Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

DEVS MARKOV MODELING AND SIMULATION OF ACTIVITY-BASED MODELS FOR MBSE APPLICATION

Abdurrahman Alshareef

Information Systems Department
College of Computer and Information Sciences
King Saud University
P.O. Box 2454, Riyadh 11451
Riyadh, SAUDI ARABIA

Chungman Seo
Anthony Kim
Bernard P. Zeigler

RTSync Corp.
6909 W Ray Rd STE 15-107
Chandler, AZ, USA

ABSTRACT

DEVS has been proposed as the basic modeling and simulation framework for Model-Based System Engineering (MBSE) methodology that supports the critical stages in top down design of complex systems. Here we propose a novel DEVS Markov simulation for Activity-Based Models to provide a means for experiment design to examine flows in activities with behavioral elements. The approach allows comparing different flows in a seamless manner. Also adjustments to the models become possible that would be costly and harder to maintain otherwise. With application to MBSE in mind, we aim to make such a process support developing more multiple alternatives in complex model design while delegating as much of code modification downstream to advanced automatic code generators. We examine the approach with application to MBSE with an example from the health care domain with different policies and patient flows. Finally, we discuss the remaining challenges and opportunities for further research.

1 INTRODUCTION

Zeigler et al. (2021) proposed a methodology that advocates using the DEVS formalism as the basic modeling and simulation (M&S) framework for MBSE methodology to support the critical stages in the design of complex systems of systems (SoS). The approach starts with high level specification of the behaviors required by the SoS to support system requirements engineering. This may be done using metamodels (e.g., UML/SysML) that map to simulation models (in formalisms such as DEVS) at an overall schematic level. Simulation infrastructures are needed to design, model, verify and validate SoS architecture designs with hybrid and co-simulation approaches. Analysis and design optimization must be supported by simulation-based exploration of design spaces to find solutions of interest.

Finally, integration support is needed to link high level architecture products and downstream models and simulators. Currently separate independent tools are needed to address the different phases of system engineering workflow. Use of such tool sets requires significant upfront investment, personnel training, and organizational burden (Chami and Bruel 2018). Recent studies have encouraged the development of architecture products that directly support downstream M&S artifacts critical for analysis and enable system design solutions to be explicitly linked to the systems engineering problems (Beery 2016). This calls for M&S to enhance the architectural products from MBSE with increased realism and enable continuity of elaboration and trace-back of high resolution simulation models to originating MBSE formulations.

In this paper, we discuss the use of Activity-Based Models to provide such integration. Such models offer a high-level complementary perspective to a variety of state-based discrete systems. The action and

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

control flow specification in the activity diagram provides a primary means to define various modeling and flow schemes such as different multi-processing regimes. While certainly insufficient to carry out a simulation experiment, our employment of the DEVS formalism (Zeigler et al. 2018) and other simulation algorithms come into play to provide a compliant yet rigorous underlying model by following practices of metamodeling, model transformation, and code generations. Such practices are heavily relied upon in software engineering as well as system engineering communities that manifest themselves in a variety of standards, guidelines, and semi-formal languages such as the Unified Modeling Language (UML), System Modeling Language (SysML), Model Driven Architecture (MDA), and Model-Based System Engineering (MBSE). There has been a growing interest in employing the DEVS formalism to provide a rigorous formal ground for such practices (Zeigler et al. 2018) to render their artifacts into concrete instances according to system-theoretic principles widely known in the theory of modeling and simulation. We use the term MBSE&TMS in the sequel to refer to the combination of MBSE and the theory of modeling and simulation (Wach et al. 2021).

We establish the relationship between activity diagrams and Markov chains through the probabilistic variant of the DEVS formalism with Markov property (aka. DEVS Markov (Seo et al. 2018)) and with further formalization for some activity constructs and decision/merge nodes in particular. The actions and control nodes are equipped with states to enable them to depict their trajectories over the time base after undergoing their corresponding transformation to manifest as DEVS atomic models. Then, the models are equipped with a probabilistic definition in some state transitions with time assignment and advance function definitions. Some elements of activities encounter stochasticities such as the decision/merge nodes, which we represent as the SELECT atomic model, and different timing definitions for the state of actions and control nodes and I/O generation.

This paper first discusses the integration of capability to create activity diagrams into MS4 Me (MS4 Systems 2018), a representative integrated M&S development environment. Then we discuss the representation and definition of activity diagrams as activity-based models in the DEVS formalism. We then discuss the Markovian representation of such models as DEVS models and the formalization of the flow selection process and state transition in the activity diagram. This is followed by demonstrating the approach with an example model of continuity of care in disease management. It will be apparent that many challenges remain and some opportunities for further research will be noted.

2 INTEGRATION OF MS4 ME AND ACTIVITY DIAGRAM

MS4 Me (MS4 Systems 2018) is a tool to construct DEVS models and simulate them, and it is built on Rich Client Platform (RCP) in an eclipse environment. MS4 Me contains eclipse environment plugins and DEVS modeling and simulation plugins. The eclipse plugins provide basic Integrated Development Environment (IDE) and the DEVS modeling and simulation plugins are for designing a domain system with graphical user interfaces (GUIs) called state diagram and sequence diagram, generating DEVS models and simulating them.

