Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

SIMULATION FRAMEWORK FOR SELF-EVOLVING AGENT-BASED MODELS: A CASE STUDY OF HOUSING MARKET MODEL

Jang Won Bae
Euihyun Paik
Dong-oh Kang
Junyoung Jung
Chun-Hee Lee

Smart Data Research Group
Electronics and Telecommunications Research Institute
Daejeon, 34129, SOUTH KOREA

ABSTRACT

Agent-based modeling and simulation (ABMS) has been applied to various domain problems. ABM consists of multiple agents and environments and focuses on their interactions. While such characteristics lead to its current popularity, they partially makes hard for the model calibration. Model calibration is generally performed by tuning model parameters, but in the ABM case, its interaction structure should be considered as well due to its huge influences to the results. To resolve this problem, this paper suggests a self-evolving ABMS framework. During the self-evolving process, ABM structure as well as parameters are explored and exploited for the model calibration. In particular, we adopt reconfigurable modeling in the proposed simulation framework, which is derived from Discrete Event System Specification (DEVS) formalism. This paper introduces a housing ABM, and the case study using this model shows that the accuracy of the model prediction was significantly increased through the self-evolving process.

1 INTRODUCTION

As the advances in ICT technologies and the dawning of big-data era, our societies and industries have become more complex and diversified. One of such examples refers to the fourth industrial revolution. For efficient and active responses to those changes, predictive analysis, which projects the future through the understanding of the present and the past changes, is undoubtedly in the limelight. In the case of socio-economic systems, in particular, the predictive analysis should entail the understanding on the reason that the predicted future happens, which makes modeling and simulation techniques being an alternative in the predictive analysis(Bae et al. 2016; Ouyang and Nelson 2017).

Among various methods, agent-based modeling and simulation (ABMS) is widely and frequently applied to the predictive analysis (Galland et al. 2014; Bae et al. 2014). ABMS generally explains behavior of target systems with agents and environments in the systems, and their interactions (Russell and Norvig 2016). Such micro-level modeling enables to analyze complex systems without clear understanding on the systems, which is normally impossible in the complex system. Therefore, this bottom-up manner would be a reason leading to ABMS's current success.

Simulation models including agent-based model (ABM) are developed referring to the dynamics and the real-data of target system. However, when predicting the future using simulation models, i.e., predictive analysis, we would face some limitations, in particular, in the long-term prediction case. Since the simulation models usually abstract the target systems at a certain time, there exists a gap (or error) between the simulated reality and the target systems in the future. The problem is that even if it was small enough to be neglected in the predictive analysis at the first time, the error would be increased and accumulated as simulation

execution proceeds. Moreover, eventually, the increased error could negatively affect the reliability of the simulation results (i.e., model validity).

The simulation models for the predicted analysis should be timely calibrated in a proper manner: the error gets larger as simulation tick goes, so the timing of the model calibration should be efficiently checked during simulation execution. Generally, the model calibration is conducted by tuning model parameters to fit the simulation results to the associated real-data. However, the general model calibration may not properly work in the ABM case due to the following reasons: 1) the interactions in ABM are the main factor on the system behavior, which, however, are referred as uncontrollable; 2) users are difficult to have a solid grasp on the complex systems that ABM are frequently applied to, and then it is also difficult to suggest a proper method for the model calibration. These problems should be considered for better predictive analysis using ABM.

This paper proposes a simulation framework to consistently maintain the ABM validity. The proposed framework, called self-evolving simulation (SES) framework, supports ABM automatically to be evolved (or calibrated) with respect to the gap between the simulation results and the associated real-data, which is called as self-evolving process. Specifically, the SES framework keeps track of the errors between simulation results and the associated real-data, and when the errors become significant, it activates model evolution process to reduce the error. The model evolution process considers the changes of not only the model parameters, but also the model composition. We note that due to the page limit, this paper mainly focuses on introducing the concept of the self-evolving process and its implementation in the SES framework.

This paper provides a case study for illustrating an efficiency of the self-evolving process in the SES framework. We developed a housing market agent-based model for the case study. This model describes housing transaction behaviors occurred in South Korea, and it also incorporates the considerations of the current Korean governmental polices on the housing market. Applying this agent-based model to the SES framework, the provided result shows that the model validity is increased through the self-evolving processes.

2 BACKGROUNDS

This section provides background knowledge for the proposed framework including self-organization in ABM and reconfigurable modeling. In contrast with other modeling methods abstracting systematic dynamics, ABM focuses on the collective behavior generated from individual behaviors and interactions. Based on such properties, ABM could be an effective method for complex systems whose overall behavior is hard to understand. In particular, it is well known that the complex systems hold a hierarchical structure inherently, and the ABM concept is efficient to represent its hierarchical structure (Ören et al. 2009; Jennings and Bussmann 2003; Cossentino et al. 2010).

Self-organization is referred as a process where the inner components varies their connecting structure in order to adapt their environments. This concept is a key factor considered in the ABM society because it generates distinguished effects, e.g., emergence, for understanding complex systems. To realize the self-organization, ABM tools should supports changes of the hierarchical structure during the simulation execution. In this sense, many researchers have worked on realizing the self-organization: Barbosa et al. proposes an ADACOR architecture for controlling manufacturing systems in the holonic perspective, and two-dimensional self-organization mechanism consisting of structural and behavioral level was incorporated (Barbosa et al. 2015); Rodiriguez et al. proposes a general-purpose agent-oriented programming language named SARL where the components in ABMs dynamically are enable to change their behaviors (Rodriguez et al. 2014); Ueda et al. applied the self-organization concept to biological manufacturing systems which adapts to dynamic changes in a product life cycle (Ueda et al. 1997).

