anylogic

MULTIMETHOD SIMULATION MODELING FOR BUSINESS APPLICATIONS

OVERVIEW WITH GUIDED MODEL BUILDING EXAMPLE

WHITE PAPER

Contents

Introduction

People frequently underestimate how often they use models. In fact, we build mental models every day. A mental model is your understanding of how things work in the real world: of friends, family, colleagues, the town where you live, the economy, politics, even of your own body. Decisions such as what to say to your child, what to eat for breakfast, and who to vote for are all based on mental models.

However, when tasks become more complicated, digital models can be superior to our mental models. This is why building models using computer software tools has become standard in business and engineering.

The most popular digital modeling approach is to use spreadsheets. Spreadsheet models are commonly used for financial analysis, investment research, and business planning. Using user-defined formulas, spreadsheets can perform mathematical, statistical, financial, and organizational transformations on sets of data. Spreadsheets also help to present data in an organized way that can help inform future planning decisions.

Building information modeling (BIM) is another good example that comes from the construction industry. Currently, before any construction work, a digital model of a building is created with tools like Autodesk® Revit®. Such models include all the architectural and engineering aspects of the future building. These models simplify the construction process, allowing the technical side to become transparent to those involved, including architects, building designers, construction professionals, and the people who will use this building in the future.

Why though, do people create models at all? First of all, to experiment without having to use real-life objects. Real-life experiments are often too expensive or even impossible. The virtual modeling world is completely risk-free. Managers, researchers, and engineers can create and test models of various system designs and answer hundreds of "what-if" questions, all by virtually experimenting in a risk-free environment.

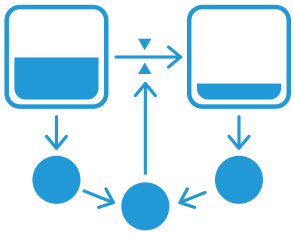
Simulation modeling stands apart within the range of modeling tools and technologies because it allows the introduction of dynamics into models. Every simulation model evolves over time, discretely or continuously changing its state. To create a simulation model means to define a set of rules that drive the system over time. Simulation modeling is a very powerful instrument that is widely used for solving numerous business challenges. This is mostly because it is very practical and easy to understand, no need guess if something is possible, just run a simulation model and see what happens. In addition, animation is a significant advantage that simulation modeling can provide over static spreadsheet models. When people see an animated model, similar to a movie or a computer game, where a system can be seen working as specified, it is easier to visually validate the model and trust its results.

In this paper, we will cover the different approaches to simulation modeling and define multimethod (also called hybrid) modeling. It should be noted that simulation modeling is widely used, even in specific areas such as engineering and scientific modeling, but in this white paper we are going to focus exclusively on business applications. Moving from theory to practice, we will build a multimethod simulation model example, analyze it, and find the solution to the most common issue of a novice modeler: how to choose the most appropriate modeling method or combination of methods to meet your challenge and purpose.

THREE METHODS IN SIMULATION MODELING

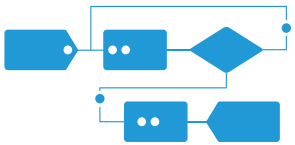
01

Method, in simulation modeling, means a general framework for mapping a real-world system to its model. A method suggests a type of language, or "terms and conditions" for model building. In modern simulation modeling for business systems, there are three main methods. Each of them serves a particular level of abstraction.



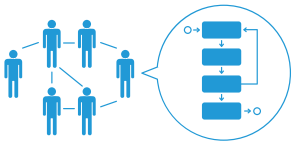
SYSTEM DYNAMICS (SD)

is the oldest simulation approach, with roots in work done by Massachusetts Institute of Technology's professor Jay Forrester in the 1950s. From the very beginning, system dynamics stayed focused on the management and organizational layers of business. This approach operates at a high abstraction level and is mostly used for strategic modeling. The basic system dynamics element is the stock and flow diagram. A stock is a digital representation of something that is modeled, e.g., a number of people, a set of requirements, or a number of products. Any process in system dynamics is modeled as flow between stocks. You can imagine that any production process is just a raw material flow to finished goods. System dynamics is most commonly used for modeling strategic management, marketing and macroeconomic issues, ecological and social systems.



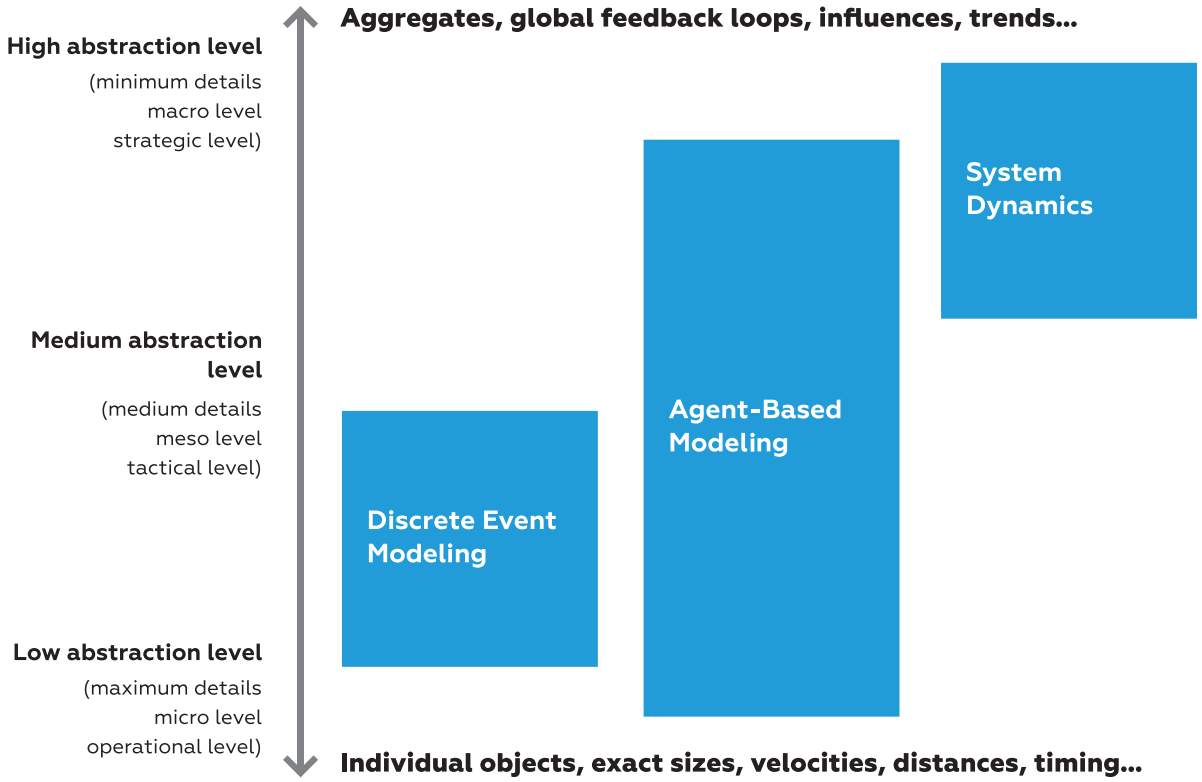
DISCRETE EVENT SIMULATION (DES)

with its underlying process-centric approach, supports medium and medium-low abstraction. Discrete event modeling is almost as old as system dynamics. In October 1961, IBM engineer Geoffrey Gordon introduced the first version of GPSS (General Purpose Simulation System, originally Gordon's Programmable Simulation System), which is considered to be the first software implementation of discrete event modeling. These days, discrete event modeling is supported by many software tools. From the practitioner's point of view, we should say that when people apply simulation, the model they build is discrete event or includes a discrete event subpart in more than 50% of the cases. The main idea of discrete event simulation is to consider a system as a process, i.e., a sequence of operations being performed across entities. A model is specified graphically as a process flowchart, where blocks represent operations. The flowchart usually begins with "source" blocks, that generate entities and inject them into the process, and ends with "sink" blocks that remove entities from the model. This type of diagram is familiar to the business world as a process diagram and is widely used to describe business processes. Discrete event simulation modeling is widely used in the manufacturing, service systems, and healthcare fields.