As shown in Figure 1, MS4 Me environment consists of Eclipse platform, Java development tooling (JDT 2021), and MS4 Me plugins which are constructed on plugin developer environment (PDE 2021). Required plugins are used to implement MS4 Me functions in MS4 Me plugins. JDT is used to compile and execute DEVS models constructed by Java language. Under eclipse platform, customized menus and editors are added for DEVS modeling and simulation. For developing a DEVS atomic model, domain specific language (DSL) is used called DEVS Natural Language (DNL) using a Xtext plugin (Xtext 2021). The DNL is automatically converted to a DEVS Java model through a Xpand plugin (Xpand 2021).

Figure 2 shows MS4 Me plugins created with required eclipse plugins. The DEVS modeling and simulation plugin contains core libraries to generate DEVS atomic and coupled Java models, and pruning process libraries with which a coupled Java model is created using an SES file and a PES file. The DEVS UI plugin creates menus and editors in the MS4 Me environment. The two plugins are constructed using JDT and PDE. DNL, SES, and PES plugins generate DSLs representing DEVS atomic models, DEVS

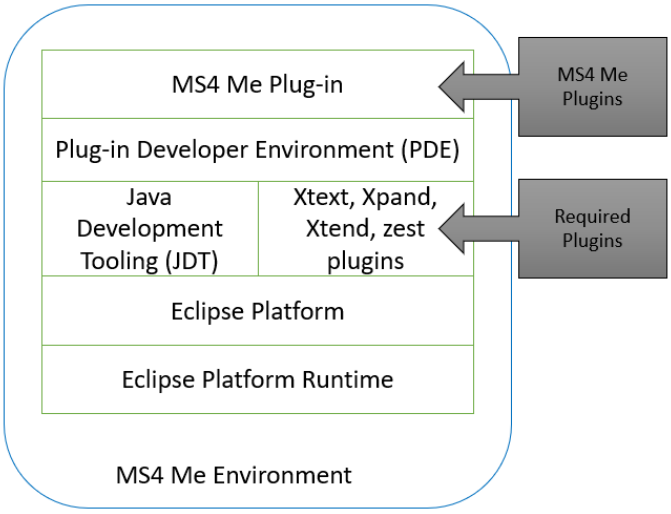


Figure 1: Structure of MS4 Me environment.

system structures, and specific DEVS coupled models, respectively. The plugins are built with Xtext, Xpand, and Xtend plugins. The Xtext is used to highlight keywords in a DNL file and transforms a Xtext model to Eclipse Model Framework model. The Xpand or Xtend (Xtend 2021) converts the EMF model (EMF 2021) to a DEVS Java atomic model.

DEVS modeling and simulation plug-in	DEVS UI plug-in	DNL, SES, and PES plug-ins	State Designer plug-in	Sequence Designer plug-in	Pruning plug-in	Simviewer plug-in	Graphs and charts plug-in	JSON plug-in
JDT & PDE		Xtext, Xpand, Xtend	SWT		Zest		Customized Plug-ins with libraries	

Figure 2: MS4 Me plugins on Eclipse plugins.

With the DNL plugin, DNL files are automatically transformed to DEVS atomic models. Unlike the DNL plugin, SES and PES plugins only highlight their keywords to help users construct DEVS system structures and their special DEVS coupled model. For domain experts to easily design their systems with the MS4 Me, a state designer and a sequence designer are added as plugins in the MS4 Me. They are customized for the DEVS models and built using Standard Widget Toolkit (SWT) plugin which contains graphic components. From the graphic design, the state designer generates a DNL file in the selected DEVS project and the sequence designer creates a SES file. To prune the SES file, MS4 Me provides a pruning plugin built using a zest plugin which is the Eclipse Visualization Toolkit. The pruning plugin displays entity nodes to be selected and after finishing the selection process, it generates a PES file. The *simviewer* plugin constructed with the zest plugin shows a coupled model and provides simulation control buttons. Users can see message flows in the *simviewer* when a step button is selected.

MS4 Me has a capability to show simulation results in various graphs using graphs and charts plugin which is built using Jfreechart open sources (Jfreechart 2021). JSON plugin containing JSON libraries is added to handle JSON messages in the DEVS models.

MS4 Me is expandable with customized plugins as seen in Figure 3. With the activity plugins, MS4 Me can design an activity model in a graphic environment provided by a Sirius plugin which helps generate customized graphical modeling tools. The Figure 3 shows the activity plugins from activity required plugins (Sirius (Sirius 2021) and Acceleo (Acceleo 2021) plugins) are integrated with MS4 Me plugins in MS4 Me environment. From the activity designer, users can build an activity diagram using the Sirius based plugin

and can generate DEVS atomic and coupled models from the diagram with the Acceleo plugin which generate DEVS Java codes from an EMF model to which the activity diagram is converted. As the activity plugins are included, MS4 Me has two project types called a DEVS Modeling project and a Modeling project for the activity diagram. The DEVS Modeling project should have DEVS Java codes generated from the activity diagram because the Modeling project cannot execute DEVS models. Before generating the DEVS Java codes, users can select a DEVS Modeling project from multiple DEVS Modeling projects. The activity plugins generate two DEVS atomic models called general DEVS atomic and DEVS Markov atomic model which contains randomness in a time advance function.

