An Object-Event Simulation Approach to Simulation of ArchiMate Models

MELLE PLOEG, University of Twente, The Netherlands

Simulation is a frequently used tool in enterprise management, with Object-Event Simulation (OES), a discrete-event simulation paradigm, being one option. However, in current implementations, simulation models in OES have to be programmed in JavaScript. ArchiMate is another common tool in enterprise management, with applications in risk modelling. This paper proposes a mapping of ArchiMate models to OES models, which allows for the automatic simulation of ArchiMate models. In an attempt to demonstrate the viability of the proposed mapping, a proof of concept has also been created.

Additional Key Words and Phrases: Discrete-event simulation, Object-Event Simulation, ArchiMate

1 INTRODUCTION

Simulation is a useful tool in many fields, enterprise management among them. Discrete-event simulation (DES) is a frequently used approach for this, with many different applications and implementations [6, 9]. Discrete-event simulations are based on simulations as a series of events, rather than simulations as a continuous progression of time. With this generic definition, there are is a multitude of DES paradigms [13]. This paper will use the paradigm of Object-Event Simulation (OES). OES models allow modelling of both processes and information, which lets users create a great variety of simulation models [13]. OES is unique in separating its models into three separate components: objects, events, and event rules. This separation makes creating a mapping easier, as a mapping can draw from the required data of each component individually, as opposed to extracting all required data at once. OES is explained further in subsection 2.2.

Another component of enterprise management is modelling. This allows people to visualise and communicate business structures and processes. One use for this is to communicate risks, in order to find solutions that mitigate these risks [11]. A very frequently used modelling tool in enterprise management is ArchiMate. When extended with the Risk and Security Overlay (RSO) [7], it allows detailed modelling of enterprises, processes and associated risks.

Enterprise modelling and simulation are two useful tools on their own, but they are even more useful when they can be connected. The obvious way to do this would be by letting a user easily simulate an enterprise model. The only other existing solution for this is, to the best of our knowledge, a proposal called xArchiMate[5], which uses a different approach to discrete-event simulation. The solution proposed in this paper uses OES, with its simulation model structure allowing for easier creation of a mapping. Additionally, current implementations of OES make it quite hard to build simulation models without technical knowledge, because they have to be programmed manually.

TScIT 41, July 5, 2024, Enschede, The Netherlands

There are two challenges to being able to simulate, that is perform formal reasoning on, an ontological modelling language such as ArchiMate [3]. The first, as mentioned by Grov et al. [3], is how to model risk in such a way that the model provides enough detail to allow for automated formal reasoning, whilst still being simple enough to be readable to humans. This same challenge applies to more general simulation modelling. This challenge has been addressed with the RSO [7], as this more detailed specification adds components that can model additional information. However, as was done for xArchiMate[5], additional specifications for ArchiMate have been created, because simulation models require a greater level of detail than most enterprise models. These specifications are outlined in the proposed solution.

The second challenge as explained by Grov et al. [3] is that of automating support for reasoning about a model, which is the primary focus of this paper. In our case, this means creating a mapping from enterprise models in ArchiMate to OES models, which can then be ran as simulations. This mapping is the primary focus of this research.

1.1 Research Questions

The central challenge of this research is creating a connection between enterprise modelling in ArchiMate and Object-Event Simulation, as a means of allowing for automated support for simulation of ArchiMate models. This leads us to the primary research question.

RQ How can automated support for Object-Event Simulation of enterprise models in ArchiMate be created?

A major component of the answer to this research question is a mapping of ArchiMate to OES, the effectiveness of which must also be studied. To illustrate this, there are two sub-research questions.

Sub-RQ 1 How can ArchiMate elements be mapped to components of an Object-Event Simulation model?

Sub-RQ 2 What is the effectiveness and appropriateness of this mapping of ArchiMate to Object-Event Simulation?

The first question will be answered through the creation of a mapping of ArchiMate to OES. The second will be answered through the creation of a proof of concept, created using a JavaScript implementation of OES and the mapping of ArchiMate to OES as proposed in this paper.

2 BACKGROUND

2.1 ArchiMate

ArchiMate is an enterprise modelling language developed by The Open Group, created to design and communicate business processes, IT systems, organisational structures, and more, in an easily understandable format[2].