This paper rather focuses on keeping the accuracy of the predictive results from ABMS, and we considered that the self-organization concept would play an important role in resolving this issue. Generally, model parameters take the role of handles for tuning model behaviors to be fitted to the target systems behavior.

In the ABM case, however, it might not be sufficient for the calibration. In other words, to calibrate ABM behaviors, model structure as well as model parameter should be considered. To realize the self-organization in a general way, this paper adopts systematical and theoretical basis for reconfigurable ABMS.

Reconfigurable ABMS means that ABMs are built from the combination of many components, and their interactions structure including a part and the whole of components could be replaced or removed during the ABMS execution. We expect that this method would broaden the scope of model calibration conducted only through the model parameter tuning. As a theoretical basis for the reconfigurable ABMS, we adopted Discrete Event System Specification (DEVS) formalism. DEVS formalism is a modeling framework supporting the hierarchical model structure (Zeigler et al. 2000). DEVS formalism is not invented for ABMS, but its semantic about modular and hierarchical structure is enough to apply to ABMS. In particular, reconfigurable modeling discussed in DEVS community can be utilized to resolve the calibration issue in ABM (Bae and Moon 2016; Zeigler et al. 2013; Bae and Kim 2010). The proposed SES framework provides model libraries for developing user models are developed as DEVS atomic and coupled models, but their roles are defined in ABM terms, such as agents, situation awareness, adopt and prediction, and decision-making. Compared with the previous works, the proposed SES framework provides a generality from DEVS semantics and advanced self-organization process where model parameters and structures are efficiently explored and exploited using AI techniques.

3 SELF-EVOLVING SIMULATION FRAMEWORK

3.1 Concept of Self-Evolving Process

The self-evolving process deals with continuous simulation-based analysis against to their decreasing accuracy as simulation time goes. The self-evolving process consists of change detection, evolution strategy, and model evolution (see Figure 1): 1) the self-evolving process keeps an eye on the simulation results so that we can specify the simulation tick when the simulated models become infeasible compared to the associated real-data. This monitoring phase would trigger and conclude the whole process (i.e., change detection phase); 2) regarding the identified simulation tick as the time to be evolved, the process investigates an evolution strategy that would improve the simulated models to be valid again (i.e., evolution strategy phase); 3) when the strategy is established, the process change the current simulation model as the strategy indicates from model parameter and structure perspectives. Then, it resumes the simulation from the identified simulation tick (i.e., model evolution phase). The below presents issues to be considered in each phase and methods that would be applied to tackle the issues.

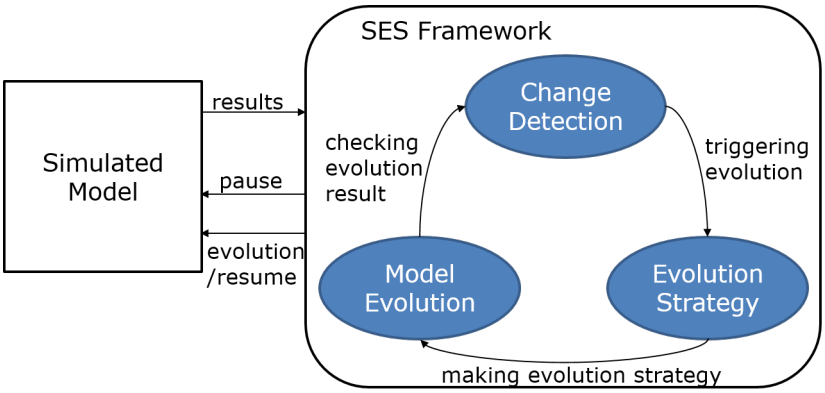


Figure 1: Self-evolving process provided from SES framework consisting of change detection, evolution strategy, and model evolution.

3.1.1 Change Detection

Change detection is to determine when a simulated model needs to be evolved. For making this decision, the real-data associated with simulation results are required, and this associations are built by model developers or users considering the objectives and the output of simulation models. For example, in the case of an ABM about housing transactions, the associated real-data could be the number of the transactions and the average price of sale prices according to size and region.

When comparing simulation results with the associated real-data, their might need data preprocessing for matching the period and the unit between two data. Generally, simulation models including ABM are developed for specific objectives, and the characteristics of their output, such as data frequency and unit, would depend on the objectives. Moreover, multiple real-data could be used to compare with a simulation result, and vice versa. In this sense, these characteristics are often required to be aligned for quantitative analyses. For example, the data frequency would be aligned via inter- or extrapolation with some losses, and the data types can be mathematically convertible, if they are computable.

With the preprocessed data, the change detection could be conducted in various ways depending on how to determine the change for model evolution. For example, the change can be derived from the point-by-point calculation, which means the change will be defined as the averaged errors on every time (e.g., root mean squared error or mean absolute percentage error). On the other hand, we focus on the change of data trend (e.g., real-data, simulation results, and their difference), model-based estimation methods(e.g., hidden Markov model or auto-regressive integrated moving average) could be applied. Having said that, we note that these decisions are made only for efficient model evolution. Figure 2 summarizes procedure and issues to be considered in the change detection phase.

