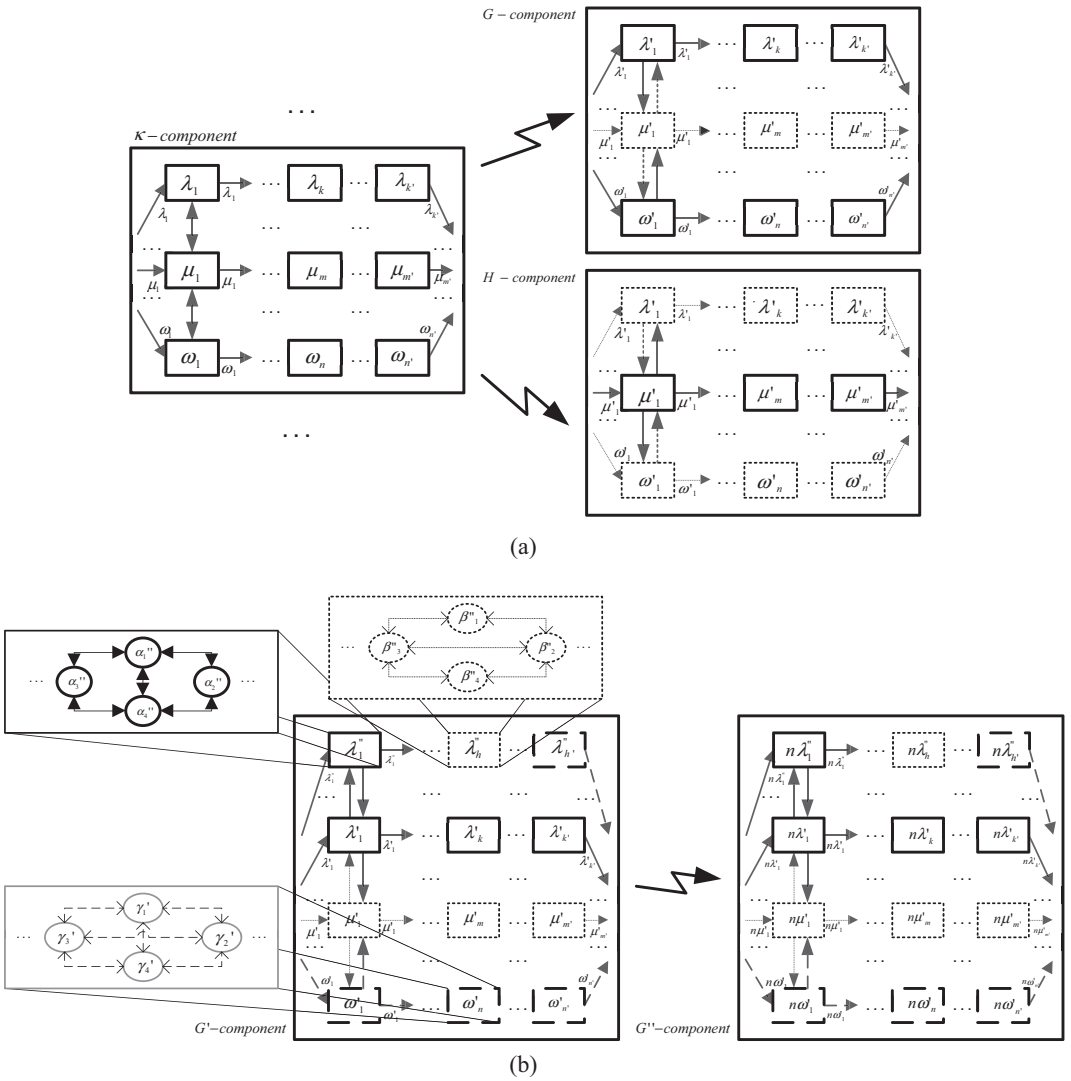


Fig. 7. Hidden inheritance of coupled *SR-DEVS* model.

component movement. Elements $\lambda''(=\{\lambda''_1,\ldots,\lambda''_h,\ldots,\lambda''_{h'}\})$ of the G'-component are child element components reproduced from the λ'.

In a coupled SR-DEVS model, an inside atomic DEVS element can reproduce its child component. In the G'-component, the element λ''_1 is reproduced as a normal component, but the element λ''_h is a hidden component by the complete hidden property. Also, the inheritance mode of ω' is changed into the hidden type during inheritance. The inheritance of G''-component is implemented without any variation. In this case, the component's hidden mechanism of G'-component leads the same hidden results for the hidden inheritance. Thus, the G''-component shows equivalent hidden characteristics to G'-component. Note that a hidden element in a coupled SR-DEVS component can be activated by a triggering interruption function such as atomic SR-DEVS component.