2.1.1 Risk and Security Overlay. While ArchiMate is versatile, it is lacking when it comes to modelling risk and security aspects. The Risk and Security Overlay (RSO) mitigates this [7]. It expands

on the standard ArchiMate specification by adding elements that allow a user to specify aspects of risk and risk management such as threats, vulnerabilities, loss events, and much more. It also adds the concept of assigning likelihoods to events, which is used in the solution proposed in this paper.

2.2 Object Event Simulation

Object-Event Simulation (OES) is a discrete-event simulation paradigm using object-oriented modelling [1]. With this, OES unites information systems engineering and discrete event simulation [13].

To create an OES system which can be executed, an object event model is required. This consists of the following, as explained by professor Wagner:

- A set of object types, which define the information structure of the system. Objects can be modelled with a number of variables to describe their attributes.
- A set of event types, which model different events that can occur. These events can affect objects or be the cause of other events, depending on the event rules.
- A set of event rules, one for each event. Event rules describe the way an event affects the objects in the model, as well as what other events are caused by the event.

A model like this, combined with an initial state, consisting of initial object states and initial events, defines an OES system[13]. An OES system state consists of a current system state, given by the current values of all objects in the system, and a list of planned events called the future events list (FEL). Event occurrences trigger event rules, which changes the state of the system by changing affected objects and the FEL. A detailed diagram modelling the information architecture can be found on GitHub[14].

2.2.1 OES Implementation in JavaScript. Prof. Wagner has created a JavaScript implementation of OES [14] called OESjs, which has been used as the target for the mapping created for this paper. It is also the OES implementation used in creating a prototype.

The basic OESjs implementation uses classes to define the object event model. There is an “oBJECT" class, the implementations of which model object types, and an “eVENT" class whose implementations model event types and event rules, through an “onEvent()" method. Every model also includes a script that models the initial state, as well as other simulation attributes like the time limit of the simulation.

3 RELATED WORK

xArchiMate is an existing solution for the execution of ArcihMate models[5]. The key difference between xArchiMate and the solution proposed in this paper, is the chosen discrete-event simulation method: xArchiMate’s uses the concept of Open Objects, which are objects that have methods as well as attributes. As a consequence, any object can model both the state of a system and an event acting upon that system. This is in contrast to OES, which, in the implementation used in this paper, strictly separates state-representing objects and state-modifying events. The Open Object approach allows for the creation of more detailed models, but in turn may increase complexity. [5] mention that, regardless of level of detail, enterprise architecture models are going to be complex.

The Discrete-Event Modelling Ontology (DeMo) is an ontology made for ontology-driven simulation, that is an ontology designed such that its models can be automatically simulated [10]. This is used in a program called DeMOForge, which maps domain ontologies to DeMo, which can then be transformed to a discrete-event simulation model. Silver et al. [10] note a key difference between mapping between domain ontologies and mapping from a domain ontology to a modelling ontology. The former usually is used to establish relationships between concepts, for example of equivalence. In the latter, concepts are connected as analogs. In our case of using ArchiMate, this could be mapping a “Resource" in ArchiMate to an Object in Object-Event Simulation, meaning the Resource will be treated as functioning as an Object. We will be using this approach to mapping in this paper, only to map ArchiMate to Object-Event Simulation models.

Professor Wagner, the creator of Discrete-Event Simulation (DES), has also done work on ontological foundations for DES [15]. Because DES requires both information and event/process modelling, there is no singular existing ontology that can model OES models [4]. As such, a method using UML class diagrams for the information component and a modified version of BPMN, called DPMN, for the events and event rules. BPMN is a business-oriented process modelling language. DPMN is more explicit than BPMN, and as such is complete enough to be simulated. However, DPMN is no longer supported, so it will not be used in this paper [12]. However, the use of a combination of UML and DPMN does demonstrate the requirements of an ontological simulation model: it needs to model both information and processes. These requirements are important for the solution proposed later in this paper.

4 PROPOSED SOLUTION

ArchiMate, as it is normally used, is intended to create clear and easily readable diagrams, which can be understood by people without technical background knowledge[2]. As such, they do not include enough detailed information to be simulated. To mitigate this, the solution as proposed here sets requirements for ArchiMate diagrams used as its input. Some of these, like the “Likelihood" element, are taken straight out of RSO, whereas some, like the “Initial" element, are newfound but defined in a similar fashion. These additional specifications were created with the intention of maintaining the readability of input diagrams. There are two examples of complete, runnable models in the appendix: Figure 3, depicting a bank account with regular transactions, and Figure 4, depicting an investment portfolio during a tumultuous time in the market.