AGENT-BASED MODELING (ABM)

is the most recent of the major simulation methods. From 1990-2000, ABM stayed a purely academic topic, but the 21st century, with its boom in computer processing capability, made ABM commercially applicable for solving high-scale business tasks. Moreover, compared with the other modeling techniques, it is now showing the fastest growth in use. ABM uses a bottom up approach, where the system is described as interacting objects with their own behaviors. System behavior emerges as a summary of the individual actions of agents. Agent-based models can vary, from very detailed, where agents represent physical objects, to highly abstract, where agents are competing projects or assets. There are some specific ABM problems, such as population, pedestrian, road traffic, and epidemiology modeling. But in fact, ABM is used for modeling almost everything, from the markets, to supply chains and logistics, in cases where we need to focus on individual objects and describe their local behavior and interactions.



Applications of Simulation Modeling

All three modeling approaches are quite independent. Each has its own tool set, unique features, strengths, and weaknesses. The choice of method should be based on the system under study and the purpose of the modeling. The scope of simulation modeling applications can be seen at the picture to the left.

However, most real-world cases are too complex to be modeled with one method, and it is convenient to describe different parts of a system with different modeling approaches. A combination of approaches may provide the opportunity to create an accurate and multifunctional model of the system without any workarounds.

EXAMPLE: SUPPLY CHAIN MODEL

Consider, as an example, a supply chain in China. This supply chain consists of three manufacturing centers and fifteen distributors in different cities that regularly order product delivery. There is a fleet of trucks at each manufacturing facility. When a manufacturing facility receives an order from a distributor, the personnel check the number of products in the storage. If the required amount is available, the facility sends a loaded truck to the distributor. Otherwise, the order waits until the factory produces enough products. Once the truck with products arrive at the distributor, the products are unloaded and put in the storage. The unloaded truck then returns back to the manufacturing facility.

How can we build a model of this logistic network? To get started, the modeler should choose the most efficient way to address all aspects of these complex systems. If they are a beginner, the modeler will be faced with challenges even at this early stage.

Let us try to build the model of the supply chain. You will see that to make an elegant model that is easy to build, understand, and manage, a combination of all three modeling methods is required.

To build the model, we will use AnyLogic® simulation software – as of 2019, it remains the only professional simulation tool that supports all three modeling methods. You can run the model we built in AnyLogic Cloud, or download the free software version to practice yourself.

Run the Manufacturing & Logistics model>>

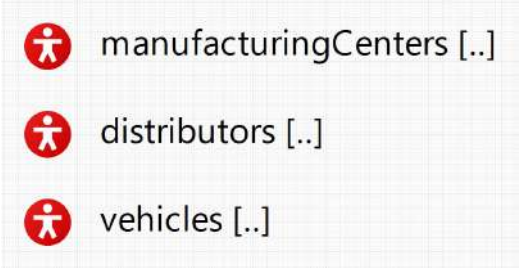
EXAMPLE: BUILDING A SUPPLY CHAIN SIMULATION MODEL

02

Supply Chain Structure and Logic – Agent-Based Modeling

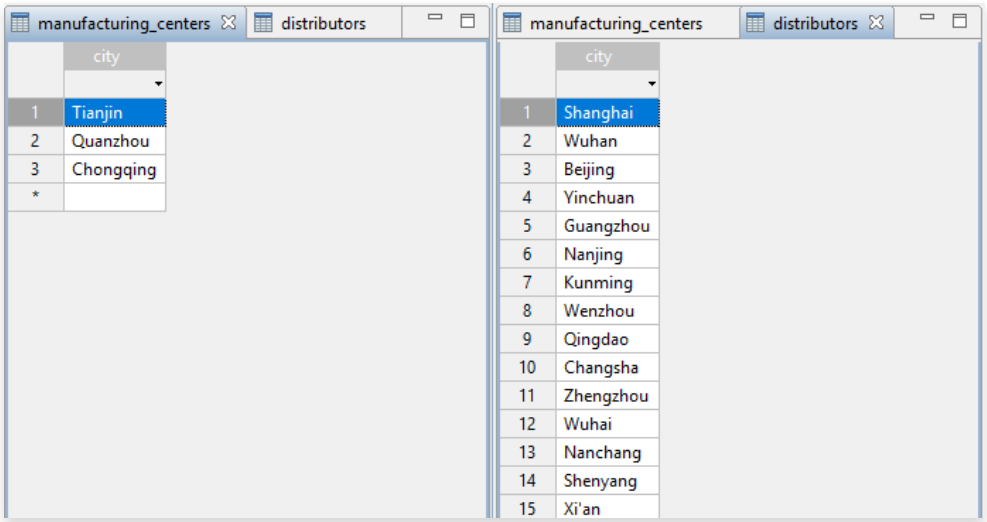
As described in the earlier example from China, the supply chain we will model includes several factories and distributors that are spread across the country, and multiple vehicles that deliver the product from the factories to distributors. To capture the supply chain structure and logic, it is convenient to use the agent-based modeling approach. Why is this approach the most appropriate in this case?

From the viewpoint of practical applications, agent-based modeling can be defined as an individual-centric approach to model design. When designing an agent-based model, the modeler identifies the agents (which can be people, facilities, products, orders, etc.), defines their behavior using *statecharts* and *events,* puts them in an environment, establishes any necessary connections, and runs the simulation. The system-level behavior then emerges as a result of the interactions of the many individual behaviors. Statecharts are one of the most powerful and commonly used ABM tools for modelers. They allow users to define an object's behavior as a sequence of states. Events allow actions to be scheduled in a model. Since each element of the supply chain is a separate unit that operates independently and communicates with others by sending *messages*, the most convenient way to build this model is with the agent-based approach.



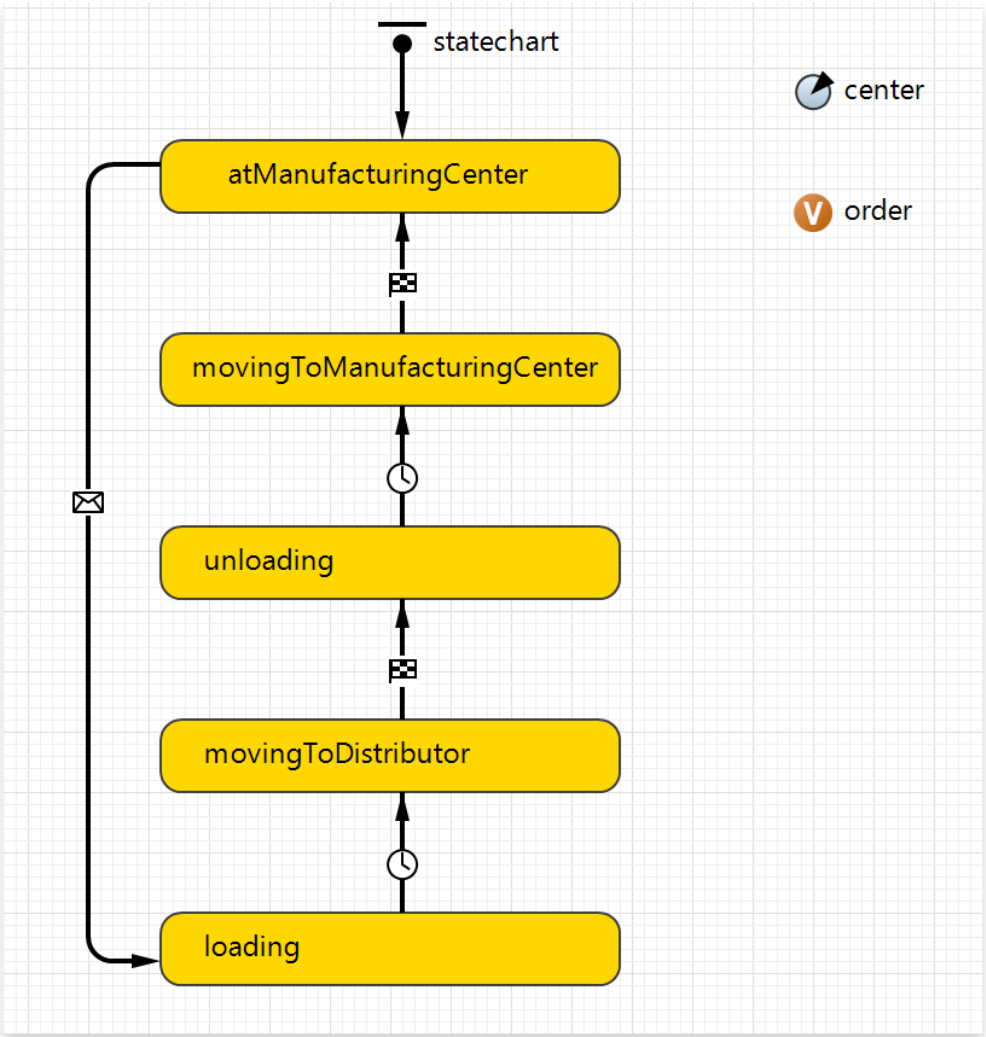
In our model, we will represent the three manufacturing facilities, fifteen distributors, and the fleets of vehicles as agents. In AnyLogic, you can create several agents of the same type simply by creating an agent population. In our case, these are ***manufacturingCenters, distributors,*** and ***vehicles*** (see picture above).