Example 6. Let us describe an example of probability model to determine the hidden inheritance by the component's hidden strength. We suppose the reproduction of coupled structure shown in Fig. 1. Its inheritance function is given by $INH_{com} = (r_{com}, Ih, Oh, d_{com}, \delta EI, \delta EO, \delta IC, chs)$. Once the parent structure implements its reproduction, the hidden determination to the received property of child structure is instantly determined by the hidden strength. Here, assume that the component's hidden strength to the reproduction of child coupled structures follows a probability distribution with $chs = f_{chs}(i)$ $(i = 0, 1, \ldots, n)$, and let the hidden determination of three components independently adopt an identical probability distribution model. We investigate two scenarios to the hidden determination through the component's hidden strength.

Table 3
Elements of *G*- and *H*-component in Fig. 7.

	G-component	*H*-component
d	$d_{\lambda' 1}, \ldots, d_{\lambda' k}, \ldots, d_{\lambda' k'}$, $d^*_{\mu' 1}, \ldots, d^*_{\mu' m}, \ldots, d^*_{\mu' m'}$, $d_{\omega' 1}, \ldots, d_{\omega' n}, \ldots, d_{\omega' n'}$	$d^*_{\lambda' 1}, \ldots, d^*_{\lambda' k}, \ldots, d^*_{\lambda' k'}$, $d_{\mu' 1}, \ldots, d_{\mu' m}, \ldots, d_{\mu' m'}$, $d^*_{\omega' 1}, \ldots, d^*_{\omega' n}, \ldots, d^*_{\omega' n'}$
δEI	$\delta ei_{\lambda' 1}, \ldots, \delta ei^*_{\mu' 1}, \ldots, \delta ei_{\omega' 1}$	$\delta ei_{\lambda' 1}, \ldots, \delta ei^*_{\mu' 1}, \ldots, \delta ei_{\omega' 1}$
δEO	$\delta eo_{\lambda' k'}, \ldots, \delta eo^*_{\mu' m'}, \ldots, \delta eo_{\omega' n'}$	$\delta eo_{\lambda' k'}, \ldots, \delta eo^*_{\mu' m'}, \ldots, \delta eo_{\omega' n'}$
δIC	$\delta ic_{\lambda' 1}, \ldots, \delta ic_{\lambda' k}, \ldots, \delta ic_{\lambda' k'}$, $\delta ic^*_{\mu' 1}, \ldots, \delta ic^*_{\mu' m}, \ldots, \delta ic^*_{\mu' m'}$, $\delta ic_{\omega' 1}, \ldots, \delta ic_{\omega' n}, \ldots, \delta ic_{\omega' n'}$	$\delta ic_{\lambda' 1}, \ldots, \delta ic_{\lambda' k}, \ldots, \delta ic_{\lambda' k'}$, $\delta ic^*_{\mu' 1}, \ldots, \delta ic^*_{\mu' m}, \ldots, \delta ic^*_{\mu' m'}$, $\delta ic_{\omega' 1}, \ldots, \delta ic_{\omega' n}, \ldots, \delta ic_{\omega' n'}$

In the first scenario, we concern that a certain component of kth child structure shows the first hidden property to its component if the parent structure continuously reproduces child coupled structures. That is, the component in the child coupled structure has been normally reproduced without having the hidden property until the $(k-1)$th reproduction, and the kth component reproduction has the hidden property. Let a random variable X be the number of hidden determination that has the Poisson distribution. Then, the probability function to the hidden determination is given by

$$f_p(X = i) = \frac{e^{-\lambda t}(\lambda t)^i}{i!} \tag{1}$$

where t is the unit time with the hidden parameter λ.

Since there exist $k-1$ hidden failures and kth hidden determination to the component, it follows the geometric distribution as

$$f_G(k) = (1-p)^{k-1} p \tag{2}$$

where p is the hidden determination probability to *chs*. Let us substitute (1) into (2). Then, we have

$$f_G(k) = \left[1 - \frac{e^{-\lambda t}(\lambda t)^i}{i!}\right]^{k-1} \left[\frac{e^{\lambda t}(\lambda t)^i}{i!}\right] \tag{3}$$

In the second scenario, we consider the occurrence number of hidden inheritance to a component. If the parent structure reproduces n child structures, j components in the n reproductions can have the hidden property. Thus, it follows the Binomial distribution as