4.1 Used Tools

To create the ArchiMate diagrams used in this paper, “Archi"[1] was used. As a reference for the proposed mapping, the JavaScript implementation of OES, “OESjs"[2], was used, as explained in subsection 2.2. For the implementation of the prototype, “ANTLR"[3], a language

[1] https://www.archimatetool.com/
[2] https://github.com/gwagner57/oes
[3] https://www.antlr.org/

recognition tool for Java, is used to parse the ArchiMate input diagrams, and “OESjs" is again used, to run the generated simulation models.

4.2 Requirements

4.2.1 Ontological Model. The capability for simulation of an ArchiMate model is important, but it is not the model’s only requirement. An input model should not only contain all the required information for a simulation, but also be able to hold additional information that may be relevant to a reader. This would entail information that either cannot be mapped to OES or is not relevant to OES. Besides that, the information that is used in the simulated component of the model should be readable to people without technical knowledge. The technical information shown in the model should therefore also be kept to a minimum.

4.2.2 Simulation Model. As explained in subsection 2.2, an OES model consists of the union of three sets, in addition to some simulation settings: a set of objects, one of events, and one of event rules. Elements of an ontological model should then be mapped to one of these three components, and these three components need to be appropriately represented in an ontological model.

4.3 Method for Creation of Proposed Solution

The mapping is based on three components: objects, events, and relations. Objects represent the information modelling component of the model have associated attributes that can be modified to change the state of the simulation. For example, a bank account may be an object and its balance can be a property. Objects can easily be modelled using UML class diagram.

Events are the processing component of the model. They are mapped to event classes in OES, and serve as a way to affect objects and trigger other events through event rules.

Relations are used in two ways. First is to associate elements with each other, for example to associate an object with an attribute. Second is to represent the event rule component of OES models, in the form of event triggering and object manipulation.

Because ArchiMate models normally do not contain enough information to be simulated[5], some additional information has to be added. This comes in the form of the purple elements, as seen in the example diagrams. These elements add important simulation information, but may also show information that could be useful to a reader.

Certain ArchiMate elements were mapped to represent either object or event classes in OES. These elements were selected based on the ArchiMate 3.2 specification[2]. The selection was made based on whether or not the element in question could reasonably be interpreted as either a kind of object or a kind of event. This was done to allow for flexibility in the creation of input diagrams, compared to a stricter approach of, for example, only interpreting ArchiMate “Event" elements as OES events.

4.4 Object Modelling

Objects in OES represent anything that is affected by events. Therefore, there are many ArchiMate elements that can be treated as objects. The full list is the following:

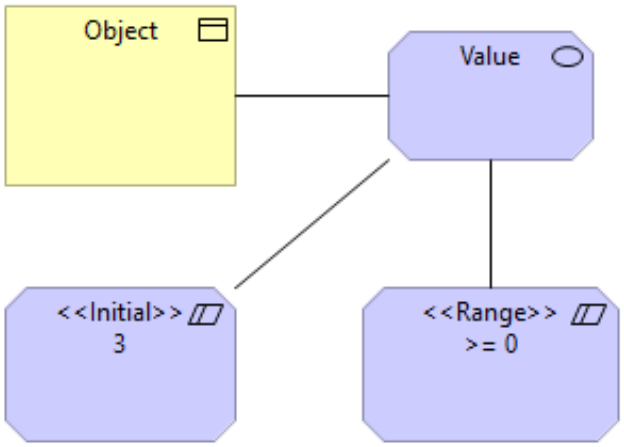

Fig. 1. ArchiMate diagram showing some arbitrary object with one associated attribute. The range specified is optional.

- Resource
- Business Object
- Contract
- Product
- Data Object
- Node
- Device
- System Software
- Communication Network
- Equipment
- Facility
- Distribution Network

OES objects have a number of attributes, each with an initial value and optionally a restricted range. This is modelled in ArchiMate in the following way: an object has at least one associated property, represented as an ArchiMate value element. Each value has one or two associated constraints. The first is stereotyped with “Initial", and denotes the initial value of the property. The optional second is stereotyped with “Range", followed by some condition that must hold for the property. Figure 1 depicts a diagram in ArchiMate of some object.