The next step is to place our agents in a virtual environment. It is logical to assume that putting agents on a map is the most straightforward decision for a supply chain model. This way we can consider the real-life locations of our agents and the distances between them. So, we will place the manufacturing facilities and distributors in different cities and make the vehicles move along existing roads and routes. The names of the locations of the manufacturing facilities and distributors are read from a database built into the model. The GIS search engine finds the agent locations and places them on the map.



Database of facility locations

The facilities were placed on the map according to location information in the DB

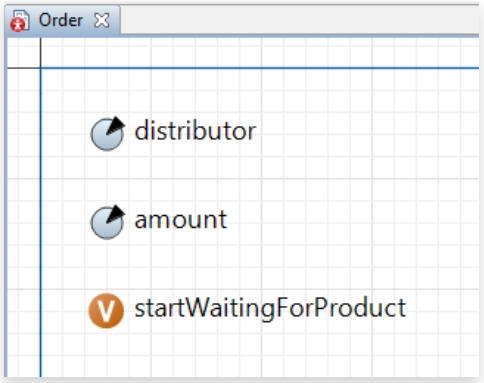


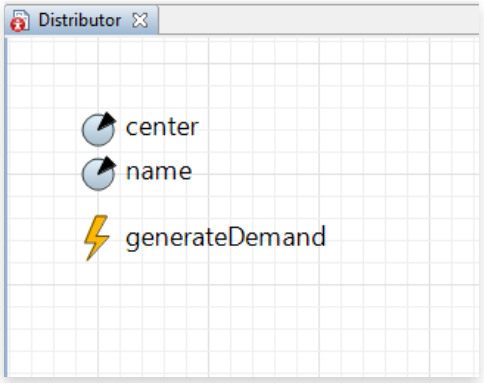
Statechart that represents vehicle behavior

Next, let us define the individual behavior of our agents. Defining behavior in agent-based terms implies the use of statecharts, events, *variables*, and *functions*.

We will define vehicle movement logic using a statechart (see left). The vehicles in our model initially wait for orders at manufacturing centers. Once an order is assigned, a vehicle loads products from storage and moves to the distributor that placed this order. On reaching the distributor, the vehicle begins the unloading process, and then moves back to the manufacturing center to await another order. Such behavior can be represented by the following statechart, created inside the agent ***vehicle.***

We use *parameters* and variables if we need to model agent characteristics. A parameter is commonly used to statically describe objects. Parameters are normally constants and are changed only when you need to adjust your model behavior. A variable represents a model state and may change during simulation. We will use the parameter ***center*** to define the manufacturing center that the vehicle belongs to. This characteristic will not change during simulation. In addition, the vehicle contains a variable, ***order,*** to hold assigned orders.



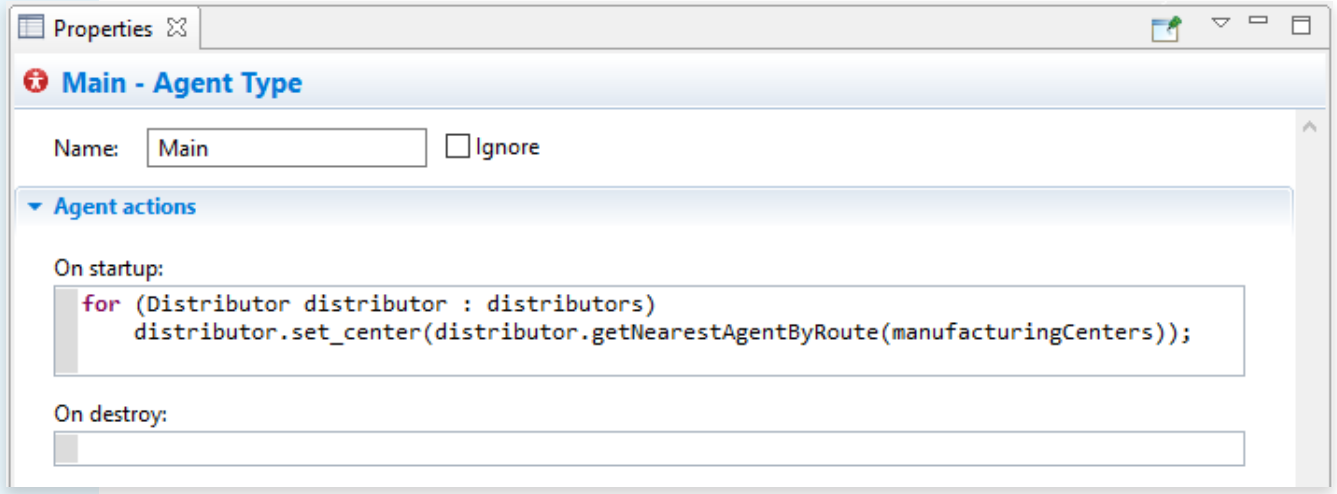


So, this finishes the vehicle behavior definition. Now, let's model how the distributors periodically request products. The distributors will order products by *sending messages* to the nearest manufacturing facility. It requires us to create a new agent type – ***order*** (its internals are shown in the picture above).

We will use the parameter ***distributor*** to define the distributor who created the order. The parameter ***amount*** is added to set the number of products this distributor requests from the manufacturing facility.

Inside agent ***distributor,*** we create the parameter ***center*** to store the manufacturing center nearest to the distributor. Since distributors order the product delivery every 1-2 days, we have to schedule these repeating actions in the model. To do so we use another frequent element of agent behavior – an event. In our model, we need to generate orders, so let us create an event and call it ***generateDemand*** (see picture above).

Coding in Agent-Based Modeling

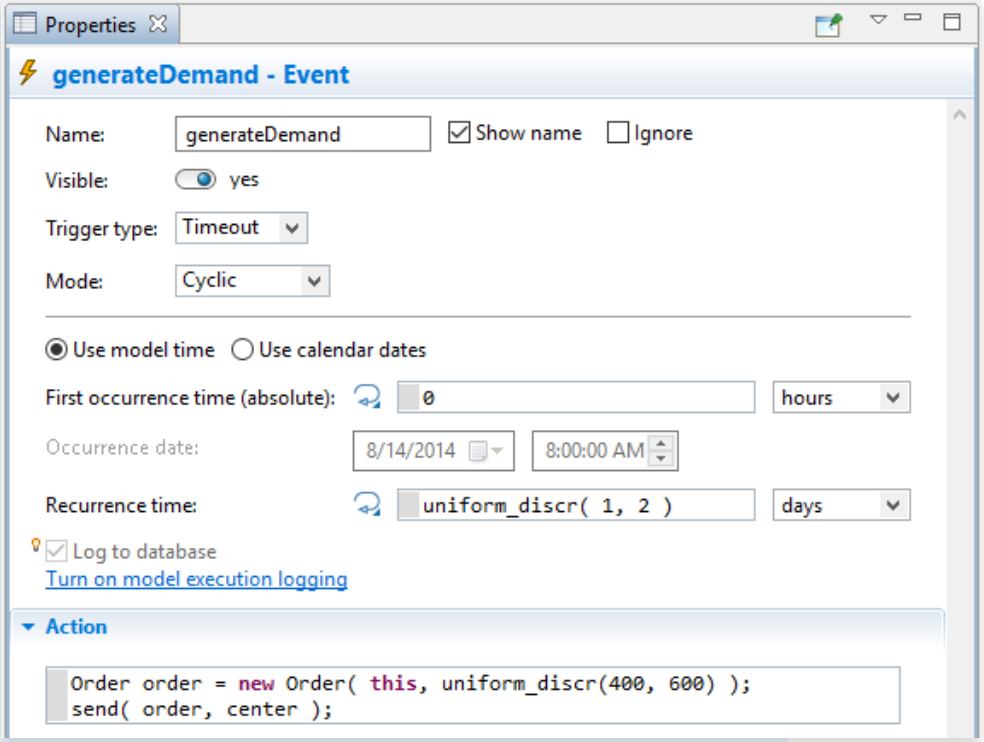


Setting up the nearest manufacturing center to each distributor

Agent-based models usually need some code scripts. There is no need for professional software development skills to create a good model, but small actions like parameter initialization, sending messages between agents, or the movement of agents are defined by adding a few lines of code. In our case, the Java language is used in the AnyLogic simulation software interface.