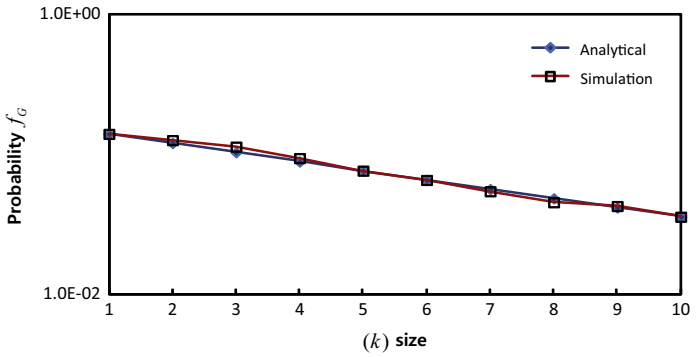
$$f_B(j) = \binom{n}{j} \left[\frac{e^{-\lambda t}(\lambda t)^i}{i!}\right]^j \left[1 - \frac{e^{-\lambda t}(\lambda t)^i}{i!}\right]^{n-j} \tag{4}$$

Fig. 8 shows the hidden probabilities of f_G and f_B by the hidden strength of component. For the hidden strength, assume that the hidden parameter λ is 5 and the number of hidden determination per unit time is 2. In this case, the probability value of hidden strength to a component of coupled structure is 0.14073. Fig. 8(a) illustrates the probability of f_G according to the different k size. The figure also shows both simulation results and analytical results. Now, we implement more than 100,000 reproduction samplings for the simulation. The results describe that the probability of the first hidden occurrence of the component to the geometric distribution is decreased gradually. It might mean that the hidden occurrence is an independent event, so that the hidden failure does not affect the next component reproduction. Also, the simulation results show almost the same probability results to the numeric analysis. Fig. 8(b) shows the probability of f_B as each j size. We assume that the parent structure reproduces 10 child structures each time. In the figure the probability f_B is dramatically decreased if the value j increases. This indicates that the probability value (0.14073) of hidden strength is not high, so that the consecutive hidden occurrence is difficult. However, when the value j is 1, the probability of hidden occurrence is higher than the value of 0. Even though the occurrence number to the hidden property is low, there can be a hidden component. The hidden determination to each component of child structure is independently determined by the hidden strength. Also, if each component has the same value of hidden strength, the distribution probability of hidden occurrence would be similar to each other.

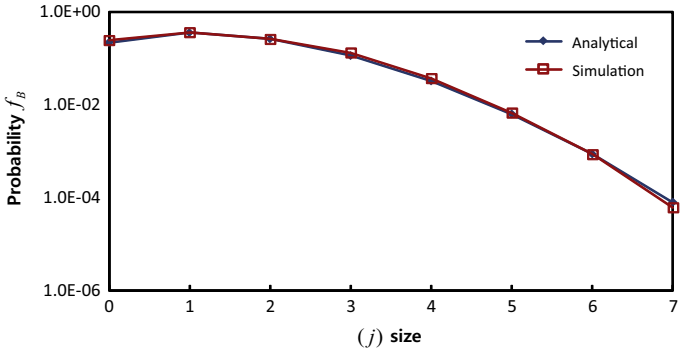
3.4. Case study of a social evolution model

In this section, the proposed model is applied to an evolution model that has a changeable property. In particular, we describe a case study of a social evolution model based on online social commerce. Also, we propose a verification methodology of evolution variation.

The social evolution model or *social Darwinism* has its origin in biological Darwinism [24,28]. It has borrowed many evolutionary endeavors from Darwinian methodologies [26]. The research range is extremely wide from social psychology to cultural evolutionary investigation, and many efforts have been made to analyze the format of evolution [27,28]. However,



(a) Probability f_G by k size



(b) Probability f_B by j size

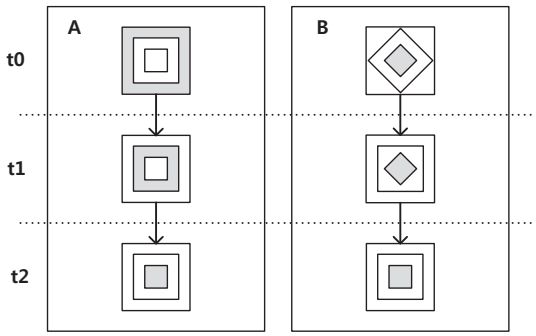
Fig. 8. Hidden probabilities of f_G and f_B by the hidden strength.

these attempts have yielded only a basic understanding that evolution is naturally continued from parent to offspring [29–31]. That is, the child generation model is merely considered to be the succeeding generation of the one that came before. Thus, inclusive fitness, for example, is simply provided as a condition for evolutionary success [25,31].