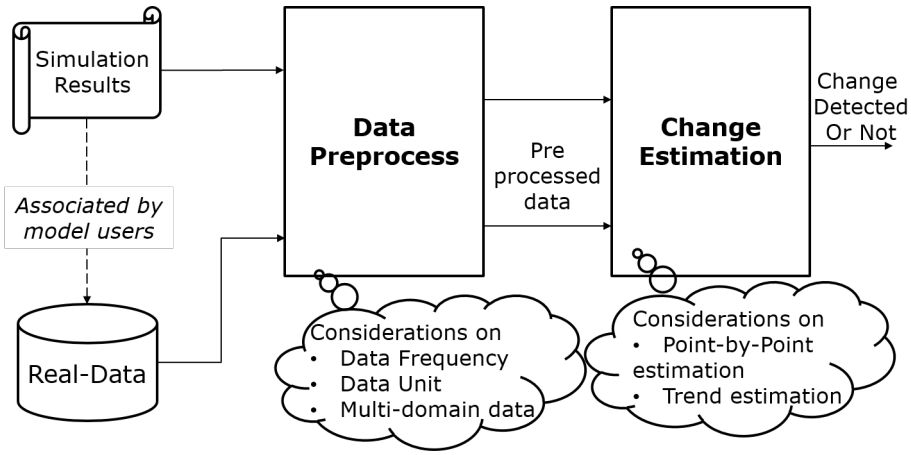


Figure 2: Procedure and issues in change detection phase.

3.1.2 Evolution Strategy

Evolution strategy, activated by the detected change, creates a guide for the simulated model how to increase its validity. In the evolution strategy phase, the self-evolving process gauges the potentials of the current simulated model and then discovers a way to improve it. The handles for evolving ABMs can be categorized into three kinds: The first is model parameters that are inherently identified to calibrate the model output; The second is modules used in a model, which could be a form of functions or sub-models. Those modules are imported for the utilization in the model development, such as better understanding and more reusability, but they also provide another axis in finding the evolution strategies; Lastly, it can be considered that the model itself would be substituted.

Meanwhile, another consideration is how to estimate model potentials and, eventually, to assure that its validity has been improved. There are two ways for these issues which are black-box (i.e., not knowing of the model details) and white-box method (i.e., knowing of how the model behaves). In the case of the black-box method, the input/output analysis is the only way to estimate the model potentials. Therefore, virtual experiments designed with the model handles are conducted, and the relationship between the model handles and outputs are analyzed using the experimental results. Based on these analyses, the model output can be improved for better model validity. On the other hand, while the black-box method requires a lot of experimental cases for understanding model behavior, the white-box method can infer it from investigating how the model is constructed and implemented. Hence, the white-box method would take shorter time than the black-box one, but the more complex the model structure is, more sophisticated inference algorithms are required. While the black-box and the white-box methods are inherently differed, for example, techniques used in simulation-based optimization and evolutionary algorithms are applicable to both. Figure 3 illustrates how the model evolution is conducted.

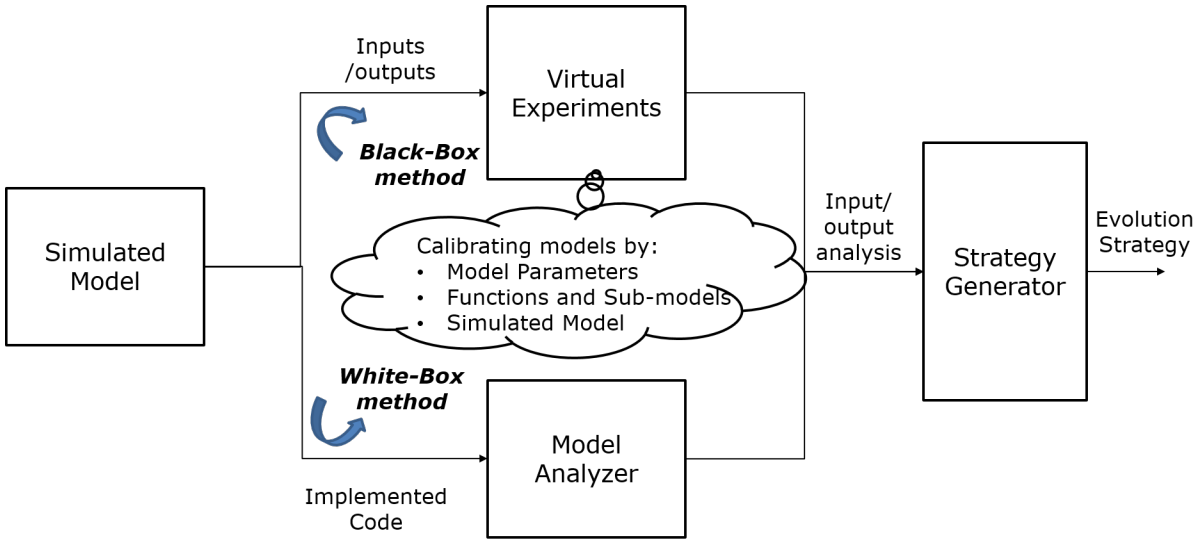


Figure 3: Procedure and issues in Evolution Strategy.

3.1.3 Model development and evolution

Model evolution is triggered by receiving an evolution strategy and interpret it to improve simulation models. Before explaining the model evolution, we would rather elaborate on the model development in the SES framework. The SES framework was designed for agent-based models where multiple agents and environments are interacted. Moreover, the considered agent-based models are hierarchically structured, which means the interactions among inner components are bounded, not globally interacted. These features have appeared in previous works, in particular, for describing complex adaptive systems (Bae et al. 2016; Jennings and Bussmann 2003). To support this ABM concept, the SES framework applied a modeling method that enables structured modeling where an ABM can be described as the loosely coupled components. This structured modeling is fully supported by, for example, Discrete Event System Specification (DEVS) formalism and its extensions (Zeigler et al. 2000; Bae and Moon 2016). Therefore, the SES framework adopted DEVS formalism as the theoretical basis of modeling and simulation method.