In our model, we use code to set up the nearest manufacturing center to each distributor when the model starts. The function ***getNearestAgentByRoute*** (population) provides the nearest agent from the specified population. It is important to note that the distance between agents is calculated using existing routes.

Moreover, we use code to model orders as messages between distributor and manufacturing centers. As we said before, distributors order product deliveries every 1-2 days. To schedule repeating actions in our model, we will use the event ***generateDemand.*** An order will be created and sent by executing code lines inside this periodic event.



Scheduling repeating orders in event

Here, the first line of code in the action field creates a new order for a random number of products between 400 and 600. At this moment, the distributor who places the order and number of products requested by this distributor are specified as order parameters.

The second line sends the order that has been created to the nearest manufacturing center. The first argument of the function ***send ()*** is the message that is sent by the function ***(order)***, and the second is the message recipient ***(center)***.

Java code scripts are used in any complex model. It is almost impossible to create a model of a real business system without coding. But anyone with some experience in business analytics can easily learn how to do it.

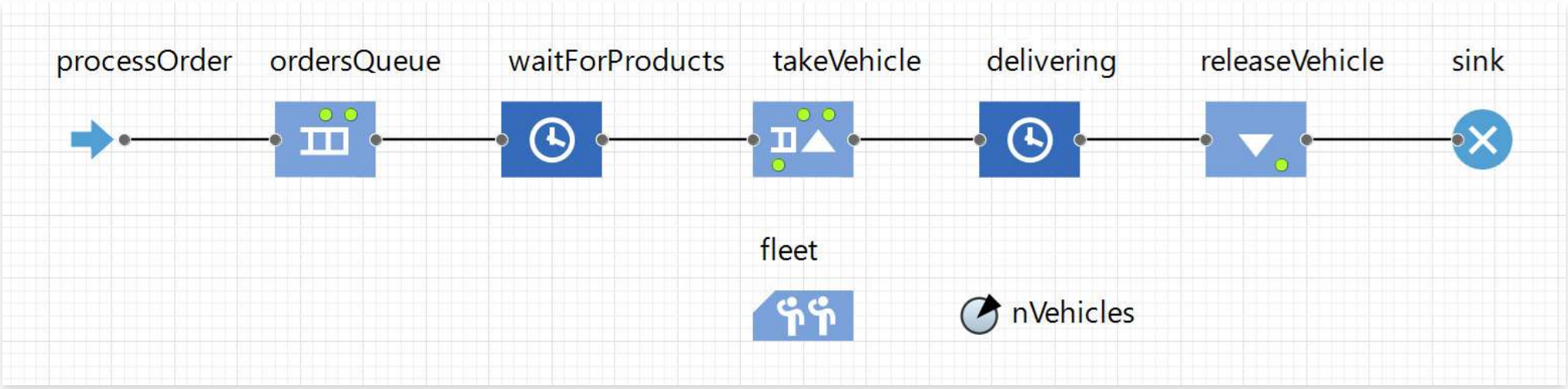
By choosing ABM, we could create the structure and basic logic of the supply chain. We created agents representing manufacturing centers and distributors that operate independently. We located them on the map of China using a database of location names. We also created agents to represent vehicles and orders. And finally, we defined how all these agents behave and interact with each other using statecharts, events, parameters, variables, and a few lines of code.

Processing Orders Inside Facilities – Discrete Event Modeling

After creating the structure and basic logic of the supply chain, let us focus on processes inside the four walls of the facilities. In real life, the internal processes of facilities have a great impact on the performance of a supply chain as a whole. More detailed models allow us to analyze business processes deeply and make more accurate forecasts.

As we said previously, when a manufacturing facility receives an order from a distributor, the staff check if the required amounts of products are available in storage. If they are, the facility sends a vehicle loaded with the products to the distributor. Otherwise, the order waits until the factory produces enough of the products. Once the vehicle with products arrives at the distributor, the products are unloaded into storage. The empty vehicle then returns to the manufacturing facility. To define the processing of orders from distributors at manufacturing facilities and the unloading of delivered products at a distributor site, it is convenient to use discrete event simulation because here we are modeling a classic service system with a clear sequence of operations.

In discrete event modeling, a system is described as a process flowchart. This is a widely adopted graphical representation of processes. Objects that go through the flowchart are often called entities and are passive. In multimethod simulation modeling tools, active agents can become entities in a flowchart. So, we will create flowcharts representing the processes inside the manufacturing centers and distributor storage facilities. The agent ***order*** that was created in the previous chapter will become an entity.



Processing Orders Inside a Manufacturing Facility Flowchart

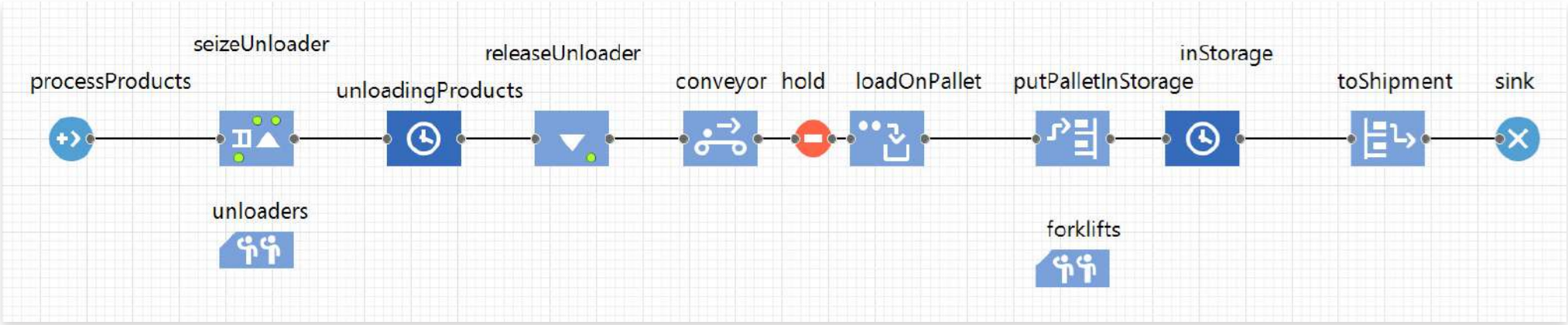
PROCESSING ORDERS INSIDE A MANUFACTURING FACILITY

When an order is received by a manufacturing center, it starts going through the process defined by the blocks of the flowchart:

1. The order is received (***processOrder***).
2. The order is placed in a queue (***ordersQueue***).
3. The order waits until the manufacturing facility produces the number of products requested by the distributor (***waitForProducts***).
4. As soon as the requested number of products is produced, the order is sent to a free vehicle (***takeVehicle***).
5. Products are loaded into a vehicle and move to the distributor (***delivering***).
6. Once the vehicle delivers the products to the distributor, it is released (***releaseVehicle***).
7. The order is completed and removed from the model (***sink***).

Consider that the *fleet* of vehicles whose behavior we already defined by a statechart in the previous chapter is represented here as a set of moving resources. In DES, resources are objects that entities/agents use to perform a given action. An entity must obtain the resource, perform the action, and then release the resource.

As you just saw, we managed to accurately describe process in terms of DES. Let us look further into the distributor storage process.