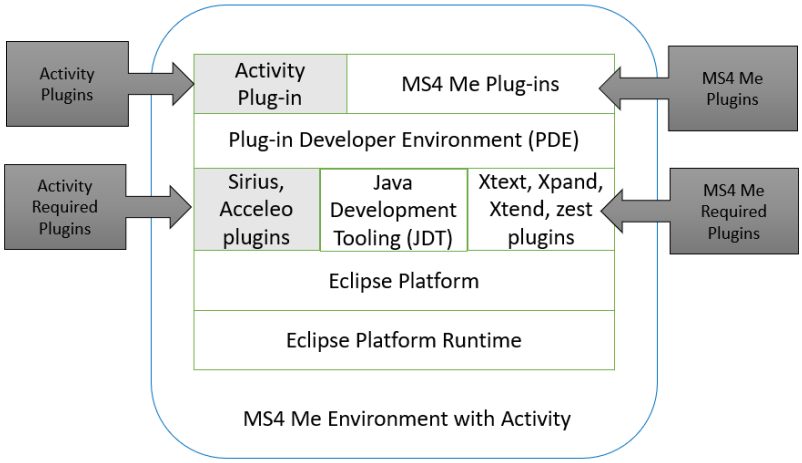


Figure 3: Activity plugins in MS4 Me.

MS4 Me provides a collaboration environment for domain experts and modelers. The domain experts can design their scenarios using the activity diagram, the sequence diagram and the state diagram. Also, they define any additional information such as message types, variables, and calculation functions used in DEVS atomic models. After designing the scenario, MS4 Me generates SES documents and DNL documents from the Sequence diagram and the state diagram. DNL documents are automatically converted to DEVS atomic models and SES documents can create PES documents using the pruning plugin. The PES document is used to create a DEVS coupled model which is displayed in the *simviewer* plugin. The other way to create a DEVS coupled model from the scenario is to use the activity diagram which generates DEVS atomic and coupled models from the activity design. The modelers help the domain experts implement the scenario requiring DEVS modeling knowledge.

3 DEVS MARKOV MODELING OF ACTIVITIES

We will briefly discuss the formalization process of the activity diagram resulting in our proposed activity-based model, which we have detailed in a recent dissertation (Alshareef 2019). Then, we will present the DEVS Markov modeling of the activity-based model, which is the paper's main contribution.

3.1 From Activity Diagram to Activity-Based DEVS Model

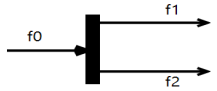
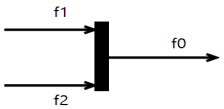
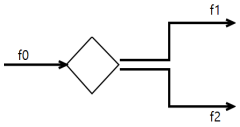
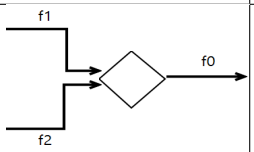
Due to fundamental limitations and inherent ambiguities in the activity diagram metamodel and the UML in general, modelers cannot seamlessly attain simulation or implementation. To be fair, the languages were initially proposed to facilitate software and system design processes. The general standards seem to be intended for visual documentation purposes rather than implementation or simulation. The consequence is that these languages pose limitations from the modeling and simulation perspective.

Therefore, we proposed in previous work (Alshareef and Sarjoughian 2017) a DEVS specification for activity elements with mapping for each construct in the activity diagram to a corespondent specification

in the DEVS formalism with accounts to their defined semantics of these constructs in the UML standards. We augmented the semantics of each activity and the nodes thereof with well-defined semantics via the formal specification of each construct. For example, the action in the activity diagram will be mapped to an atomic model with the necessary additional definition to make it formal in terms of a state. We also accounted for the parallel simulation as defined in the DEVS formalism to carry out for the parallel flows in activities. The parallelism semantics defined in this way is either missing or inadequate in the UML standards.

The nodes responsible for handling the control flow in activities are mapped into two atomic DEVS models. The *SYNC* atomic model represents fork/join nodes which are described in the first row of Table 1. And the *SELECT* atomic model represents decision/merge nodes which are described in the second row of Table 1. The behavior of all nodes and consequently the atomic models, by which their semantics are defined, is briefly described in Table 1. We note that in the activity-based model, we allow multiple flows to occur simultaneously or sequentially at a single run. The specification of the SYNC node will enforce a waiting phase in case some incoming flows are expected. Otherwise, it will proceed instantaneously. The flow through the SELECT node will proceed instantaneously. Other timing specifications can be defined by manipulating the time advance function. In the case of multiple inputs arriving simultaneously in the SELECT node, we do not currently provide a storage unit. Therefore one input will be selected from the bag of inputs, and others will be lost. We plan to provide other mechanisms to handle that in future developments.

Table 1: Different scenarios for control flow.

	After the arrival of I/O through $f0$, it will proceed after the fork node through both $f1$ and $f2$.		The I/O will proceed through $f0$ only when the join node receives it through both $f1$ and $f2$.
	After the arrival of I/O through $f0$, it will proceed through either $f1$ or $f2$ after the decision node based on the utility of the probabilistic state transition.		The I/O will proceed through $f0$ whenever the merge node receives it through either $f1$ or $f2$.