Applying DEVS formalism to the SES framework is not only for the model development, but also the model evolution. The model evolution is the final output of the self-evolving process. As we mentioned above, the model handles are used to develop an evolution strategy, and their levels (e.g., parameter-level, module-level, and model-level) would be varied by modeling methods. In particular, module- and model-

level handles are realized by DEVS formalism due to its modular components and abstract simulation algorithm concept. Based on such considerations, the self-evolving process develops an evolution strategy beyond the parameter level and enables dynamic model reconfiguration (i.e., changing a part of a model during the simulation execution), which was demonstrated as plug-in manner simulation (Bae and Kim 2010; Ewald et al. 2010), for the model evolution. Moreover, to realize component-level model reconfiguration, the SES framework should discover new component inserted into the simulated model. To this end, the self-evolving process requires a model repository where model components built before are stored, managed, and queried via DEVS semantics. From the perspective of the repository concept, it is similar with System Entity Structure (SES) formalism (Zeigler et al. 2013; Hwang 2005). Figure 4 shows an illustration of the above considerations.

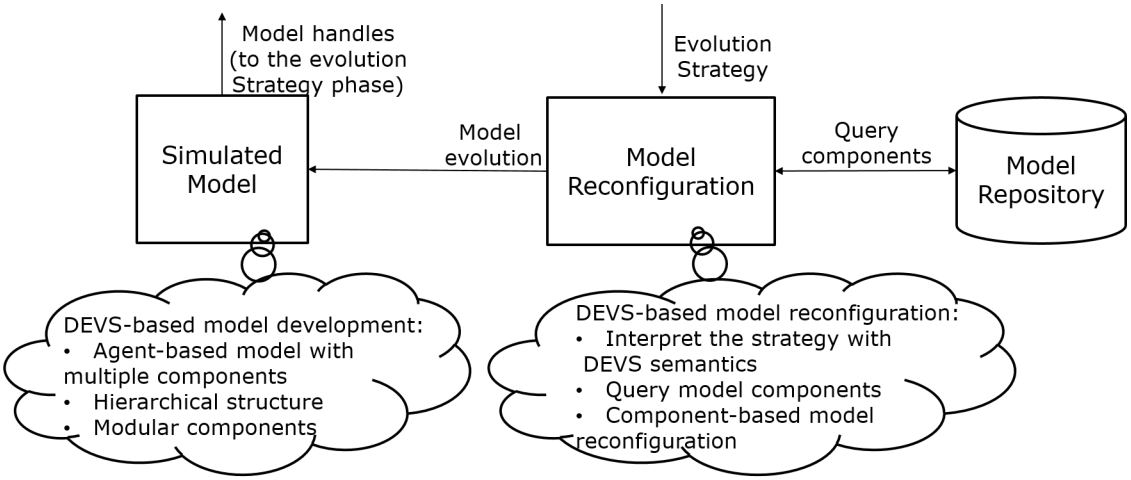


Figure 4: DEVS-based model development and evolution in SES framework.

3.2 SES Framework: System Architecture

To realize the self-evolving process explained above, we designed a self-evolving simulation (SES) framework. Figure 5 illustrates the architecture of the developed system consisting of the self-evolving process, ABM executor (i.e., simulation engine), and repository for data and components. Blocks for the self-evolving process (blue lined boxes in Figure 5) are change detector (CD), evolution manager (EM), model reconfiguration manager (MRM), and evolution life-cycle manager (ELM) blocks. Details of each block are presented as below:

- Change Detector (CD): CD block measures dissimilarity between simulation results and their associated real-data (i.e., prediction error). Detecting a change, it considers multi-level and -domain data and discovers meaningful features to find out the change.
- Evolution Manager (EM): EM block develops an evolution strategy to decrease prediction errors. To generate an evolution strategy, it identifies model handles of the concerned model and applies black- and white-box methods by the user's preference.
- Model Reconfiguration Manager (MRM): MRM block rebuilds simulation models following the received evolution strategy. To do this, it explores required components from the repository and assembles new simulation models with the discovered components. Moreover, for better component validity and query, component verification and component-specification conversion were provided.
- Evolution Life-Cycle Manager (ELM): ELM block controls the overall self-evolving process and mediates among the above blocks. For these roles, it holds the process controlling and monitoring functions.