4.5 Event Modelling

Events in OES are composed of event types and event rules. Event types are modelled in ArchiMate as any of the following elements:

- Any process
- Any function
- Any interaction
- Any event
- Any service

Event rules are of two types; those that affect objects, and thus change the state of the simulation, and those that trigger objects, adding to the future events list. Figure 2 depicts a small diagram with two events, demonstrating how different event rules may be modelled.

4.5.1 Affecting Objects. Events can affect objects by changing their attributes. An event that is modelled as having an “association" relationship with an object in ArchiMate is treated as affecting that object. Changing a property of that object is modelled with

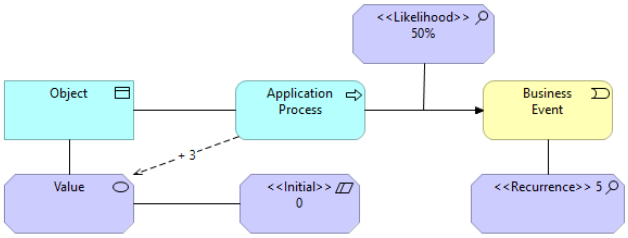

Fig. 2. ArchiMate diagram showing some arbitrary event. The “Application Process" is mapped as an OES event class, with the event rule that it changes the “Value" attribute of the blue “Object" by adding 3 to it. Additionally, it has a 50% chance to trigger the “Business Event", which will subsequently recur every 5 simulation turns if it is triggered.

an “influences" relationship between the event and the object’s property, represented as a “value" element. This relationship should be labelled with the mathematical operation to be performed when the event occurs. An example of this can be seen in Figure 2.

4.5.2 Triggering Events. An event being triggered means it being added to the future events list. Events can be triggered in three ways. First, they can be a start event of a simulation. Events that are not triggered by other events are considered to be start events, and are put on to the future events list at the start of the simulation.

The second way is by being triggered by other events, representing causal relationships. This is modelled by having a “triggering" relationship from one event to another. When the causing event occurs, the event on the receiving end of the relationship is put on to the future events list. Optionally, an “assessment" element, stereotyped with “Likelihood" followed by some percentage, can be associated with the relationship. This represents the chance of the triggering relationship happening.

An event can be modelled to trigger one of multiple events. For this purpose, an “or junction" is used. This junction has an incoming triggering relation from the causing event, and outgoing triggering relations to caused events. Each outgoing trigger should have a an associated likelihood (see random chance). If the sum of the likelihoods is less than 100%, the remaining chance is the chance of no new event being triggered.

Thirdly, events can be triggered exogenously, that is by causes from outside the simulation, through recurrence. This is modelled by associating the event with an assessment element, stereotyped with “Recurrence", followed by either one or two numbers. If there is just one number, the event will reoccur once in that many simulation turns. If there are two numbers, then the time between recurrence will be picked randomly between the two numbers. Note that for an event to start recurring, it must first occur, either through being a start event or through being triggered a first time.

4.6 Time Limit

OES models with recurrent events should have a time limit, since otherwise the simulation will go on infinitely. This time limit is not modelled in the ArchiMate model, in order to keep it more readable. Therefore, an implementation of this mapping should acquire a time limit in some other way, for example by asking the user when the simulation is ran.

4.7 Non-Simulation Modelling

As ArchiMate is an enterprise modelling language[2], diagrams made in it should be understandable for those without technical knowledge. To help with this, the proposed solution in this paper tries to minimise the technical detail in its diagrams. For example, the type of an attribute is not explicitly stated, but rather inferred from the initial value. Additionally, diagrams that can be simulated should be able to model more than just the simulation. As such, elements that are not in the mapping or that are not “correctly" used, such that they can be simulated, should be ignored for the simulation. In this way, diagrams can serve multiple purposes; as a visual representation of some business, and as a simulation of that business.

5 PROOF OF CONCEPT

In an attempt to show the viability of the proposed solution, a proof of concept has been created [8]. This fairly straightforward implementation generates a full OESjs model from an ArchiMate diagram in the form of a CSV file. Note that this is a simple prototype only intended to show the viability of the created mapping. Therefore, it does not implement all features, such as ignoring modelling components irrelevant to the simulation, or the use of restricted ranges for attributes.