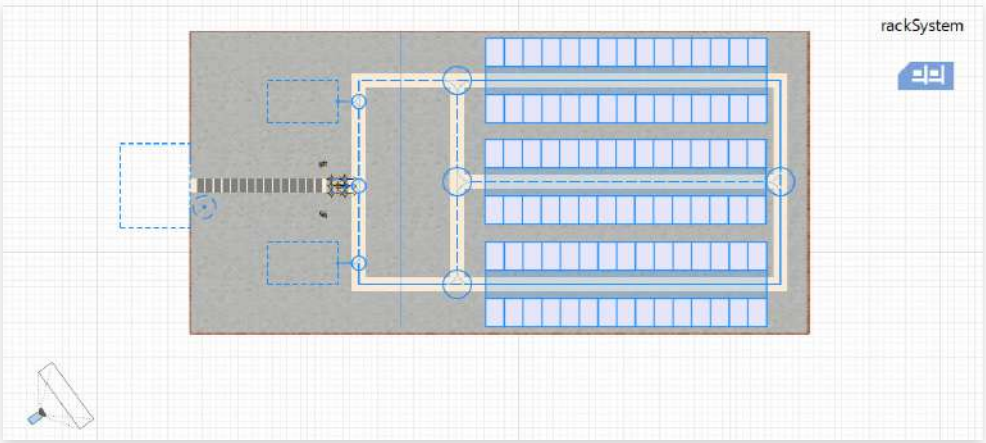


Distributor Storage Order Processing Flowchart

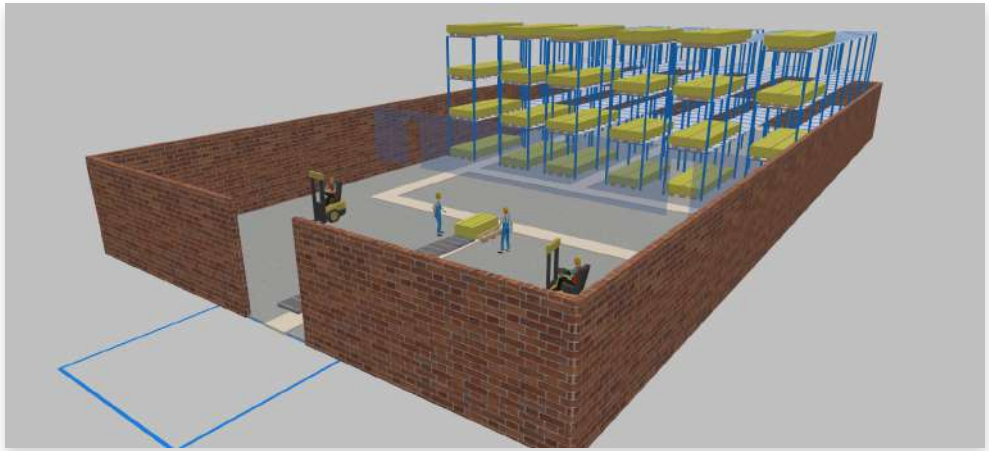
DISTRIBUTOR STORAGE ORDER PROCESSING

Let us model the unloading process at distributor sites, considering warehouse space, and add 3D animation of the process – another powerful feature of dynamic simulation modeling.

When the vehicle arrives at a distributor, the unloading process starts. Workers unload the products, which then move on conveyors to a pallet stacking zone. After palletizing, the products on pallets are moved by forklift trucks to a storage zone. The pallets stay in storage for some time and then disappear as if they were shipped to a customer (let's abstract from this actual activity). In this case, workers and forklift trucks will act as *resources*. These operations are defined by the process flowchart that includes new blocks such as ***conveyor***, *batch* (***loadOnPallet***), *rack store* (***putPalletInStorage***), and *rack pick* (***toShipment***). They represent processing on conveyors, converting several products to a pallet with the stacked products, and working with a storage zone. See the described flowchart on the right.



Defining configuration of distributor's storage



3D animation of storage

The blocks also enable us to develop animation for the process. For most of the blocks, the animation is defined using *path* and *nodes* space markup elements. These elements define the locations of agents in the model environment.

Together, nodes and paths make up a network that agents can use to move along the shortest paths between their origin and destination nodes. A network is created when layout-based modeling is needed. Usually, it is used when a process takes place in a certain physical space and involves moving agents and resources. For example, this can be a hospital or a plant logistics process. In our case, it is a warehouse with unloading, palletizing, and storage zones.

Let us define the network for the warehouse process and connect blocks to the corresponding paths and nodes of this network (see picture above).

Now, we can simply add a 3D window to see 3D animation of our storage. When we launch the model, we will see how the forklift trucks load/unload products and how the storage is filled with boxes in real time.

Using discrete event modeling we have accurately described the order processing inside manufacturing facilities and distributor storage facilities. Adding these details to our agent-based model, we make it much more realistic and trustworthy in terms of supply chain performance analysis or decision making. Moreover, we added 3D animation of distributor storage to make our model illustrative and more easily verifiable.

Production Rate and Staff Rotation – System Dynamics

As was mentioned in the previous chapter, production directly influences the processing of orders. If the required number of products are not available from a manufacturing facility's storage, the order waits until the factory produces enough. This means an underperforming manufacturing facility may cause delays in fulfilling orders and be the cause of inefficiency across a whole supply chain.

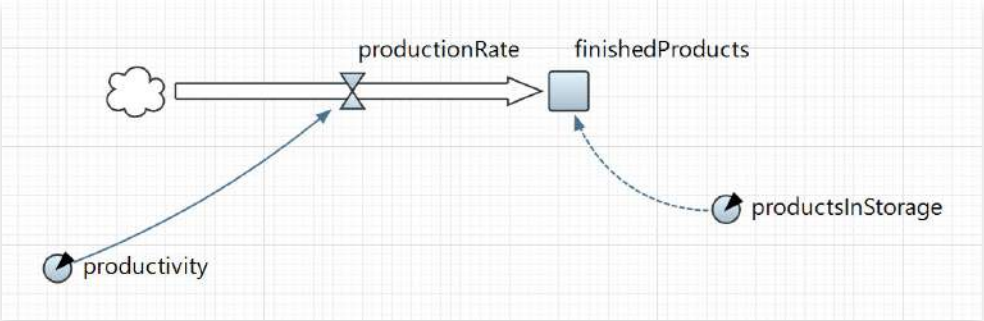
It is important to understand the fact that in the manufacturing field, discrete event models are much more common than any others. They are accurate, detailed, and can be animated in 2D or 3D (as was illustrated in the previous chapter). But to define the production process in our multimethod model, we will add a system dynamics part.

We will use the system dynamics approach because, in this case, we can consider the manufacturing process at a high level of abstraction. In the simplest terms, we are not interested in the manufacturing process itself, we just want to know any impact it has on supply chain performance. From this perspective, the production rate seems to be the most important factor. Let's assume we need to measure the impact of staffing levels on production rate and order fulfilment. As such, we must think about the manufacturing process in terms of *stocks* and *flows*, the two basic elements of SD.

We will start the SD modeling with production. As we are describing the internal processes of manufacturing facilities, the stock and flow diagram should be created inside the ***manufacturingCenter*** agent. This allows it to change the behavior of all three manufacturing facilities.

In our model, ***finishedProducts*** stock represents the amount of finished goods that are ready to be transported from a manufacturing facility to distributors. It is increased by the ***productionRate*** flow. ***finishedProducts*** is also continuously increased with the parameter ***productivity*** per worker per day. The parameter ***productsInStorage*** is used as an initial stock value. Parameters are used in SD to define inputs.

This diagram represents the simplest system dynamics model of production. Since we decided that the staffing level is the most important factor for production rate, in the next step we should build the staff dynamics part of the model.



Stock and flow diagram of production

To build the staff dynamics part of the model we create one more stock called staff with ingoing and outgoing flows representing staff rotation.

We define the initial number of workers with the *parameter* ***initialStaff*** and the target staff level with the parameter ***staffThreshold***. The dynamic variable ***gapInStaff*** defines the difference between ***staffThreshold*** and current ***staff***:

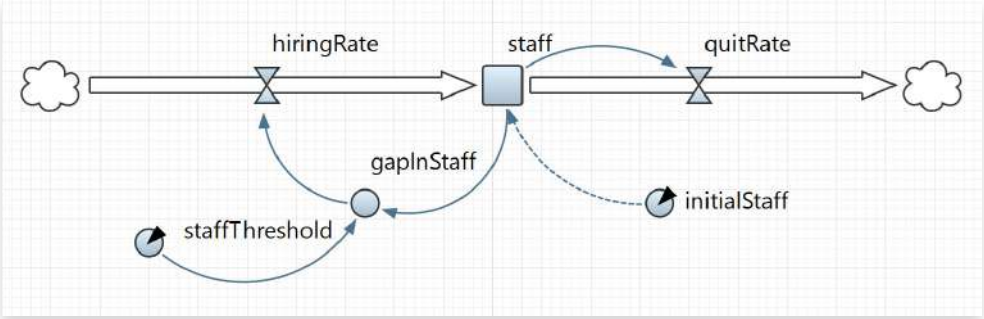
gapInStaff=staffThreshold-staff

We should also define a formula for the hiring rate. Let's hire people if there is a gap between the target staff level and the current one. The formula will be as follows. This is the Java code for the conditional expression, condition before the question mark is checked, if it is true the first value will be chosen, if not then the second one:

hiringRate=gapInStaff>0?5:0

The same conditional operator is used for the quit rate, since we do not want the staff level to go below zero.

quitRate=staff>0?2:0



Stock and flow diagram of staff dynamics

So, we've created an SD model of staff dynamics. Now let us integrate it with the production part. This combination should allow us to support a few realistic assumptions:

- Production rate depends on the number of workers.
- Manufacturing sites stop hiring if there are too many finished goods in stock, so we do not need to produce with the same rate.
- Overproduction causes staff dismissals.