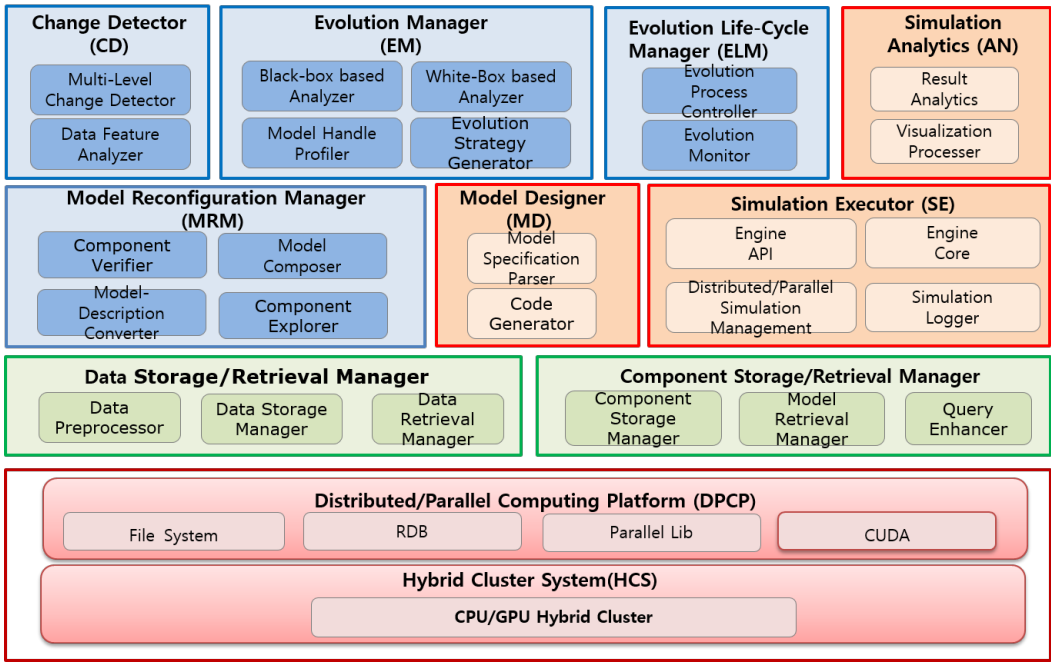


Figure 5: SES System Architecture: self-evolving process (blue line), simulation engine (red lines), and repository (green line).

The SES framework supports to design model components, execute component-based ABM, and analyze simulation results. Each function is implemented as Model Designer, Simulation Executor, and Simulation Analytics block:

- Model Designer (MD): MD block generates executable model codes from user-defined model specifications. To do this, there should exist a solid foundation between model development and simulation execution, and the SES framework applies DEVS formalism as the theoretical foundation. Specifically, the SES framework uses JSON for describing DEVS semantics, and MD block parse it and generate C++ code using the model library supported by simulation executor block.
- Simulation Executor (SE): SE block executes agent-based models described in DEVS semantics. SE block consists of engine APIs for user model development and engine core implementing the abstract simulation algorithm provided in DEVS formalism. In addition, it supervises distributed and parallel simulation execution using hybrid cluster system.
- Simulation Analytics (AN): AN block conducts statistical processes for mining meaningful results from the generated simulation logs and transforms the data structure for data visualization. Moreover, these operations are largely dependent on user's interests.

There are two kinds of warehouses in the SES framework: for real-data and for model components. Data storage and retrieval manager performs the preprocessing on the real-data associated with simulated models, or the manually preprocessed data are inserted into the repository. The stored data are queried by CD block for the comparison with simulation results. Component storage and retrieval manager stores user-developed model components, which are developed from MD block, and deals with a query for asking changeable components from MRM block.

Figure 6 shows the sequence diagram of the self-evolving process performed in the SES framework. Let us explain how the self-evolving process is conducted with the alphabetical order (from A to D) in Figure 6: initially, an agent-based model is simulated by the SE block, and then the generated simulation

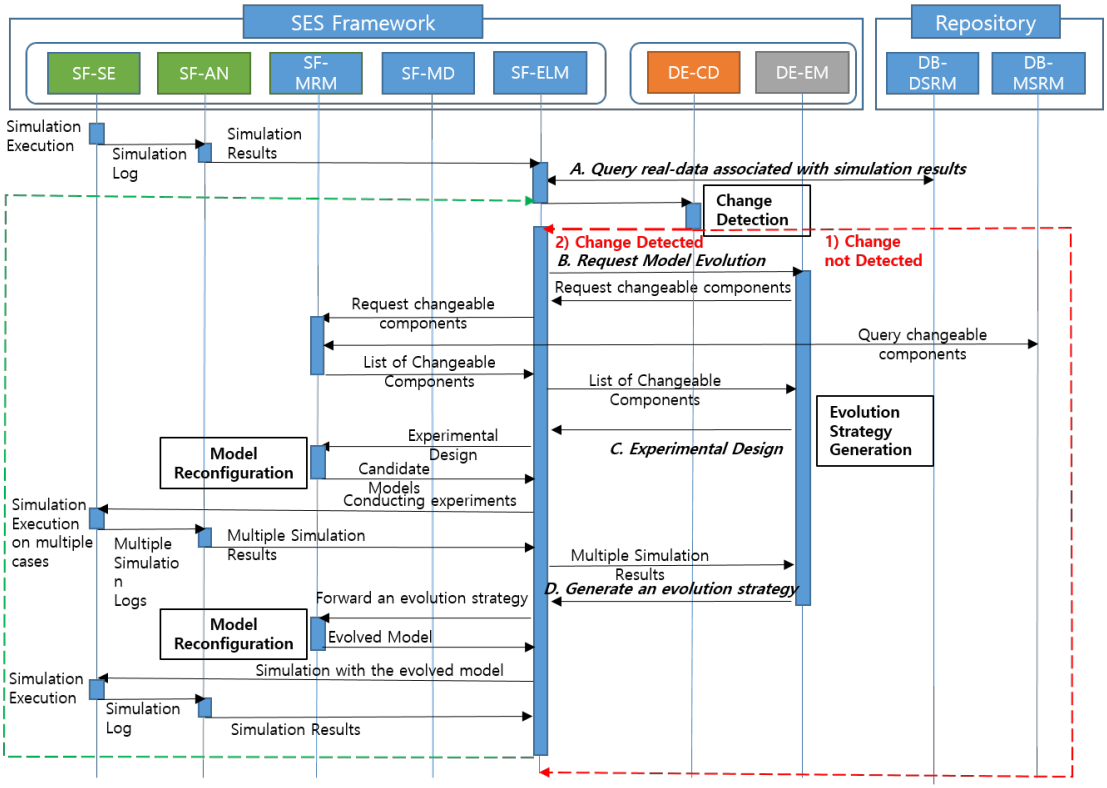


Figure 6: Sequence diagram for self-evolving process in the SES framework.

logs are forwarded to the AN block for processing simulation results; A) receiving the simulation results, the ELM block asks CD block to check the validity of the simulated model with the associated real-data. If the model validity turns out to be doubtful, the ELM block starts the self-evolving process; B) as the self-evolving process, the EM block firstly requests changeable components of the simulated model. The MRM block knowing the structure of the simulated model queries the changeable components to the model repository; C) The EM block builds an experimental design with the combination of the model parameter set and the changeable components from the model repository. Simulation results of each experimental cases are generated from the SE and AN blocks, and the EM block identifies the best evolution strategy for the simulated model; D) the generated strategy is realized by the MRM block. Simulation results from the evolved model are generated by the SE and AN blocks, and the CD block check the validity of the evolved model with the associated real data (go back to A phase, and this is called one-cycle of model evolution, or simply one evolution).