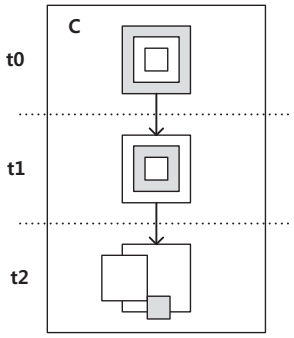
However, verification of evolutionary succession or variation can be requested in the evolution environment. Fig. 9 shows examples of an evolution model. Let the origin model appear at time $t0$, and suppose that the child model is evolved on the time flow. Fig. 9(a) shows the child evolution models for models A and B, respectively. Although the two original models are different from each other, at time $t2$, the two child models show the same evolved shape. In this case, it may provoke the question of whether the two child models at time $t2$ are the same or not. Fig. 9(b) shows an example of an evolution division model. Although the child model at time $t1$ can be regarded as a similar model of its parent with slight differences, the next child model at time $t2$ may look like a completely different model from its parent, raising the question of whether the next child model is a new-born model or not. Fig. 9(c) depicts another example of evolution division. The origin model of D reproduces two child models at time $t1$, which can be regarded as sibling models. However, at time $t2$, the next two child models take on different shapes, prompting additional questions about the evolutionary difference between the next two child models. For these questions, we must consider the evolutionary property of *similarity* in order to verify evolutionary succession or variation.

Similarity refers to the extent of resemblance between two or more evolutionary models and may provide a way to measure evolutionary succession change. Accordingly, we define the basic rules of *similarity* as follows:

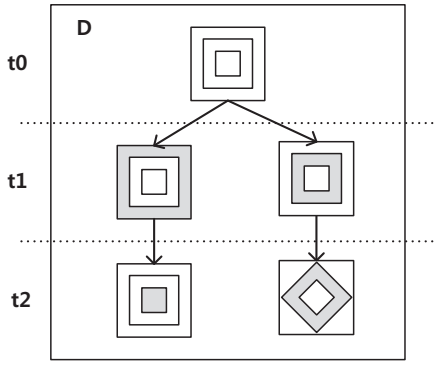
- If a child model shows a high *similarity* to its parent model, it can be considered as the succession model.
- If a child model shows low *similarity*, it is a division model or a new-born model.
- If two or more evolved models have some *similarity*, they can be considered sibling models based on the degree of this similarity. In this case, they should satisfy certain *similarity* conditions, such as deriving from the same original model or sharing an evolutionary history.
- If two or more evolved models show the same evolutionary properties, they are twin models. However, this decision depends on the measurement of *similarity*.
- The *similarity* can be represented as quantitative or qualitative values, and the evolution resemblance and variation can be judged through the measurement method.



(a) Two evolution models



(b) Evolution division I



(c) Evolution division II

Fig. 9. Examples of evolution model.

Human society is evolving toward tremendously complicated social models, including independent as well as global social models. Also present are applied social mechanisms, such as online social networks [32,33]. In these networks, some social evolution models are founded on adaptive social behaviors. Here, we note that online social networks may show particular social behavioral styles that compare to the autonomous social model, and the construction of this social model helps the social evolution [34].

An online shopping service that uses a social network for sales and advertising is an example of a recent phenomenon that combines two distinct online activities: shopping and social networking [35]. To identify the above-mentioned evolution variation, we consider the online shopping service that operates on a social network. Before online shopping, people were typically limited to purchasing products in face-to-face encounters between a customer and a seller. However, with the rapid development of online shopping, the social element of commerce has progressively changed. Through the online medium, people can have shopping experiences without the need to make a face-to-face purchase. Additionally, combining online shopping and social networking leads to a new shopping pattern by means of the online community, i.e., people using smart phones and tablet PCs to develop social relationships and build a social community based within the Internet. Huang and

Table 4
Social behaviors of shopping.

Social behaviors	Online social shopping
Shopping place	○ [36,38]
Shop visiting time	X [38,42]
Shop display	○ [39]
Product purchase contact	△ [38]
Purchase negotiation	○ [35]
Advertisement	○ [36]
Customer privacy management	◎ [40,41]
Government legal safeguard	○ [35,41]
Customer management	○ [42]
Purchase motivation and evaluation	◎ [43,44]

(○ – Continuous inheritance, ◎ – changed inheritance, △ – hidden inheritance and X – extinction).

Benyoucef [36] introduced a social commerce design model that includes commerce, community, conversation and individual elements. The community element, including conversation, supports social commerce. Within the social community, 83% of online shoppers attempt to share shopping information with other people [37]. In traditional offline shopping, there is undoubtedly "word of mouth" communication of purchase behavior. However, people receive this information only from family, friends, and a very limited social community.

This social behavior change affects social evolution and can lead to change in the social model. Online social commerce shows both different and similar social behaviors to the offline shopping experience. In Table 4, we list the social behaviors associated with social shopping that appear in recent literature. The first column of the table presents social behaviors of an offline shopping setting. To investigate the variation of social evolution, we classify the listed social behaviors into four variation properties: continuous inheritance, changed inheritance, hidden inheritance and extinction properties.