First, we should determine the relationship between ***staff*** and ***productionRate***:

- If ***staff*** is more than 0, productivity is multiplied by the number of ***staff***.
- If ***staff*** is equal to 0, ***productionRate*** is zero, since we have no workers.

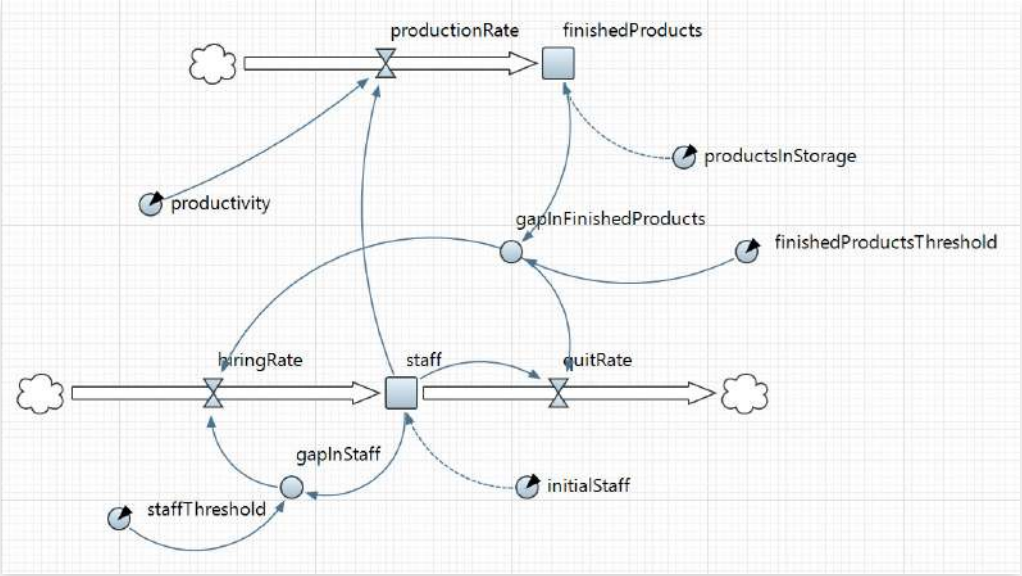
$$productionRate=productivity * (staff>0?staff:0)$$

Then we add two new elements:

- Parameter ***FinishedProductsTreshold*** represents the level of overproduction, when we want to stop hiring new people.
- *Dynamic variable* ***gapInFinishedProducts*** represents the difference between ***finishedProductsThreshold*** and ***finishedProducts***.

Now we should modify the formulas for ***hiringRate*** and ***quitRate***.
A manufacturing center will hire workers only if there is a positive ***gapInFinisedProducts***.

$$hiringRate = gapInFinishedProducts>0? (gapInStaff>0?5:0):0$$



Combined production and staff dynamics parts of SD model

We will apply a little bit more complex logic for the quit rate. There are three cases:

1. When there are no workers at a manufacturing facility, staff is equal to zero, so the ***quitRate*** is zero too.
2. Staff are working at a facility and the amount of ***finishedProducts*** has not yet achieved the threshold level. In this case the quit rate is 5/***gapInFinisedProducts***.
3. There are workers and the threshold in production has been achieved, in this case two workers per hour quit the ***manufacturingCenter***.

You can see these conditions described in the formula below:

```
quitRate = staff > 0?5/week() / (gapInFinishedProducts>0? gapInFinishedProducts:1):0
```

As we already know, the internal processes of the network sites have an impact on supply chain performance. SD is a natural manner for describing things at a high level of abstraction, when you do not need to dig into details. Using the system dynamics approach, we have described the relationship between production rate and staffing levels at manufacturing facilities. By adding the SD dimension, we have made our simulation model more functional. Now it represents not just network level interactions, but also the inside 4 walls processes and internal dynamics of manufacturing facilities and distributor sites.

Integrating Models and Adding Statistics

Previously, we described different parts of the supply chain with the ABM, SD, and DES methods. To make them into a single, efficient, and manageable multimethod model that represents the behavior of the whole system, we should do a few more things.

First, we have to combine the ABM, DES, and SD parts together. Creating interfaces between ABM, DES, and SD parts of a model is an essential part of building a multimethod model.

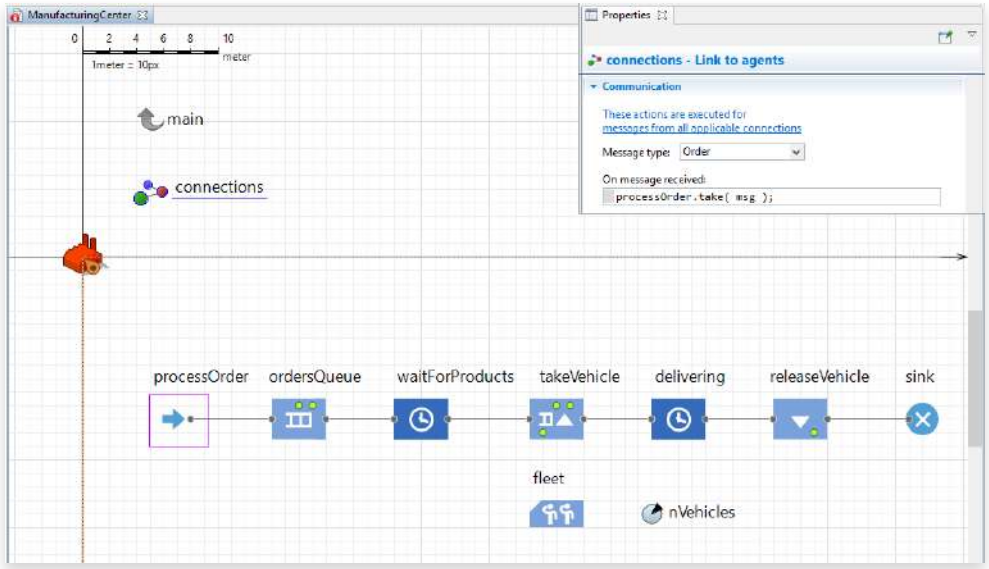
Statecharts, process blocks, markup elements, SD variables, and all the other elements we used were added to different parts of the same model. So, all of them are accessible from any other element. Sending messages, using mathematical operations, and calling functions are the three methods of connecting different parts of the model. Object functions are usually called by Java code and make these objects perform certain actions. For instance, to insert an agent via an object, you call its *take (agent)* function.

Let us look at some examples from our supply chain model.

1. ABM – DES CONNECTION: SENDING ORDERS TO PRODUCTION SITES

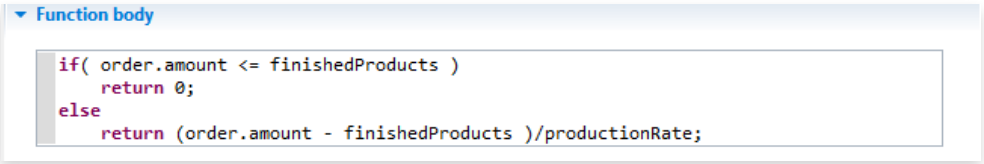
When a distributor sends an order to its nearest manufacturing facility (ABM), the order should enter the flowchart that simulates order processing inside the manufacturing facility (DES). This connection can be logically divided into two parts:

- Sending a message from agent ***distributor*** to agent ***manufacturingCenter***
- Processing this incoming message by calling the function ***take()*** with Java code