4 CASE STUDY: HOUSING MARKET MODEL

To exemplify the self-evolving process supported by the SES framework, we developed a housing market agent-based model (HousingABM) using DEVS-based modeling library from the MD block. This section shows an increase of the model validity through the repeated model evolutions (refer to Figure 6).

4.1 Housing Market Agent-Based Model

HousingABM describes housing transactions occurred in South Korea, and these transactions are dealt with a realtor that holds the information of listed houses (i.e., for sale) and mediates buying agents to the sellers. Results of the HousingABM are the number of transactions and the average price of selling houses in month. Moreover, transactions in HousingABM consider housing type (e.g., single house, multi-units

house, and apartment), dealing type (e.g., sale and rental (deposit with monthly pay)), and regional factor (e.g., capital and non-capital houses), since house prices are dependent on these features in South Korea. Also, we put the South Korea's governmental efforts to control the housing market, such as tax policies (e.g., transfer tax, acquisition tax, and loan regulations (e.g., Loan-To-value (LTV) and Debt-To-Income (DTI) ratios) into HousingABM. Transacted houses and agents are populated using panel data from the survey of household finances and living conditions in South Korea (KOSIS 2018).

HousingABM includes various model parameters about housing price and tax policies, but this paper would rather introduce a part of them that closely related with the self-evolving process. For the model evolution, parameters about the consumption sentiment were introduced in HousingABM. Specifically, housing agents are clustered into four groups by individual attributes, such as age, education level, marriage, etc., and external factors, such as GDP (Gross Domestic Product), the ratio of house transactions, inflation rate, etc. These groups represents an intent of the transactions. Hence, the introduced parameters are the size (i.e., how big the cluster size is) and the strength (i.e., how aggressively agents in a cluster would do the transactions) of each cluster. We note that these parameters have distinguished characteristics distinguished from others: they are not physically observed or recorded in the real-world (i.e., hidden variables). Therefore, they could be used as the calibrated parameters for the model evolution.

As evaluation measures for HousingABM, the number of transactions and the average of selling prices are calculated from the simulation results. Specifically, we break down these measures with the transaction type: the number of transactions of selling and rental houses; and the average of selling price of selling and rental houses. To quantitatively gauge the model validity, there need the corresponded real-values to these four measures. Fortunately, Korean government runs an official sites providing the information of housing transactions including price, region, house type, and transaction type (MOLIT 2018), so these real-data are used to check the validity of the HousingABM.

4.2 Result Analysis: Self-Evolution in HousingABM

To evaluate the error between simulation results and the associated real-data, we employed three types of error equations: 1) root mean square error (RMSE) is widely used measure for the difference between prediction and its real value. In particular, this paper defines the difference as a percent; 2) means absolute percentage error (MAPE) is another measure for the prediction accuracy of simulation results in trend estimation; 3) fitness is proportionally measuring the similarity of two values. The equations of three error measures are as follows:

$$\text{RMSE} = \sqrt{\sum_{t \le T} \frac{1}{T} \left(\frac{RealData_t - SimResult_t}{RealData_t}\right)^2} \tag{1}$$

$$\text{MAPE} = \frac{100}{T} \sum_{t \le T} \left|\frac{RealData_t - SimResult_t}{RealData_t}\right| \tag{2}$$

$$\text{Fitness} = \begin{cases} \frac{RealData_t}{SimResult_t}, & \text{if } RealData_t \le SimResult_t \\ \frac{SimResult_t}{RealData_t}, & \text{otherwise} \end{cases} \tag{3}$$

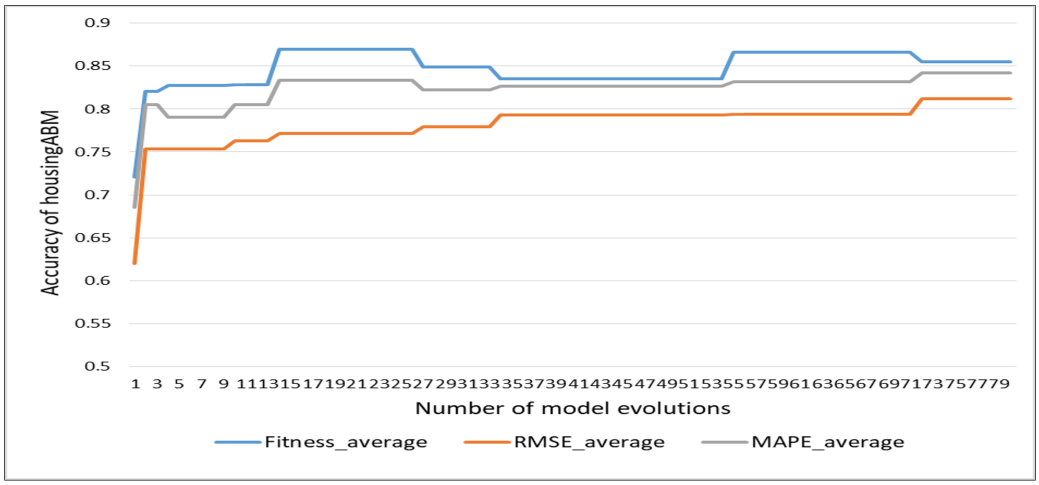
(where T is simulation end tick; $SimResult_t$ is the simulation results from HousingABM at simulation tick t; $RealData_t$ is the corresponded real-data to $SimResult_t$)