Inheritance means that the child's social behavior reflects its parent's social behavioral characteristics. In continuous inheritance, the social behavior is conveyed as is; however, in changed inheritance, the social behavior changes in certain situations while the social behavior succeeds to the next generation. In online social commerce, information security and privacy management should be carried out by the individual customer [40,41]. Also, regarding purchase motivation and evaluation, although people receive information by word of mouth for offline shopping, they tend to create a social community through online social commerce [43,44]. The hidden inheritance property is that the child behavior becomes hidden in the inheritance process. For example, a product purchase in modern offline shopping reflects the same behavior in terms of face-to-face contact between a buyer and a seller it did in the previous century. However, when people use online social commerce, they only engage with the online shop, not an actual person [36,38]. Online shopping can offer the possibility of face-to-face contact at some point in the transaction, but the social behavior of this contact might be hidden.

Sometimes, a social behavior can be stopped in a social environment so that it is the extinction of social behavior. The restriction of shop visiting time is released in online shopping. When people visit the offline shop, they can only go to the shop in the shop open time, but the online shop visiting is available 24 h [38,42].

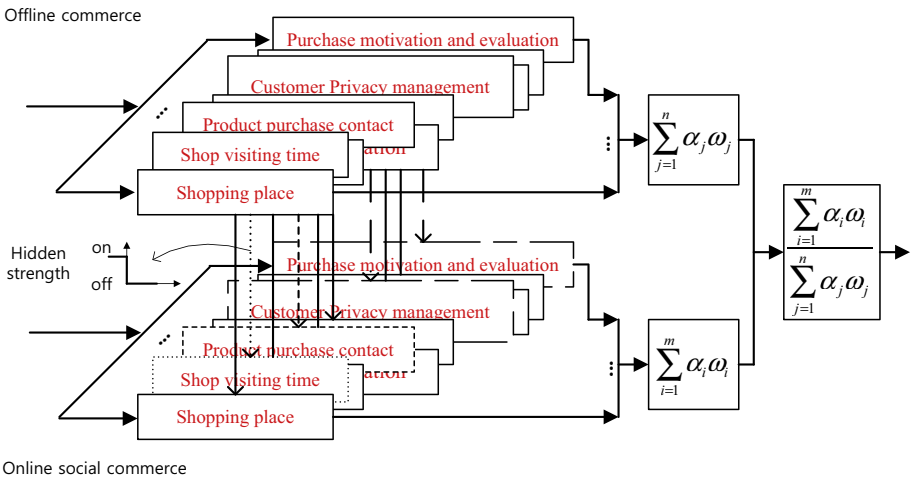
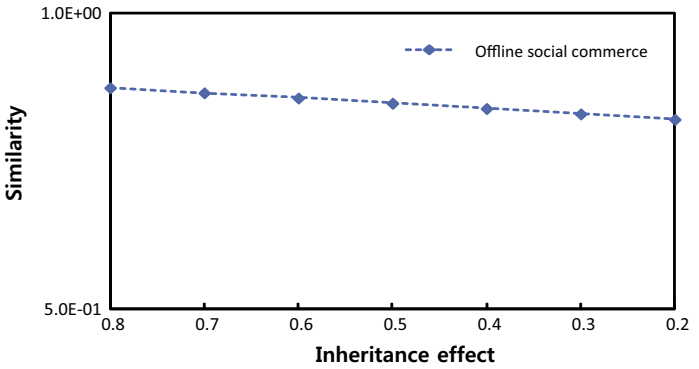
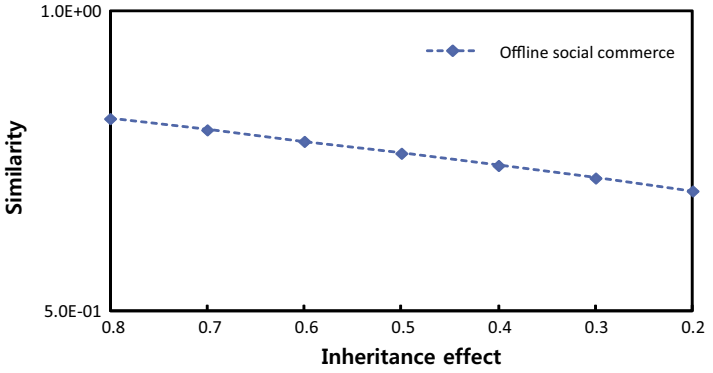