Given a syntactically correct input diagram, the model can generate an appropriate, executable, and correct OES model. Though input diagrams still require some fairly specific syntax for the sake of completeness, this may be easier for those without programming knowledge, compared to programming an OES in JavaScript. The prototype is not capable of ignoring irrelevant ArchiMate elements in its input diagram. The ArchiMate diagrams shown in the appendix are both fully capable of being automatically converted to functioning OESjs models.

5.1 Example Execution

To illustrate the process by which ArchiMate diagrams may be converted to OES models, here is an example of how that works for the bank account diagram depicted in Figure 3 in the prototype implementation created for this paper. A JavaScript OES model consists of a number of object and event classes, as well as a class to setup the simulation.

The “Bank Account" element, as a business object, converted to an object class, with an attribute “Balance" which is set to 0 in the class constructor. The three business event elements, “Salary", “Rent", and “Go To The Pub", are each converted to an event class. Becuase they are each associated with the “Bank Account" object, they each receive that object in their constructor. With this, the set of objects and set of events have been programmed. As explained in subsection 2.2, all that is then left is the set of event rules, as well as an initial state for the simulation.

Each event class has an “onEvent()" method that is called when the event is triggered. This method contains the code to execute the event rules associated with the event in question. The “influences"

relations in the diagram (arrows with dotted line) are converted to event rules that affect the state of an object, in this case the balance of the bank account. The implementation of this is rather simple, as, for example, the “Rent" event class simply subtracts 80 from the “Balance" attribute of the “Bank Account" object it was passed when the event class was constructed.

The “onEvent()" method, in addition to containing code, also returns a list of events triggered by the event the method is in. In this case, that would be “Rent" triggered by “Salary". In the case of an associated “Likelihood", this only happens by a certain chance, in this case 20%. Events that are recurrent have a “createNextEvent()" method in OESjs, which is called when the event has occurred to put the next occurrence of this event on the future event list, with a certain delay. In this case, only “Salary" is recurrent, and so whenever it occurs another instance of it gets put on the future events list with a delay of 5. Additionally, because it does not have any incoming triggering relations, it is also interpreted as a start event, meaning that when the simulation starts it is put on the future events list. This happens in a separate script that also initialises the simulation’s objects.

6 DISCUSSION

6.1 Results

The proposed mapping, as described above, is complete enough to allow for the creation of basic simulation scenarios in ArchiMate. The simulation features it implements are all contained in OES core 0Wagner [14], which is the most basic OES implementation. Input diagrams require less detailing and specification compared to xArchiMate[5], whilst not hiding any information from the user. The input specification also does not require any new ArchiMate elements or modification to the ArchiMate language, because it reuses standard ArchiMate elements. An unexpected outcome of the requirements is that the technical information that is contained in input diagrams, is all in the “Motivation Layer" (purple elements). As a result, all these motivation layer elements can be hidden to create a “Cleaner", more readable diagram, albeit one that shows less detail.

6.2 Limitations of Proposed Solution

The mapping proposed in this paper provides a basic set of features, but is still quite limited in its uses on three aspects. First, not every element in ArchiMate is mapped to OES. This means that when a diagram containing unmapped elements is simulated, it is not simulated in its entirety, which may have implications for the outcome one might expect. Second, not every simulation mechanic a user may require is mapped to. This ranges from fairly straightforward features such as conditional triggering relations, to more complex ones such as agent based simulation[14]. Finally, is a problem inherent with creating ontological models that are both complete and readable. OES models require precise simulation data, whereas ArchiMate diagrams are preferably made to be quite generic, with the specification of the ArchiMate language being as small as possible[5]. As a result, any mapping has to compromise on both ends: some features in OES are not mapped to in order to keep the input diagrams simpler, but on the other hand input diagrams require some quite specific syntax in order to be complete enough to be simulated, which increases their complexity.

6.3 Future Research

This research has demonstrated the possibility of automatic simulation of ArchiMate diagrams, however, it is not complete enough for practical use. It may be commercially interesting to do more research, as automatic simulation of commonly used diagrams might be useful to businesses.

Future work could expand on the mapping proposed in this solution, in order to allow for more detailed creation of simulation models. The challenge lies in extracting a as much information as possible out of diagrams that are as easy to read and “clean" as possible.