We have discussed the mapping, as well as the accounts for the parallel simulation with more details in previous works (Alshareef and Sarjoughian 2017; Alshareef and Sarjoughian 2018), respectively. We have also developed a code generation facility to generate the corresponding code for DEVS models in DEVS-Suite (ACIMS 2019) and MS4 Me environments. Then, we extended the code generation facility for DEVS Markov models. We discuss their formalizations and specification in this paper and an application in the health care domain.

3.2 DEVS Markov Formalization of the Activity-Based Models

We will discuss the activity-based-model formalization to arrive at a DEVS Markov model and, therefore, simulate them for some particular domain. We begin with the definition of aspects in the modeling of activities where stochasticity can be encountered.

First, we define the generator to feed the activity model with inputs based on a random variable (X) at some time instance t to be determined based on some probability p for X_t through which output port

the output is dispatched, in the case of having a single output port in the generator, $p = 1$. However, this definition targets the often case where generators have multiple output ports to generate different outputs and output types where each demands different processing. In addition to the determination of the output port, the time advance function (or aka. the sigma) is also determined based on some probabilistic distribution where the modelers can easily modify according to specific needs.

Secondly, and more akin to the activity modeling process, we define the selection process in *SELECT* atomic model, the correspondent model to *decision* and *merge* nodes in the activity diagram. This definition is relevant to the decision node where the selection of an outgoing flow for the output to proceed through is random. The *SELECT* atomic model is defined with a finite set of phases to be in. The number of phases is equal to the number of output ports (i.e., outgoing flows in activities) and passive and active phases. The phases that correspond to the output ports are defined as transitory states to identify through which output port the output/job should proceed. The time advance is defined for the active phase to indicate the time the job has consumed based on the selected probabilistic distribution. In addition to determining the output port, the time advance function is also defined similarly to its counterpart in the generator model. Figure 4a shows an instance of a decision node with three outgoing flows. Figure 4b shows the state diagram for the *SELECT* atomic model.

The specification in Figure 4 captures the essence of the continuous-time stochastic process in which the state of the system is defined across the continuous-time base in the DEVS formalism. The decision node with three outgoing flows has five phases $\{passive, active, stateOut_1, stateOut_2, stateOut_3\}$. The model stays at phase *passive*, and upon receiving input, it transitions deterministically to the active phase. The time advance function will be then determined based on the selected distribution for the active phase. The model then transitions to one of the three phases mentioned above defined as transitory states for the mere selection of the output port to assign the generated output to proceed through. The model then transitions from the selected phase to passive in a deterministic manner and stays there until further input is received. Hence the inputs that are received while the model in an active state is lost. We will discuss the other possibility where a storage unit is employed in the patient care example.

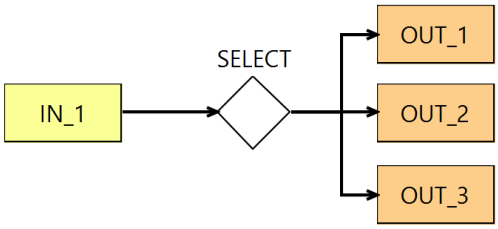
4 MODEL FOR SIMULATION BASED CONTINUITY OF HEALTH CARE

Modeling and simulation play an integral part in the delivery of health care services. The coordination between different agencies in such complex and critical systems is necessary for providing effective and successful treatment. One challenging aspect in treating patients is making sure a successful hand-off across multiple components on the continuity of care (CoC) pipeline (Zeigler 2016). The work in (Zeigler 2016) employs DEVS for modeling different pathways of coordinated care for HIV-AIDS patients.

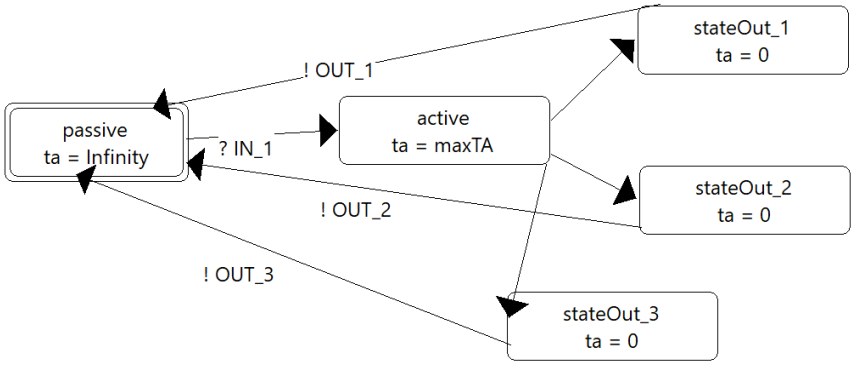
Work on that model has led to recognition that a critical unmet need in population disease management is how to integrate available population health data from multiple clinical and non-clinical sources, in real time. Such real-time data integration is needed to model, simulate, and analyze intervention outcomes, and ultimately optimize intervention planning of care coordination processes to support value-based population healthcare management. In an application to Diabetes Type 2 management, an MBSE-based approach to such integration starts with modeling the patient journey in the CoC pipeline as illustrated in Figure 5. Here the first three stages of the model concern screening for general healthcare access and diabetes problems. This is followed by induction into a bifurcated design enabling randomized testing for potential benefit of proposed interventions into current diabetes prevention programs.