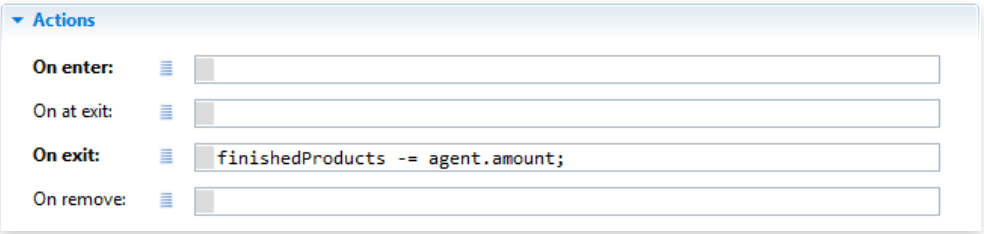


2. DES – SD CONNECTION: WHEN ORDERS WAIT FOR PRODUCTS TO BE PRODUCED

If there are not enough products in the factory's storage, represented by products stock (SD), the order will wait in the ***waitForProducts*** block (DES) of the factory's order processing flowchart. Using mathematical operations, we can calculate the waiting time depending on the availability of products. If the number of products ordered by the distributor is less than what is available in storage, it can be shipped immediately. In this case there will be no waiting time. Otherwise, the model will calculate the time required to produce this number of products depending on the production rate. This is done through calling the following function in the ***waitForProducts*** block:



In both cases, when the products are shipped, the finished goods inventory level of the factory will be decreased by the number of products specified in the order. This is how it is reflected ***waitForProducts*** block:



3. DES – ABM CONNECTION: ASSIGNING A VEHICLE TO FULFILL AN ORDER

When a vehicle is assigned to an order in the processing orders flowchart of a manufacturing facility, in the ***takeVehicle*** block (DES), the vehicle enters the ***atManufacturingCenter*** state of the statechart (ABM). We need to send the order to the vehicle that has been chosen to fulfill the order so that the vehicle can load the products and start moving to the distributor that placed the order. Doing so changes the state of the chosen vehicle to ***loading,*** and then, after the time required to load the products, it changes to ***movingToDistributor***. So, in this case, a message acts like a trigger to change an agent's state in the statechart.

These are just three examples. In addition, in our supply chain model, we connected:

- The vehicle statechart with the distributor flowchart to start the process of transferring products into a storage zone after a vehicle arrives at a distributor site and has been unloaded.
- The vehicle statechart with the manufacturing center flowchart to complete an order when a vehicle assigned to an order returns to the manufacturing center.

In a multimethod model of a real business system there would be many more connections. Nevertheless, basically they are built in the three ways described above:

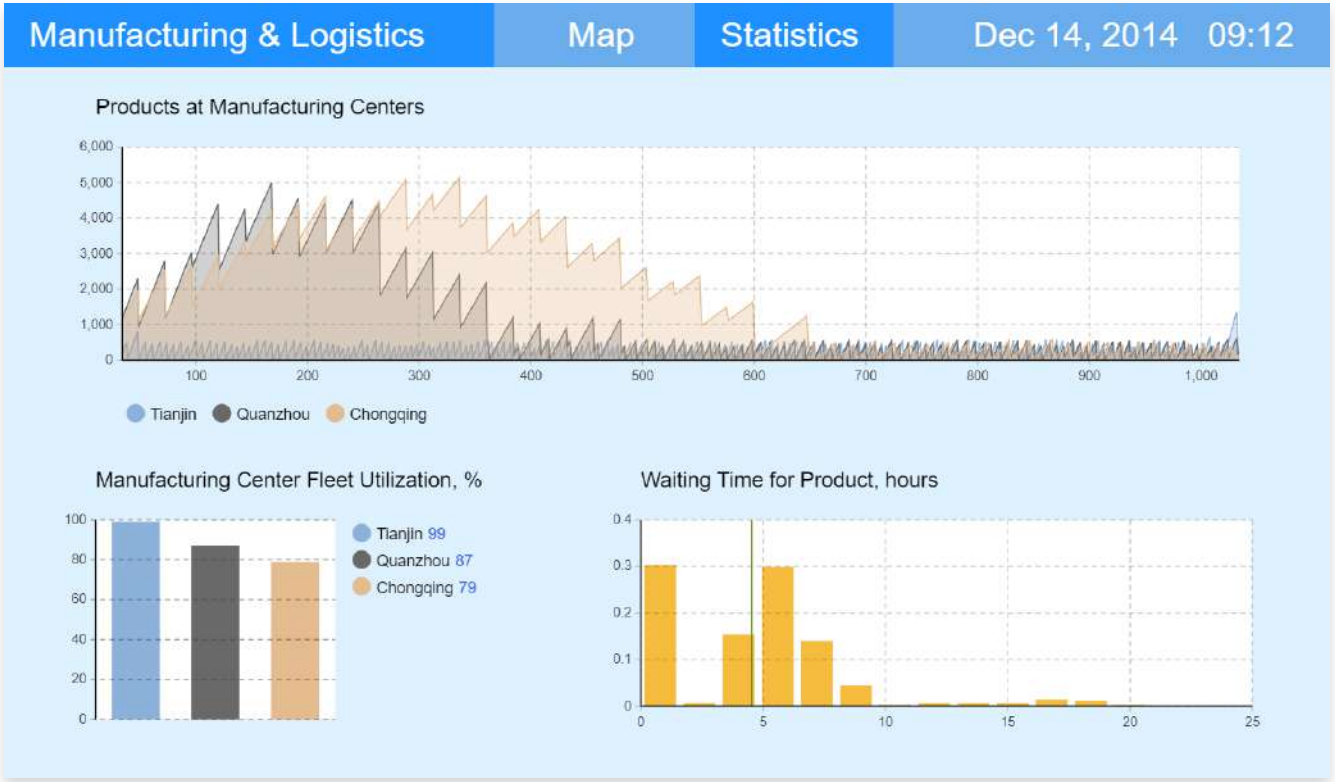
- Calling functions with Java code
- Sending messages (can sometimes be combined with Java code)
- Mathematical operations

After all the elements are interconnected and everything works correctly, we can add the visual representation of any outputs we need using histograms, charts, plots, or any other graphics components.

In our example model, we decided to graph the finished goods inventory level (***Products at Manufacturing Centers***), fleet utilization, and waiting time for product. See the example model statistics on the right.

Statistics are what most of the models are built for. Experimenting and analyzing them is often the most important part of the project.

[Run the complete Manufacturing & Logistics model in AnyLogic Cloud>>]



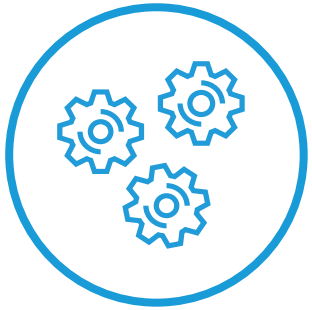
Supply Chain Multimethod Model Statistics

Theoretically, we can add a few more parts to our example model and then measure and represent on charts: order lead time, service level, distribution center inventory level, distribution center capacity utilization rates, staffing levels, production rates, and many more.

MULTIMETHOD MODELING ADVANTAGES

In previous chapters we have built, step by step, a model of a supply chain in China. We figured out how to use simulation software tools to create good and consistent models and we created our model using multimethod modeling software, combining ABM, DES, and SD.

Historically, all simulation tools have been single-method, but we believe that good simulation software for business applications must be hybrid and use multiple modeling approaches. We believe this because of the following benefits:



CAPTURE THE NATURE OF YOUR BUSINESS SYSTEM

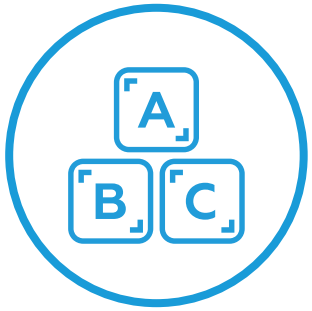
All systems are different and before creating a model, you need to understand how the system to be modeled is organized and how you can represent all the important aspects of it most accurately. What would be best for solving our problem: to consider the system as a whole, as a process, or as separate interacting elements? Depending on the answer to this question, we can choose the most suitable simulation modeling method: SD, DES, or ABM.

But what if some elements inside that system are organized in a different way to the rest of the system? Using multimethod simulation software, you can describe different parts of a system with different methods. So, the ability to capture business systems and their real complexity is almost unlimited.

DECIDE WHICH DETAILS ARE IMPORTANT

The process of model building should be started with a clear understanding of its purpose. When the purpose of the modeling is clear, you can easily decide what should be captured and how it should be modeled to make the model useful.

To represent a real system in a simulation model, we need to use a modeling language. This process involves abstraction. If you use single-method simulation software, you do not have many options. Every simulation modeling method works at a certain level of abstraction and has a limited set of tools. Only a combination of different approaches will allow you to capture all the important details and make your model fit to meet your challenges. With multimethod simulation software, your model structure will depend on what input data you have, what outputs you want to get, and your vision for the model, but not on your software constraints.