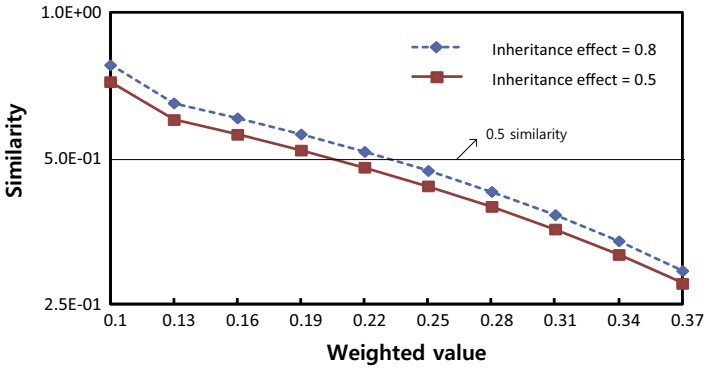
Fig. 10. DEVS simulation model of offline/online shopping.



(a) Hidden inheritance behavior



(b) Changed inheritance behaviors



(c) Similarity to weighted value

Fig. 11. Similarity of offline social commerce.

We apply Jaccard coefficient to investigate the *similarity* between the offline and the online social commerce as follows [45]:

$$Co(A, A') = \frac{A \cap A'}{A}. \tag{5}$$

Here, the denominator is A, because the comparative model is based on the parent model. That is, only parent social behaviors to the child model are considered to determine the *similarity*. If the social behaviors of intersection of sets between parent and child models are extracted such as $A \cap A' = \{\alpha_1, \alpha_2, \ldots \alpha_m\}$, we have the sum of the similarity value of each behavior as follows:

$$\lambda_{AA'} = \alpha_1\omega_1 + \alpha_2\omega_2 + \cdots + \alpha_m\omega_m = \sum_{i=1}^{m} \alpha_i\omega_i \tag{6}$$

where a is the inheritance effect of social behavior showing the variation property, and ω is the priority or weighted value to the social behavior. Hence, from (5) and (6), we have

$$\lambda_{sim} = \frac{\sum_{i=1}^{m} \alpha_i \omega_i}{\sum_{j=1}^{n} \alpha_j \omega_j} \tag{7}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_m, \alpha_{m+1}, \ldots \alpha_n (m \leqslant n)$, and $\sum_{j=1}^{n} \omega_j = 1$.

If a parent model reproduces a child model, and the child model has high similarity level ($\lambda_{sim} \geqslant \lambda'$), the child model is the succession model to the parent model. If ($\lambda_{sim} < \lambda'$), the child model will be a division model. Here, λ' is a relative value for the determination of similarity.

In the simulation implementation, we assume that the offline shopping model is the parent model to the online social commerce as shown in Fig. 10. Note that the social behaviors of parent model and child inheritance for the evaluation of evolution variation are presented in Table 4. If the continuous inheritance effect of social behavior is 1, the hidden strength of social behavior is simply determined by an on/off model. If the online social commerce receives continuous inheritance of all social behaviors from the offline commerce model, its similarity becomes 1 so that it is the same model without variation.

Fig. 11 shows the similarity of online social commerce. We assume that all of continuous inheritance behaviors in Table 4 have the same values of inheritance effect (= 1.0) and behavior weight (= 0.1), respectively. Fig. 11(a) shows the similarity by the hidden inheritance effect variation of purchase contact behavior. The figure indicates that if the hidden inheritance effect decreases, the similarity would also decreases. If a social behavior is hidden with some absent impacts, the child model may look different. Fig. 11(b) shows the similarity of changed inheritance behaviors. In Table 4, two social behaviors are taken as changed inheritance behaviors. The figure also shows that the similarity decreases if the inheritance effect decreases. The similarity of changed inheritance behaviors is lower than that of hidden ones. It means that the addition of changed behavior affects the evolution similarity. Fig. 11(c) describes the similarity by the weighted value variation. In the simulation, we increase the weight values of shop visiting time and purchase contact. Generally, in offline commerce, people visit the offline shop in the shop open time, and they have bought products through face-to-face contact to sellers. The restriction of shop visiting time and the face-to-face trade for the purchase contact traditionally seem to be basic social behaviors to offline commerce. Hence, if those social behaviors are assumed as important social behaviors of offline commerce, the online social commerce can have low similarity. In the figure, if two social behaviors have more than 0.25 weighted values with the inheritance effect of changed behavior of 0.8, the similarity drops below 0.5. Also, when the inheritance effect of changed behavior is 0.5, the similarity already drops below 0.5 to the weighted value of 0.22. We know that the similarity may be low if the social community of online commerce is regarded as more changed social behavior from the offline commerce. Here, if the similarity value of 0.5 is the decision value of evolution variation to the commerce model, the online social commerce is a division model. In other words, it becomes a new social evolution model in human society.