In this manner, real-time data capture and model-based analysis offers insight into the effect of the various interventions patients receive at each stage and affords the possibility of refining these interventions in real-time as the data accumulates.

In the following sections, we illustrate the application of activity-based M&S in MBSE by discussing two models. The first elaborates on the patient journey in the first three stages of program initiation while the second models the downstream bifurcation component. Each model illustrates different capabilities of activity-based M&S in an MBSE top-down design approach.



(a) Example *SELECT* with three outgoing flows.



(b) The state transition diagram for the *SELECT* atomic model.

Figure 4: The DEVS Markov specification for the decision node in the activity diagram.

Figure 6 starts to show how we create a DEVS simulation model for the initiation part of the CoC pipeline. This construction will illustrate how we devised the activity-based metamodel with close consideration to Parallel DEVS (PDEVS) to help, on the one hand, in complementing the activity modeling process with the precise semantics offered by the DEVS formalism, and on the other hand, to behaviorally enrich the simulation modeling process with the flow control concepts offered in the activity approach.

4.1 Modeling the initiation part of the CoC Pipeline

We start the simulation modeling process by sketching the activity diagram that can represent the patient journey to capture the related element in a particular case. In our case, a patient arrives at the clinic via three possible ports, contact tracers, community organizations, or by self-referral. Then, the patient admission to the clinic goes through an eligibility assessment process where the outcome determines whether the patient is admitted to the care program after undergoing another two assessment processes, one for the food security program and another for the physician care program (PCP). Each one of these three eligibility assessments is represented by a *SELECT* (decision) node, which will correspond to a formal DEVS specification detailed in (Alshareef 2019), as well as the formal specification of the *SYNC* node. From a behavioral standpoint, the former describes the semantics of the decision and merging nodes of the activity in a simulation modeling arena after complementing them with the necessary state definitions, initialization, transition functions, time advance, and output functions. The three *SELECT* nodes that correspond to each assessment are *EligibilityAssessment*, *LackFoodSecurity*, and *LackPCPother*. We also define three other *SELECT* nodes to merge their incoming inputs.

Each alternative in Figure 6 has some possible entry parameters. The activity in Figure 6a has one initial eligibility assessment before going through the other two specific eligibility assessments. This alternative

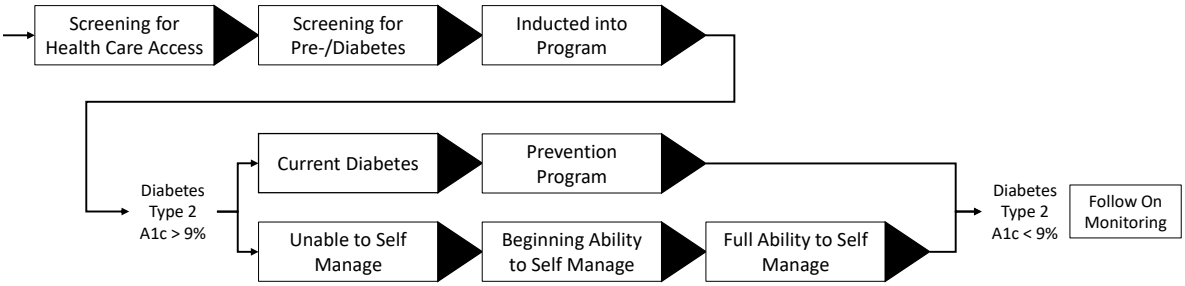


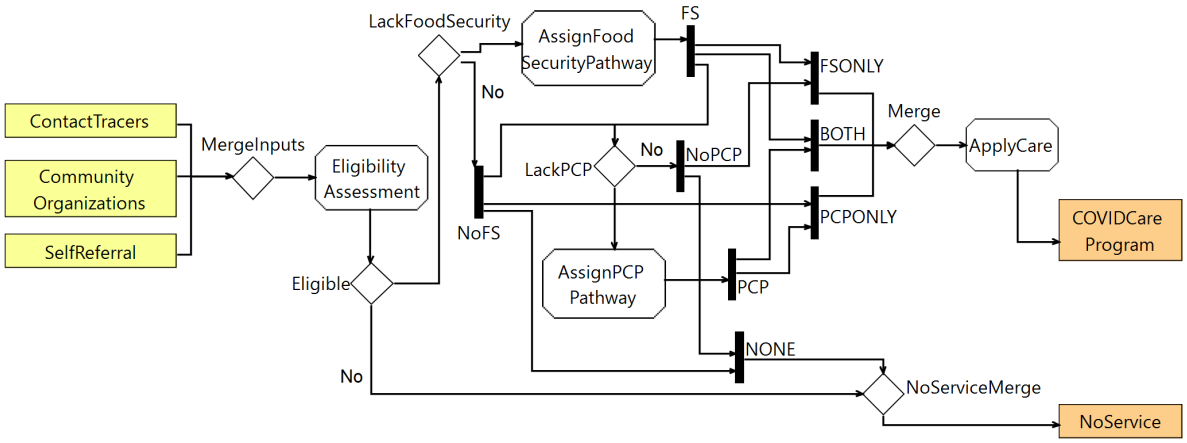
Figure 5: Continuum of Care (CoC) Model showing generic disease management stages with bifurcation into two paths that enable randomized control testing of interventions. Legend: rectangles represent stages of care, triangles represent hand-offs between stages where loss of patents occurs from one to the next.