MAKE YOUR MODELS SIMPLE AND EASY-TO-USE

We believe that simulation models that look unnatural and inadequate are most likely not useful. Trying to use one modeling method to solve a complex business problem often leads to compromising workarounds to overcome the constraints of the chosen approach. Even if you manage to make a model as accurate as possible, models built using workarounds are more often unnecessarily difficult to use. The complexity of such models makes it time consuming to find mistakes, to extend, and to support. With a multimethod tool you can create natural-looking and easy-to-use models with no hacks or workarounds.

EXTEND AND REORGANIZE YOUR MODEL AS MANY TIMES AS NECESSARY

In real life, it is sometimes impossible to accurately identify the causes of problems. When building a model, we start to explore and understand the structure and behavior of the original system.

As we have proven with the supply chain example, a model is usually developed over many iterations. First, we created an ABM supply chain network, then we added a DES element to represent the processing of orders inside manufacturing facilities, and so on. We did this to illustrate the possibilities of each method, but how many iterations would be required to model a real business system?

Imagine that you created a real-life supply chain model, validated it, and then started to search for the reasons behind a low production rate. Your first guess is that the problem is with the level of staffing. So, you built the same SD diagram of staff dynamics as we did and validated your model. When you started to experiment with the model, and varied your virtual HR policies, it became clear that they do not really affect production rate much. At this point, with a multimethod modeling tool, you can easily change assumptions and, for example, replace the SD model of staffing levels with a DES model of equipment maintenance.

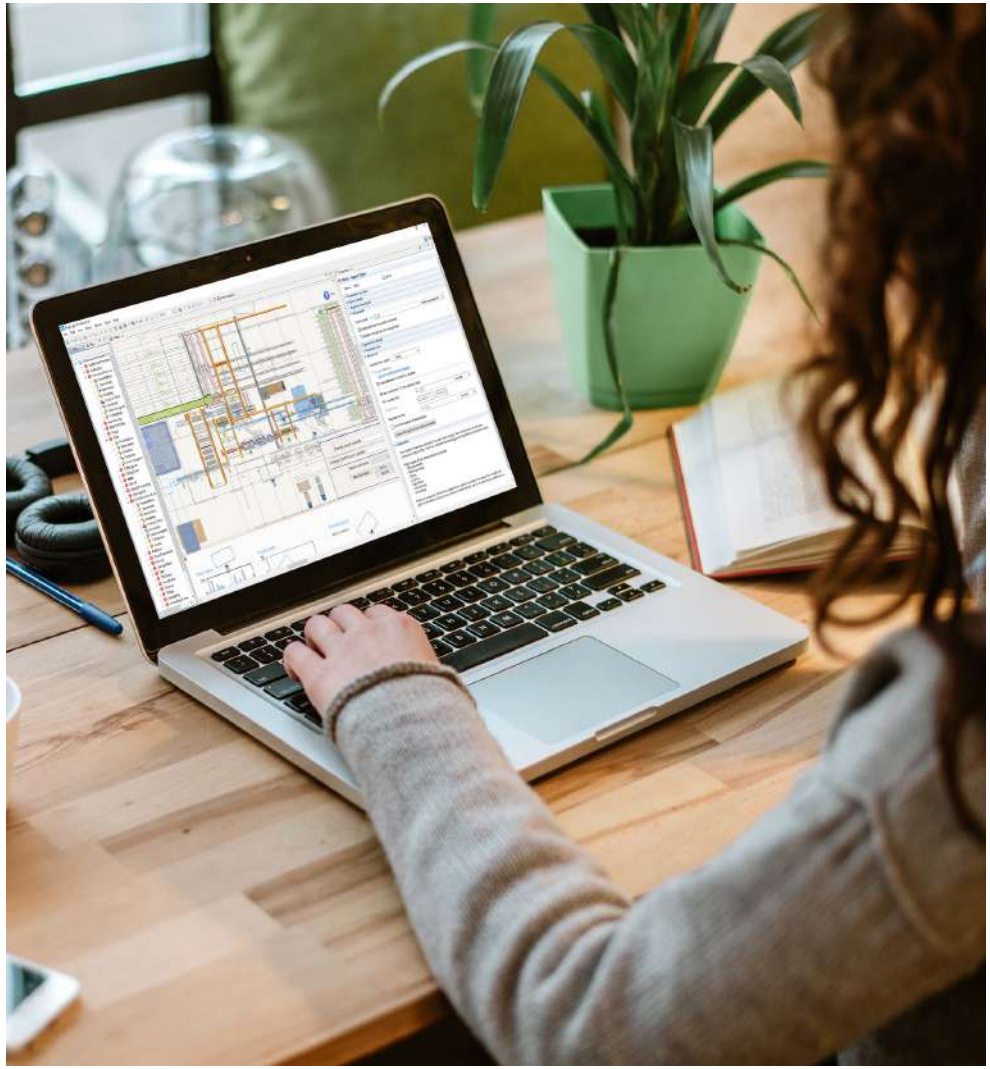
USE YOUR SIMULATION SOFTWARE FOR THE ENTIRE BUSINESS LIFECYCLE

With single-method software you are always limited by the capabilities of the approach you use. For example, system dynamics is good for representing and analyzing global dependencies, but it should not be used if it is more convenient to describe your system as a process or if you need to analyze the behavior of a system that consists of many independent objects.

Multimethod modeling gives you the opportunity to implement modeling in all areas of your organization's activities. When you already have a supply chain model, you can go on to model and optimize your warehouses and manufacturing facilities, from their layouts to their internal business processes, or build a country-wide market model that will provide demand data for the supply chain. Multimethod modeling allows analysts to integrate models across the entire organization and share expertise, as they all can be built using the same software.

The main benefit of multimethod modeling software is that you can decide how to represent your system best. You can decide what elements to include in the model and not have to make compromises to fit the constraints of single-method software. Using multimethod modeling, you have a complete stack of technologies that give you the freedom to decide how to build a model. In fact, you can build a model for any business area. You can create efficient and manageable models without using workarounds.

Most simulation software tools are single-method and do not give you such options. With multimethod software tools you can build models at any abstraction level you need and find a solution for almost any problem.



Conclusion

Simulation modeling has accumulated many success stories in a very wide and diverse range of applications. Traditional modeling tools, like spreadsheets, fail to handle the modern demands of more and more companies, and these companies are finding that integrating simulation models into their business processes provides the answers they need. With computer power growing, and with the rise of new methods like ABM, simulation is penetrating new application areas, such as pedestrian, road traffic, population modeling, and epidemiology. It is also allowing modelers to create more complex and realistic models. Working with simulation software tools, especially discrete event, became a standard for business leaders a long time ago. Nowadays, we can see how multimethod (or hybrid) modeling is developing into the standard for business applications.

Building high-quality simulation models typically requires at least some level of training and a period of learning. However, the efforts invested in the adoption of simulation technology pay off and allow you to experiment in a risk-free environment, analyze any system with dynamic behavior, or find answers to “what-if” questions.

anylogic

In this paper we covered three major methodologies commonly used for building dynamic business simulation models: discrete event modeling, agent-based modeling, and system dynamics. We also illustrated what multimethod modeling is by building a supply chain model using all three methods. In general, the choice of method should be based on the system being modeled and the purpose of the modeling. Each method serves a range of abstraction levels and is suitable to deal with certain types of challenges and tasks.

The problem is that most real-world business systems are very complex. The ability to capture business systems with their real complexity and interactions can be seriously limited by using only one modeling method. So, single-method software tools are not ideal for handling complicated tasks.

With multimethod modeling software tools, you can create efficient and manageable models without using workarounds. You can choose the best representation of your system, not limited by methods or tools. Using multimethod simulation software gives you the flexibility needed to successfully solve any problem without excluding important elements or having to develop workarounds. With multimethod modeling, you can always choose the most efficient combination of approaches and develop the best solutions for your problems.

Additional resources

- White papers
- Related white papers for this work:
  - Simulation Software Comparison
  - Developing Disruptive Business Strategies with Simulation
  - An Introduction to Digital Twin Development
  - Simulation Modeling for Consulting
- Case Studies
- Videos
- Books
- Download AnyLogic Simulation Software
- Seminars and Trainings

Contacts

THE ANYLOGIC COMPANY

info@anylogic.com

Contact your local office >>