4. Conclusion

DEVS formalism is a typical methodology use in modeling and simulation that is continuously expanded to simulate discrete-based computing mechanisms. In various simulation modeling environments, this more elaborate kind of formalism can establish the validity of particular simulations and modeling. In this paper, we have proposed extended SR-DEVS, an additional modeling formalism, that models lifelike reproduction. We have introduced *hidden inheritance* to the SR-DEVS component model so that a hidden property occurs in the inheritance process. An SR-DEVS component that has a hidden behavior is characterized by limited functional behavior. In addition, we have considered the concept of hidden strength to determine the influence of a hidden property on inheritance. The hidden behavior would be an active behavior if the hidden property were stopped by an interruption function. In a coupled SR-DEVS model, a hidden element with self-execution affects the internal movement of the coupled component. Some SR-DEVS components can express different behavioral properties from one another although they share the same parent component. In this paper, we have also explained a case study that applies our proposed scheme to a social evolution model based on online social commerce. We have studied the evolution variation of online social consumerism versus the offline commercial experience. Accordingly, we expect that the extended SR-DEVS formalism can be used to design tools for modeling and simulating social system analysis, evolutionary or biological structural systems and similar computing systems with hidden properties.

References

[1] F.J. Barros, Modeling formalism for dynamic structure systems, ACM Trans. Model. Comput. Simulat. 7 (4) (1997) 501–515.
[2] F.J. Barros, Dynamic structure multiparadigm modeling and simulation, ACM Trans. Model. Comput. Simulat. 13 (3) (2003) 259–275.
[3] S.D. Chi, Model-based reasoning methodology using the symbolic DEVS simulation, Trans. Soc. Comput. Simulat. 14 (4) (1997) 141–152.
[4] A.C. Chow, B.P. Zeigler, Doo Hwan Kim, Abstract simulator for the parallel DEVS formalism, in: Proc. of IEEE AI, Simulation and Planning in High Autonomy Systems, December 1994, pp. 157–163.
[5] A.C. Chow, B.P. Zeigler, Parallel DEVS: a parallel, hierarchical, modular modeling formalism, in: Proc. of IEEE Winter Simulation Conference, vol. 2(2), December 1994, pp. 716–722.
[6] Seongmyun Cho, Taggon Kim, Real-time DEVS simulation: concurrent, time-selective execution of combined RT-DEVS model and interactive environment, in: Proc. of SCSC-98, July 1998, pp. 410–415.
[7] P.A. Fishwick, Simulation Model Design and Execution, Prentice Hall, 1995.