contain eight *SYNC* node that corresponds to either fork node, join, or both, in the activity diagram. The alternative described in Figure 6a has one more node to indicate that both assessments take place concurrently. If the patient is admitted to a food security program by the decision node *LackFoodSecurity*, they will go through the assignment action, then the flow will go through the fork node *FS*. Since the patient may or may not be granted a physician care program (PCP), we account for that by dispatching through two different flows, one for the *FSONLY* join node in the case of which the patient has been denied PCP. Another outgoing flow to *BOTH* join node in the case of which the patient has been granted a PCP. Suppose the patient has been denied a food security program. In that case, the flow goes through *NoFS* fork node by which two outgoing flows are produced, one to the join node where the patient will be admitted to PCP care only *(PCPONLY)*, and another flow to the join node where the patient will be denied both *(NONE)*. In the same vein, the subsequent flow of the *LackPCP* decision node will take place accordingly. In other words, the *FSONLY SYNC* node will be enabled if the patient has been accepted to a food security program but denied a PCP. The *BOTH SYNC* node will be activated if the patient has been accepted in both programs. *PCPONLY SYNC* node will be activated if the patient has been denied a food security program but accepted to a PCP program. The *NONE SYNC* node will be activated if the patient has been denied both. We define a mechanism to take care of the excessive generated flows and ensure that each *SYNC* node is enabled correctly. The discussion of this mechanism is outside of the scope of this paper.

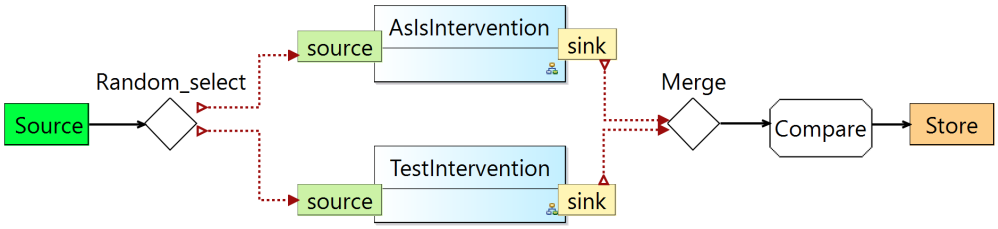
We also devise an alternative scenario which represents another flow where the two eligibility assessments takes place sequentially. The food security assessment takes place before the PCP assignment assessment. There are two reasons for devising such an alternative, besides the domain-specific purpose. The first reason is to describe how different flow alternatives can be devised by manipulating the control elements. The other reason is to demonstrate the underlying parallel notion benefited by employing the PDEVS formalism. The potential visual clutter can be avoided by employing hierarchical construction (Alshareef and Sarjoughian 2021) that will be illustrated in the bifurcation model to be discussed.

4.2 The Code Generation and the Experiment Results

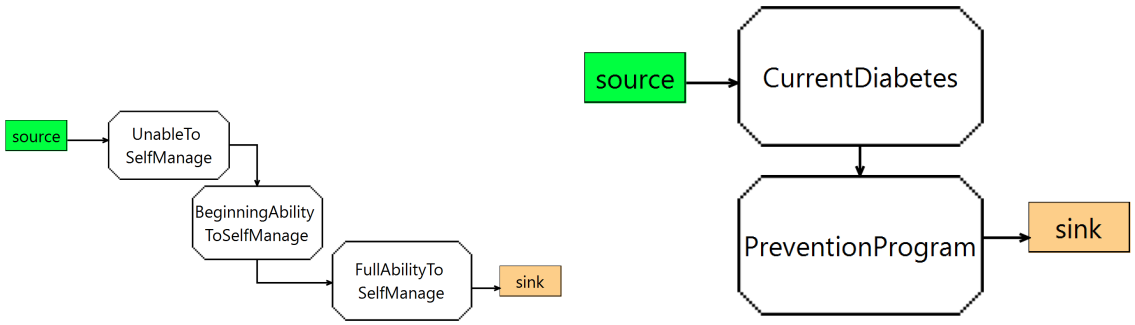
After devising all activity scenarios, the code generation step automatically results in sets of models, one set for each scenario. Each set contains an experimental frame consisting mainly of a generator (Markov-based generator for feeding the model through different external input coupling interchangeably). The other major component in the set is a coupled model corresponding to the whole activity. Each activity element has a DEVS counterpart, such as an atomic model, coupling, or I/O and ports. The models are given an initial probabilistic parameterization for the corresponding models dynamically based on the structural variability of the flow, such as the number of outgoing flows from a *SELECT* node or the number of input parameters of the activity. Those initial parameterizations, such as the probability distribution, can be modified and



(a) An application for the patient flow is modeled for primary healthcare example where both the lack of food security and the primary care physician are carried out in parallel. Hence *fork* node.