To quantitatively evaluate the model validity, we defined the accuracy of simulation models using the above error measures. In particular, if the simulation model creates multiple results (e.g., the transaction number and the average price in HousingABM), we calculate error measures for each result and average them for calculating the model accuracy:

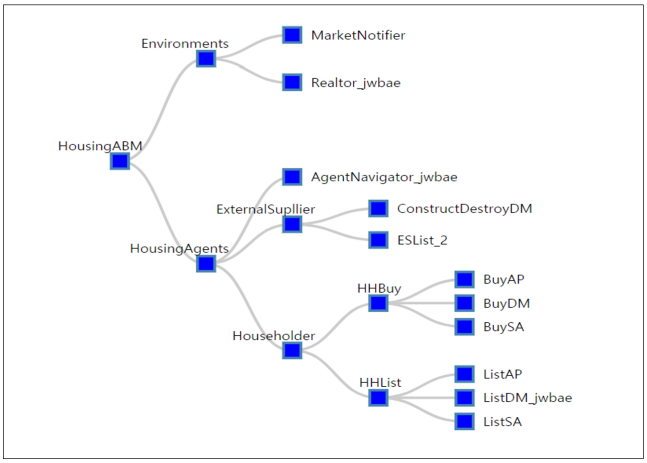
$$Accuracy_m(\text{Accuracy of model (m)}) = \begin{cases} (1 - Average_{i \in Idx}) * 100, & \text{RMSE and MAPE case} \\ Average_{i \in Idx} * 100, & \text{Fitness case} \end{cases} \tag{4}$$

(where Idx is a set of indexes of simulation results; $Average_{i \in Idx}$ is the average of the error measures for simulation results (in Idx))

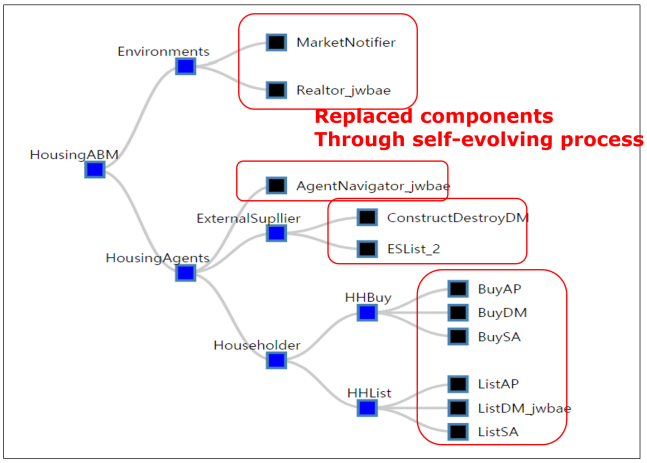
To test the HousingABM validity enhancement through the self-evolving process, we first evaluate the HousingABM accuracy without applying the self-evolving process, which becomes the baseline to further analysis. Then, we performed five trials of the self-evolving process on HousingABM. Due to the stochastic properties in the HousingABM, simulation executions are repeated in 30 times for each trial. Figure 7 illustrates the change of the model accuracy and the model structures as the model evolution proceeds in Trial 1; Table 1 presents the comparison of the model accuracy from the baseline and each trial.



Change of Model Accuracy as Model Evolution Proceeds



Model Structure at Initial



Model Structure after 70 evolutions

Figure 7: Result of self-evolving process: the accuracy change and the model structure change of HousingABM as model evolution proceeds.

Figure 7 shows that the model evolution strategy was developed along with increasing the RMSE based accuracy, so the accuracy values based on other error measures (i.e., MAPE and fitness) fluctuates while the RMSE based one keep increasing. Also, we can see the accuracy values are highly increased at the earlier stage, which is largely dependent on the initial model setting. Table 1 illustrates that the self-evolving process in the SES framework significantly increases the accuracy of HousingABM (maximally 19.2% increase in RMSE case). Despite of this success, we focuses on points to be considered from the Table. During the trial executions, for reference, each trial is manually stopped when the accuracy is over 80%,

Table 1: Accuracy of HousingABM with and without the self-evolving process.

Case	# of Evolutions	RMSE-based	MAPE-based	Fitness-based
Baseline	-	62.0%	68.5%	72.1%
Trial 1	70	81.2%	84.2%	85.4%
Trial 2	24	82.5%	84.9%	86.3%
Trial 3	35	81.2%	85.3%	87.7%
Trial 4	32	80.4%	83.0%	84.8%
Trial 5	28	80.7%	84.9%	87.4%
Avg. (w/ Baseline)	37.8	81.2% (+19.2%)	84.4% (+15.9%)	86.3% (+14.2%)

so the numbers of evolutions from the trials are varied. However, its variance is quite large (e.g., Trial 1 needs 70 evolutions, but Trial 2 does 24 evolutions). Moreover, the last evolution strategies of each trial have different sets of model parameters and structure. It indicates there may exist several way to improve the model accuracy, but the current version of the self-evolving process often fails to find best way.