Allowing for more flexibility and requiring less strict syntax for input diagrams would be beneficial to users, as it would allow them to tailor diagrams they create to their own needs rather than to the automatic simulation mapping. Research could be done in to the use of AI for interpreting these more diverse models, given that said AI is well trained on interpreting the intention behind user made diagrams. An implementation of this could make for a commercially viable product, although extensive testing and research would have to be done on the correctness of an AI model made for this purpose.

7 CONCLUSION

The mapping proposed in this paper is grounded in the requirements of OES models, whilst attempting to let ArchiMate diagrams that can be simulated to be used as clear visual models. The proposed mapping does not use all elements of ArchiMate, nor does it map to every possible feature of OES. Further expansion could be the subject of future research. While the mapping is not as extensive as it could be, it still allows for the creation of simulation models that may be found useful in enterprise management. Therefore, it can be considered as a valid answer to the first sub research question.

The proof of concept created for this paper has shown that an implementation of this mapping is both possible and usable, even given its limited scope. The prototype does, however, also highlight the limits of the proposed mapping, as there are many components of ArchiMate that are not mapped to OES, and, conversely, there are simulation scenarios that one may want to create, but cannot be created with this mapping. The answer to the second research question, then, is that the proposed mapping is effective in the limited features it does implement, but there are many ways in which it could be improved in future research.

The answers to these sub research questions leads to a straightforward answer to the main research question: automated support for Object-Event Simulation of ArchiMate models can be created by designing and implementing a mapping of ArchiMate to OES. It would be possible to create a complete implementation of the mapping proposed in this paper, however we would recommend further research into expanding the mapping before committing to development of a complete product.

REFERENCES

[1] [n. d.]. GitHub - gwagner57/oes: Various simulators for Object Event Simulation (OES), which is a Discrete Event Simulation paradigm combining object-oriented

modeling with the simulation approach of event scheduling. https://github.com/gwagner57/oes?tab=readme-ov-file
[2] [n. d.]. The Open FAIR™ Body of Knowledge | www.opengroup.org. https://www.opengroup.org/open-fair
[3] Gudmund Grov, Federico Mancini, and Elsie Margrethe Staff Mestl. 2019. Challenges for risk and security modelling in enterprise architecture. *Lecture Notes in Business Information Processing* 369 (2019), 215–225. https://doi.org/10.1007/978-3-030-35151-9{_}14/FIGURES/3
[4] Giancarlo Guizzardi and Gerd Wagner. [n. d.]. DISPOSITIONS AND CAUSAL LAWS AS THE ONTOLOGICAL FOUNDATION OF TRANSITION RULES IN SIMULATION MODELS. ([n. d.]).
[5] Laura Manzur, Jorge Mario Ulloa, Mario Sánchez, and Jorge Villalobos. 2015. xArchiMate: Enterprise Architecture simulation, experimentation and analysis. *SIMULATION* 91, 3 (2015), 276–301. https://doi.org/10.1177/0037549715575188/FORMAT/EPUB
[6] Renan Moritz Varnier, Rodrigues Almeida, Rogério Pires Santos, Wagner Coelho, Albuquerque Pereira, | Renan, and Moritz Varnier. 2022. Discrete-event models for the simulation of computed tomography sectors according to hospital structural/organizational changes and expected patient arrival rates. *The International Journal of Health Planning and Management* 37, 1 (1 2022), 536–542. https://doi.org/10.1002/HPM.3335
[7] Ítalo Oliveira, Tiago Prince Sales, João A Paulo Almeida, Riccardo Baratella, Mattia Fumagalli, Giancarlo Guizzardi, Kurt Sandkuhl, Balbir Barn, Tony Clark, Souvik B Barat Ítalo Oliveira, and Giancarlo Guizzardi gguizzardi. [n. d.]. Software and Systems Modeling Ontology-based security modeling in ArchiMate. ([n. d.]). https://doi.org/10.1007/s10270-024-01149-1
[8] Melle Ploeg. 2024. OES-in-ArchiMate. https://github.com/Melle-Ploeg/OES-in-ArchiMate
[9] Francesca Sala, Gianluca D'Urso, and Claudio Giardini. 2023. Discrete-event simulation study of a COVID-19 mass vaccination centre. *International journal of medical informatics* 170 (2 2023). https://doi.org/10.1016/J.IJMEDINF.2022.104940
[10] Gregory A. Silver, Kushel Rai Bellipady, John A. Miller, Krys J. Kochut, and William York. 2009. Supporting interoperability using the discrete-event modeling ontology (DeMO). *Proceedings - Winter Simulation Conference* (2009), 1399–1410. https://doi.org/10.1109/WSC.2009.5429288
[11] Bondarenko Svitlana, Bodenchuk Liliya, Krynytska Oksana, and Gayvoronska Inna. 2019. Modelling Instruments in Risk Management. *International Journal of Civil Engineering and Technology* 10, 01 (2019), 1561–1568. http://www.iaeme.com/IJCIET/index.asp1561http://www.iaeme.com/ijciet/issues.asp?JType=IJCIET&VType=10&IType=01http://www.iaeme.com/IJCIET/issues.asp?JType=IJCIET&VType=10&IType=01
[12] Gerd Wagner. [n. d.]. BUSINESS PROCESS MODELING AND SIMULATION WITH DPMN: PROCESSING ACTIVITIES. ([n. d.]).
[13] Gerd Wagner. 2022. *Object Event Modeling for DES and IS Engineering*. [Aachen, Germany]. https://ceur-ws.org/Vol-3211/CR_099.pdf
[14] Gerd Wagner. 2024. OESjs. https://github.com/gwagner57/oes
[15] Gerd Wagner, G Wagner@b, and Tu De. 2018. Information and Process Modeling for Simulation – Part I. *Journal of Simulation Engineering* 1 (2018), 1–1. https://doi.org/10.5281/ZENODO.11110129