(b) Determining the intervention based on random select.



(c) *Test Intervention* and *As Is* activities that reside at the second level within the activity shown in Figure 5b

Figure 6: The flow alternatives are for the patient journey and intervention in the health care system modeled as activities.

adjusted in MS4 Me environment simply by changing the values in the generated files as needed. An initial legitimate DEVS model is obtained automatically without those changes.

4.3 Model of Bifurcation to enable Randomized Control Testing of Interventions

Figure 6b illustrates a modification of the CoC model in which after initial screening for healthcare access and diabetes issues, the flow of patients is bifurcated into two arms as earlier described. The upper arm models a conventional approach in which Andersen et al. (2020) examined diabetes management outcomes with type2 diabetes in a remote monitoring and coaching program. The study used patient activation as an effective way to foster patients" self-management of their diseases and facilitate self-monitoring skills (Hibbard and Greene 2013; Barello et al. 2014). In contrast, the intervention arm is distinguished in having three distinct stages based on measurement of increasing levels of patient activation and coached by community health workers (CHWs) to increase these skills. Since the Anderson protocol only measured these levels at the beginning and end of the study they could not affect patient progress along this dimension. Our intent is to emulate Anderson in essential respects save for this more intensive coaching, thereby enabling potential improvement in outcomes.

In relation to activity-based M&S, the Figure 6b illustrates the use of hierarchical construction in model development. At the top level, we emphasize the randomization of patients into the As Is and Test Intervention arms using the SELECT primitive and the subsequent merging of these streams to enable comparison of results. The second level elaboration of the two top-level interventions illustrates the capability of hierarchical construction to hide complexity at the top level while exposing it in consistent fashion at the next level down. Notice, that although specified at the high level of activity modeling, the subsequent elaboration enables final implementation of this protocol by integrated real-time data collection technology. The utility of activity modeling is demonstrated in such implementation which represents a continuous ongoing realization of the randomized control trial protocol that is usually set up on a once-per-study basis.

5 STRENGTHS AND LIMITATIONS

Integration of activity-based M&S into the MS4 Me environment demonstrates that support can be developed to link high level architecture products and downstream models and simulators. Following practices of metamodeling, model transformation, and code generation, we implemented a capability to create activity diagrams and to map them into executable activity-based models in the DEVS Markov formalism. This enables formalization of the flow selection process and state transition in the activity diagram and parameterized specification of stochastic behavior when desired. The automated code generation supports fast development and simulation activity-based models. This is very helpful for high-level exploration of workflow architectures and illustration of system operation for customers and other stakeholders. However, the mapped DEVS models in MS4 Me still require a multitude of parameter values to be set to achieve detailed desired behavior, although they are exposed to enable such adjustment. Moreover, changes in such models may put them outside the scope of activity-based specification, making "round-trip engineering" difficult. Another limitation is the workflow experimental frame to which the current activity-based formalism is intended. The current targeted messages and actions have to be enhanced to enable operations that generate the behavior required in other experimental frames. For example, currently the Compare action in Figure 6b cannot actually perform comparison of intervention outcomes. Finally, the targeted atomic models cannot easily be used in compositions with other DEVS models. To fully support such reuse would require also generating SES and DNL files that then merge with other DEVS models. Research is needed to understand whether and how activity diagram specification can be extended to enable such generation of SES and DNL documents to increase model reusability.

6 CONCLUSIONS AND FURTHER RESEARCH

Current MBSE calls for formalized models to replace documents as the fundamental building blocks of systems engineering. However, practicality demands that such models eventually support all the activities typically associated with the simulation discipline. Current MBSE formalisms stop well short of this capability (Alshareef and Sarjoughian 2017). One approach to bridging this gap is to enable mappings to be defined that precisely specify simulation models that realize their behaviors. Taken to practical limits, this approach entails building more capability into such a formalism so that it eventually replicates all capabilities associated with traditional simulation methodology. Although there are attempts to achieve this goal, our discussion above shows that there are fundamental reasons why it is not attainable. As demonstrated by Zeigler et al. (2021) one approach is to tie models at different levels of abstraction with informal but well documented links. Another approach is develop formalized links such as homomorphic mappings as successive levels of refinements are developed. Much further research is needed into these approaches to develop tools that are both convenient to use as well as theoretically well-founded.

REFERENCES

Acceleo 2021. "Generate anything from any EMF model". Available at https://www.eclipse.org/acceleo/ (Accessed May 13, 2021).

ACIMS 2019. "DEVS-Suite Simulator version 5.0.0". Available at https://acims.asu.edu/software/devs-suite/ (Accessed April 1, 2021).

Alshareef, A. 2019. "Activity Specification for Time-based Discrete Event Simulation Models". *Ph.D. Dissertation, Arizona State University*.

Alshareef, A., and H. S. Sarjoughian. 2017. "DEVS specification for modeling and simulation of the UML activities". In *Proceedings of the Symposium on Model-driven Approaches for Simulation Engineering*.

Alshareef, A., and H. S. Sarjoughian. 2018. "Parallelism semantics in modeling activities". In *Proceedings of the 4th ACM International Conference of Computing for Engineering and Sciences*.