[8] Joonsung Hong, Haesang Song, Taggon Kim, Kyuho Park, A real-time discrete event system specification formalism for seamless real-time software development, Discr. Event Dyn. Syst. 7 (4) (1997) 355–375.
[9] Sangjoon Park, Byunggi Kim, Self-reproducible DEVS formalism, J. Parall. Distrib. Comput. 65 (11) (2005) 1329–1336.
[10] H. Saadawi, G. Wainer, Rational time-advanced DEVS (RTA-DEVS), in: Proc. of Spring Simulation Conference DEVS Symposium, April 2010, pp. 199–206.
[11] H. Saadawi, G. Wainer, From DEVS to RTA-DEVS, in: Proc. of IEEE/ACM Symposium on Distributed Simulation and Real-Time Application, October 2010, pp. 207–210.
[12] Changho Sung, Taggon Kim, Collaborative modeling process for development of domain-specific discrete event simulation systems, IEEE Trans. Syst., Man Cybernet. – Part C: Appl. Rev. 42 (4) (2012) 532–546.
[13] A. Troccoli, G. Wainer, Implementing parallel Cell-DEVS, in: Proc. of IEEE Annual Simulation Symposium, March 2003, pp. 273–280.
[14] Y.H. Wang, B.P. Zeigler, Extending the DEVS formalism for massively parallel simulation, Discr. Event Dyn. Syst.: Theory Appl. 3 (2/3) (1993) 193–218.
[15] B.P. Zeigler, H. Praehofer, T.G. Kim, Theory of Modeling and Simulation, Academic Press, 2000.
[16] B.P. Zeigler, S.D. Chi, Symbolic discrete event system specification, IEEE Trans. Syst., Man, Cybernet. 22 (6) (1992) 1428–1443.
[17] G. Wainer, N. Giambiasi, N-dimensional cell DEVS, Discr. Event Syst.: Theory Appl. 12 (1) (2002) 135–157.
[18] Hesham Saadawi, Gabriel Wainer, From DEVS to RTA-DEVS, in: Proc. of IEEE/ACM DS-RT, October 2010, pp. 207–210.
[19] Y. Huang, M.M. Seck, A. Verbraeck, LIBROS-II: railway modeling with DEVS, in: Proc. of IEEE WSC, December 2010, pp. 2150–2160.
[20] P. Bastien, A.S. Thierry, Wireless sensor network deployment using DEVS formalism and GIS representation, in: Proc. of IEEE SPECTS, July 2012, pp. 1–6.
[21] G. Wainer, M. Tavanpour, E. Broutin, Application of the DEVS and Cell-DEVS formalisms for modeling networking applications, in: Proc. of IEEE WSC, December 2013, pp. 2923–2934.
[22] S. Wang, M.V. Schyndel, G. Wainer, V.S. Rajus, R. Woodbury, DEVS-based building information modeling and simulation for emergency evaluation, in: Proc. of IEEE WSC, December 2012, pp. 1–12.
[23] S.E. Olamide, T.M. Kaba, Formal verification and validation of DEVS simulation models, in: Proc. of IEEE AFRCON, September 2013, pp. 1–6.
[24] C. Darwin, On the Origin of Species by Means of Natural Selection, or, the Preservation of Favoured Races in the Struggle for Life, 1859.
[25] W.D. Hamilton, The genetical evolution of social behavior. I, J. Theor. Biol. 7 (1) (1964) 1–16.
[26] W.D Hamilton, Narrow Roads of Gene Lean: Evolution of Social Behavior, vol. I, Oxford University Press, 1998.
[27] Richard R. Nelson, Evolutionary social science and universal Darwinism, J. Evol. Econ. 16 (5) (2006) 491–510.
[28] Geoffrey M. Hodgson, Generalizing Darwinism to social evolution: some early attempts, J. Econ. 39 (4) (2005) 899–914.
[29] T. Szekely, A.J. Moore, J. Komdeur, Social Behavior, Genes, Ecology and Evolution, Cambridge University Press, 2010.
[30] Adrian V. Bell, Peter J. Richerson, Richard McElreath, Culture rather than genes provides greater scope for the evolution of large-scale human prosociality, PNAS 106 (42) (2009) 17671–17674.
[31] Herbert Gintis, Inclusive fitness and the sociobiology of the genome, Biol. Philos. (November) (2013) (Online paper).
[32] Haibo Hu, Xiaofan Wang, Evolution of a large online social network, Phys. Lett. A 373 (12) (2009) 1105–1110.
[33] H. Kwak, C. Lee, H. Park, S. Moon, What is Twitter, a social network or a news media?, in: Proc. of WWW, April 2010, pp. 591–600.
[34] Yuan-Chu Hwang, Positive social evolution in autonomous social network communities, in: Proc. of EDR, May 2011, pp. 324–328.
[35] W.K. Tan, Y.J. Tan, Online or offline group buying?, in: Proc. of IEEE FSKD, August 2010, pp. 2853–2857.
[36] Zhao Huang, Morad Benyoucef, From e-commerce to social commerce: a close look at design features, Elsevier Electron. Commer. Res. Appl. 12 (4) (2013) 246–259.
[37] J. Chen, X.L. Shen, Z.J. Chen, Understanding social commerce intention: a relation view, in: Proc. of IEEE HICSS, January 2014, pp. 1793–1802.
[38] J. Zhang, Y. Yu, B. Xu, K. Zhu, A social network service-oriented architecture for mass customization, in: Proc. of IEEE CAIDCD, November 2009, pp. 2012–2015.
[39] R. Irfan, G. Bickler, S.U. Khan, J. Kolodziej, Hongxiang Li, Dan Chen, L. Wang, K. Hayat, S.A. Madani, B. Nazir, I.A. Khan, R. Ranjan, Survey on social networking services, IET Netw. 2 (4) (2013) 224–234.
[40] Moo Nam Ko, G.P. Cheek, M. Shehab, R. Sandhu, Social networks connect services, IEEE Comput. 43 (8) (2010) 37–43.
[41] Xiao Jiang, Privacy concern toward using social networking service: a conceptual model, in: Proc. of IEEE AIMSEC, August 2011, pp. 3180–3183.
[42] C. Grange, I. Benbasat, Online social shopping: the function and symbols of design artifacts, in: Proc. of IEEE HICSS, January 2010, pp. 1–10.
[43] Z. Chong, W. Bian, L. Benfu, P. Geng, Social network characteristics of online shopping interpersonal relationship in real and virtual communities, in: Proc. of IEEE CEC, September 2012, pp. 101–106.
[44] You Rie Kang, Cheol Park, Acceptance factors of social shopping, in: Proc. of IEEE ICACT, February 2009, pp. 2155–2159.
[45] S. Niwattnakul, J. Singthongchai, E. Naenudorn, S. Wanapu, Using of Jaccard coefficient for keywords similarity, in: Proc. of IMECS, March 2013, pp. 380–384.