5 CONCLUSION AND FURTHER WORKS

The self-evolving process enables simulation models to maintain their accuracy consistently, and this process is realized as the sequence of change detection by comparison of simulation results and the associated real data, evolution strategy identified via virtual experiment analysis, and model evolution with component-based model reconfiguration. The provided example model, HousingABM, is introduced to check an efficiency of the self-evolving simulation from the proposed framework. In the result analysis, we showed that errors between simulation results from HousingABM and the real-data associated with the simulation results were reduced after repeated model evolutions through the self-evolving process. There remains further works for the proposed framework: 1) on-line self-evolving process; 2) efficient components query in model reconfiguration, and 3) efficient method for discovering evolution strategies.

ACKNOWLEDGEMENTS

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (1711065140, Development of Predictive Analysis Technology on Socio-Economics using Self-Evolving Agent-Based Simulation Embedded with Incremental Machine Learning).

REFERENCES

Bae, J. W., S. W. Bae, I.-C. Moon, and T. G. Kim. 2016. "Efficient Flattening Algorithm for Hierarchical and Dynamic Structure Discrete Event Models". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 26(4):25.

Bae, J. W., and T. G. Kim. 2010. "DEVS Based Plug-In Framework For Interoperability of Simulators". In *Proceedings of the 2010 Spring Simulation Multiconference*, 127-1–127-7. Society for Computer Simulation International.

Bae, J. W., S. Lee, J. H. Hong, and I.-C. Moon. 2014. "Simulation-Based Analyses of An Evacuation from A Metropolis during A Bombardment". *SIMULATION* 90(11):1244–1267.

Bae, J. W., and I.-C. Moon. 2016. "LDEF Formalism for Agent-Based Model Development". *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46(6):793–808.

Bae, J. W., E. Paik, K. Kim, K. Singh, and M. Sajjad. 2016. "Combining Microsimulation and Agent-Based Model for Micro-Level Population Dynamics". *Procedia Computer Science* 80(1):507–517.

Barbosa, J., P. Leitão, E. Adam, and D. Trentesaux. 2015. "Dynamic Self-Organization in Holonic Multi-Agent Manufacturing Systems: The ADACOR Evolution". *Computers in Industry* 66(1):99–111.

Cossentino, M., N. Gaud, V. Hilaire, S. Galland, and A. Koukam. 2010. "ASPECS: An Agent-Oriented Software Process for Engineering Complex Systems". *Autonomous Agents and Multi-Agent Systems* 20(2):260–304.

Ewald, R., J. Himmelspach, M. Jeschke, S. Leye, and A. M. Uhrmacher. 2010. "Flexible Experimentation in The Modeling and Simulation Framework JAMES IIImplications For Computational Systems Biology". *Briefings in Bioinformatics* 11(3):290–300.

Galland, S., L. Knapen, N. Gaud, D. Janssens, O. Lamotte, A. Koukam, G. Wets et al. 2014. "Multi-Agent Simulation of Individual Mobility Behavior in Carpooling". *Transportation Research Part C: Emerging Technologies* 45(1):83–98.

Hwang, M.-H. 2005. "Generating Finite-State Global Behavior of Reconfigurable Automation Systems: DEVS Approach". In *Proceedings of the 2005 IEEE International Conference on Automation Science and Engineering*, 254–260. Piscataway, New Jersey: IEEE.

Jennings, N. R., and S. Bussmann. 2003. "Agent-based Control Systems: Why Are They Suited to Engineering Complex Systems?". *IEEE CONTROL SYSTEMS* 23(3):61–73.

KOSIS 2018. "Korean Statistical Information Service". http://kosis.kr/search/search.do. Accessed: 2018-04-15.

MOLIT 2018. "Korean Open Housing Transactions". http://rt.molit.go.kr/. Accessed: 2018-04-15.

Ören, T. I., L. Yilmaz, and T. Ören. 2009. *Agent-Directed Simulation and Systems Engineering*. Wiley-VCH.

Ouyang, H., and B. L. Nelson. 2017. "Simulation-Based Predictive Analytics for Dynamic Queueing Systems". In *Proceedings of the 2017 Winter Simulation Conference (WSC)*, 1716–1727. IEEE.

Rodriguez, S., N. Gaud, and S. Galland. 2014, Aug. "SARL: A General-Purpose Agent-Oriented Programming Language". In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)* 3(1): 103–110.

Russell, S. J., and P. Norvig. 2016. *Artificial intelligence: A Modern Approach*. Malaysia; Pearson Education Limited,.

Ueda, K., J. Vaario, and K. Ohkura. 1997. "Modelling of Biological Manufacturing Systems for Dynamic Reconfiguration". *CIRP Annals* 46(1):343–346.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Cambridge, Massachusetts; Academic press.

Zeigler, B. P., C. Seo, and D. Kim. 2013. "System Entity Structures for Suites of Simulation Models". *International Journal of Modeling, Simulation, and Scientific Computing* 4(3):1340006-1–1340006-11.

AUTHOR BIOGRAPHIES

JANG WON BAE, Ph.D, is a senior researcher at the Smart Data Research Group in ETRI. His e-mail address is jwbae@etri.re.kr.

EUIHYUN PAIK, Ph.D, is a principal researcher at the Smart Data Research Group in ETRI. His email address is ehpaik@etri.re.kr.

DONG-OH KANG, Ph.D, is a principal researcher at the Smart Data Research Group in ETRI. His email address is dongoh@etri.re.kr.

JUNYOUNG JUNG, Ph.D, is a principal researcher at the Smart Data Research Group in ETRI. His email address is jyjung21@etri.re.kr.

CHUN-HEE LEE, Ph.D, is a senior researcher at the Smart Data Research Group in ETRI. His email address is ch.lee@etri.re.kr.