Alshareef, A., and H. S. Sarjoughian. 2021. "Hierarchical Activity-Based Models for Control Flows in Parallel Discrete Event System Specification Simulation Models". *IEEE Access* 9:80970–80985.

Andersen, J. A., D. Scoggins, T. Michaud, N. Wan, M. Wen, and D. Su. 2020. "Racial Disparities in Diabetes Management Outcomes: Evidence from a Remote Patient Monitoring Program for Type 2 Diabetic Patients". *Telemedicine and e-Health*.

Barello, S., G. Graffigna, E. Vegni, and A. C. Bosio. 2014. "The challenges of conceptualizing patient engagement in health care: a lexicographic literature review". *Journal of Participatory Medicine* 6(11):259–267.

Beery, P. T. 2016. "A model-based systems engineering methodology for employing architecture in system analysis: developing simulation models using systems modeling language products to link architecture and analysis". Technical report, The Naval Postgraduate School in Monterey, California, United States.

Chami, M., and J.-M. Bruel. 2018. "A survey on MBSE adoption challenges".

EMF 2021. "Eclipse Modeling Framework". Available at https://www.eclipse.org/modeling/emf/ (Accessed May 13, 2021).

Hibbard, J. H., and J. Greene. 2013. "What the evidence shows about patient activation: better health outcomes and care experiences; fewer data on costs". *Health affairs* 32(2):207–214.

JDT 2021. "Eclipse Java development tools". Available at https://www.eclipse.org/jdt/ (Accessed May 13, 2021).

Jfreechart 2021. "Java chart library". Available at https://www.jfree.org/jfreechart/ (Accessed May 13, 2021).

MS4 Systems 2018. "MS4 Me Simulator version 3.0". Available at http://ms4systems.com/pages/ms4me.php (Accessed April 1, 2021).

PDE 2021. "Eclipse for Plug-in and RCP Developers". Available at https://www.eclipse.org/callisto/plugin-dev.php (Accessed May 13, 2021).

Seo, C., B. P. Zeigler, and D. Kim. 2018. "DEVS markov modeling and simulation: formal definition and implementation". In *Proceedings of the 4th ACM Inter. Conference of for Engineering and Sciences*.

Sirius 2021. "The easiest way to get your own modeling tool". Available at https://www.eclipse.org/sirius/ (Accessed May 13, 2021).

Wach, P., B. P. Zeigler, and A. Salado. 2021. "Conjoining Wymore's Systems Theoretic Framework and the DEVS Modeling Formalism: Toward Scientific Foundations for MBSE". *Applied Sciences* 11(11).

Xpand 2021. "Code generation based on EMF models". Available at https://projects.eclipse.org/projects/modeling.m2t.xpand (Accessed May 13, 2021).

Xtend 2021. "JAVA with SPICE. Eclipse Modeling Framework". Available at https://www.eclipse.org/xtend/ (Accessed May 13, 2021).

Xtext 2021. "Language engineering for everyone". Available at https://www.eclipse.org/Xtext/ (Accessed May 13, 2021).

Zeigler, B. P. 2016. "Discrete event system specification framework for self-improving healthcare service systems". *IEEE Systems Journal* 12(1):196–207.

Zeigler, B. P., A. Alshareef, M. J. Blas, M. Bonaventura, T. Paris, and A. Yacoub. 2021. *Using DEVS for full life cycle Model-Based System Engineering in Complex Network Design*. A chapter in Advances in Computing, Informatics, Networking and Cybersecurity, A Book Dedicated to M. Obaidat. Springer, NY, In Press.

Zeigler, B. P., S. Mittal, and M. K. Traore. 2018. "MBSE with/out Simulation: State of the Art and Way Forward". *Systems* 6(4):40.

Zeigler, B. P., A. Muzy, and E. Kofman. 2018. *Theory of modeling and simulation: discrete event and iterative system computational foundations*. Academic press.

AUTHOR BIOGRAPHIES

ABDURRAHMAN ALSHAREEF is an Assistant Professor at the College of Computer and Information Sciences, King Saud University, and a member of Arizona Center for Integrative Modeling & Simulation (ACIMS). He is also a consultant at RTSync Corp. His research interests lie in modeling and simulation, metamodeling, and activity-based modeling. His email address is ashareef@ksu.edu.sa.

CHUNGMAN SEO is a senior research engineer at RTSync Corp. and a member of Arizona Center for Integrative Modeling & Simulation (ACIMS). He received his Ph.D. in Electrical and Computer Engineering from the University of Arizona in 2009. His research includes Model Based System Engineering, DEVS based testing environment with DEVS inverse modeling, DEVS interoperability for Collaborative Development. His email address is cseo@rtsync.com.

ANTHONY KIM is a software engineer intern at RTSync Corp., who is a junior at Basis Ahwatukee. He has been involved in the testing and evaluation of MS4 Me software and technical support of User Experience. His email address is akim@rtsync.com.

BERNARD ZEIGLER is a Professor Emeritus from the University of Arizona and Chief Scientist at RTSync Corp. His biography appears in Wikipedia. His email address is zeigler@rtsync.com.