A DIAGRAMS

Figure 3 and Figure 4 show full ArchiMate diagrams that can be simulated with the use of the mapping proposed in this paper.

B SUMMARY OF PROPOSAL

Table 1 shows a summary of the mapping of ArchiMate elements to OES. Elements not described in the table are not included in the mapping.

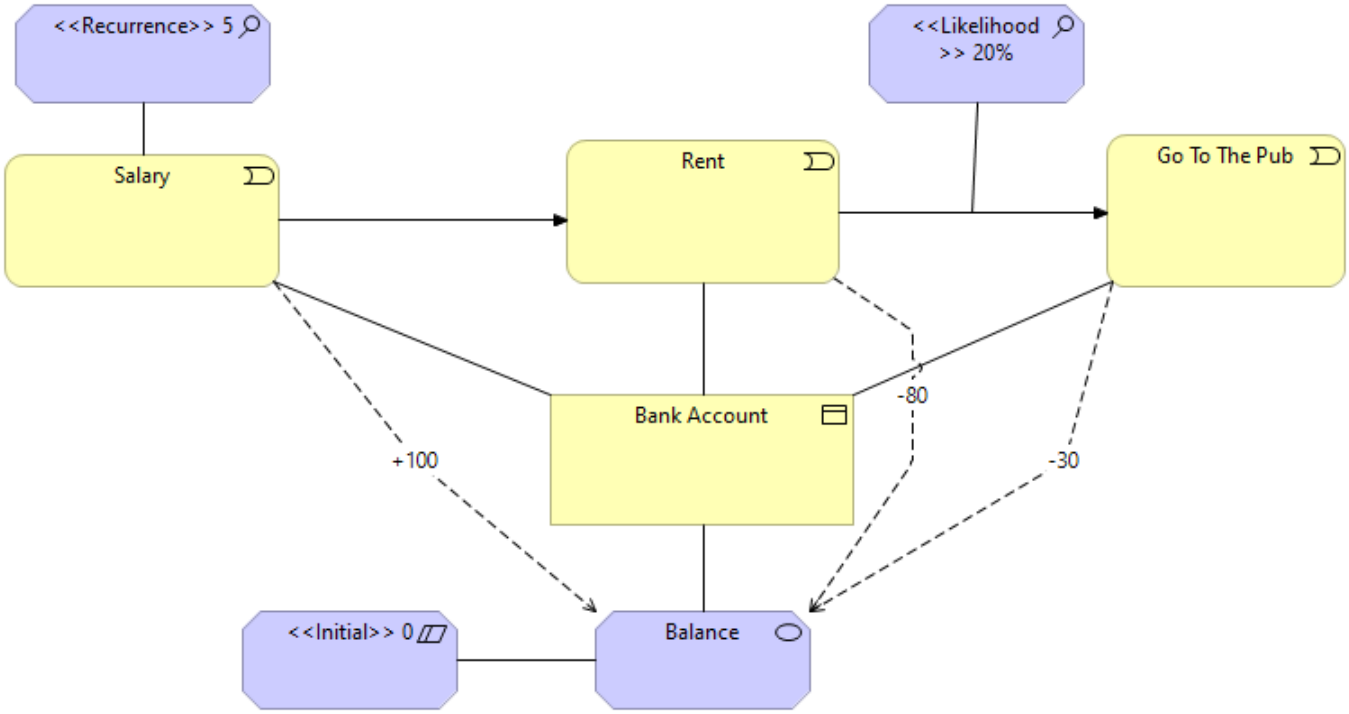

Fig. 3. ArchiMate diagram showing a simulation model where a salary gets added, followed by rent subtracted, and a potential of going to the pub.

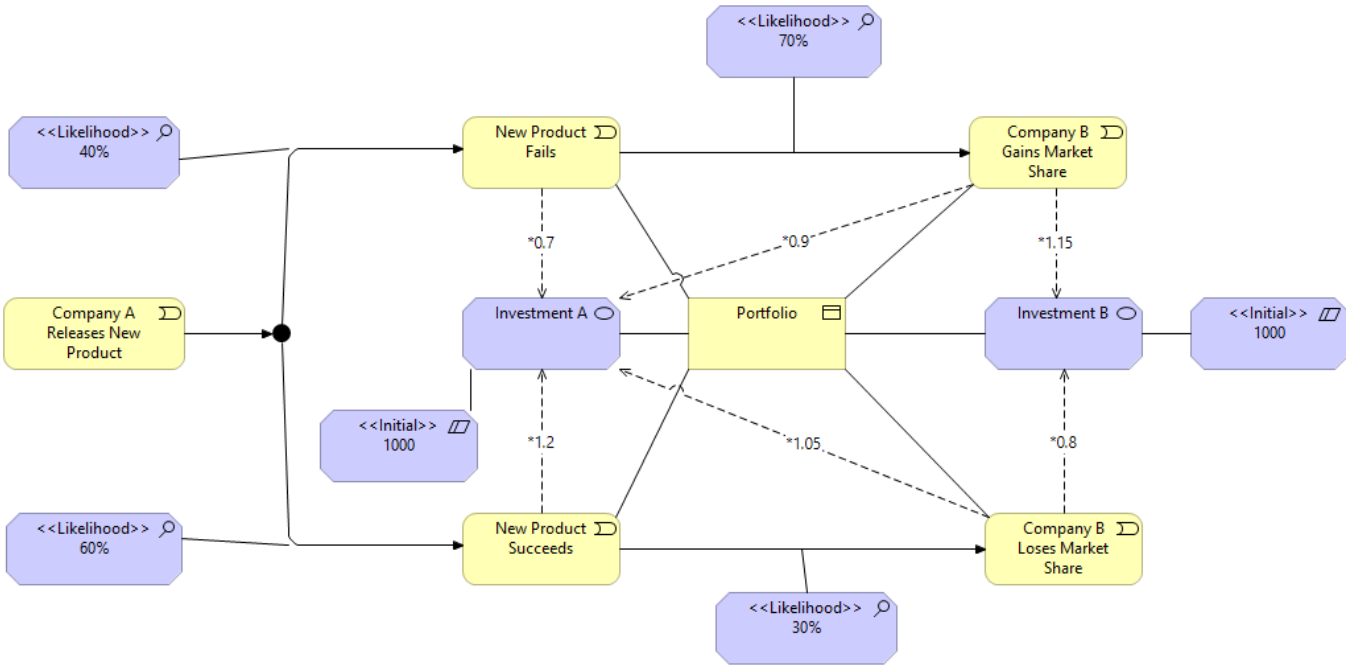

Fig. 4. ArchiMate diagram showing a simulation model where a company makes an investment, which either does or does not pay off. This then affects an investor's portfolio.

Table 1. Mapping of ArchiMate to OES

ArchiMate element	OES equivalent
Any from ArchiMate Object List 4.4	OES object classes
Any from ArchiMate Event List 4.5	OES event classes
Triggering relationships	Causal event rules
Influences relationships	State-changing event rules
Value elements	Object attributes
Initial stereotyped constraints	Attribute initial values
Recurrence stereotyped assessments	Event recurrences
Likelihood stereotyped assessments	Likelihoods of triggering relations