Simulation Masterclass

Assigment 1 - group 14

Claudio Prosperini	6084931
Juliëtte van Alst	5402409
Gerben Bultema	5309247
Roelof Kooijman	5389437

TU Delft

SEN9110

Content

- Practical information
- Description model
- Description Experiments
- Results of the simulation experiment
- Comparison Salabim, MESA and SD
- What was easy and what was difficult

Practical information

- Link to Github repository: https://github.com/roesel1/SEN9110-G14
- The correct version has the commit message: FINAL ASSIGNMENT 1
- The basic supermarket model is in the file *ASSIGNMENT_1_Basic_supermarket_model.ipynb*
- Unfortunately, we were not able to get a dataset containing the average waiting times per resource for each experiment, and to create plots. However, we still wanted to explore the model and therefore we created are own experiment: what is the best number of checkouts to improve the average shopping time? This is in the file named *ASSIGNMENT_1_Experiments.ipynb*
- *A requirement* file is added. If it does not work then creating a env with python 3.10 and pip installing salabim,pandas,numpy, notebook, jupyterlab, seaborn and matplotlib should work

Description model - Initialization of the model

- The model and its initialization is also conceptualized in slide 5.

Creating environment: The environment is initiated by *env = sim.Environment(time_unit='seconds').*

Resources: Resource classes are created for the basket and carts. The capacity of the resources are currently hard coded. However, it should be better if this is loaded from a separate file with resource data. Then a distribution is defined to indicate what the probability is that a customer takes a cart or a basket. Also, for both the bread and cheese&diary sections a resource class is defined, for which the number of clerks are the capacity.

Picking items: For for every event, an appropriate distribution is created from which the holding times for event can be sampled. For example, fetching a frozen foods item takes between 2 and 19 seconds with a mode of 8 seconds and be defined as a triangular distribution. This can be defined using salabim as *frozen_foods_distribution = sim.Triangular(2, 19, 8).*

Routes: The presets for the routes are made by creating two list which have the correct order of the sections and then defining a probability density function which can be used to assign the routes to the customers. The time durations, routes and distribution will be stored in separate dataset in the future.

ComponentGenerator: The model needs to know when and how many customers to generate this done by creating salabim ComponentGenerators. How the various generators are created is shown in the code block below. The *customer_distribution* contains the average number of customers which should come in every hour. Then a for loop is used to create a generator which per vale creates an exponential distribution (shown red) then for the starting time for the generator 3600 is multiplied with the index to get the correct time to start creating customers. The duration is set at 3600 so that the generator stops after an hour.

```
customer_distribution = [30, 80, 110, 90, 80, 70, 80, 90, 100, 120, 90, 40]
for index,i  in enumerate(customer_distribution):
    sim.ComponentGenerator(Customer, iat=sim.Exponential(i, time_unit='hours'), at=index*60*60, duration=60*60)
```

Checkouts: The last step for the initialization of the model is the creation of checkouts which is done by defining a clerk which is a resource with linked to it. The number of checkouts with clerks will be defined by a separate dataset but is now hard coded.

Time: The model starts at 8 AM and customers are generated for twelve hours. Unfortunately, customers are still in the shop after the shop closes, since they are not done shopping yet. Therefore, we have not set a duration time for the run, and the model will stop running if all the events stopped (and all the customers left the supermarket). This is defined as *env.run().*

Description model – Event scheduler and processing event

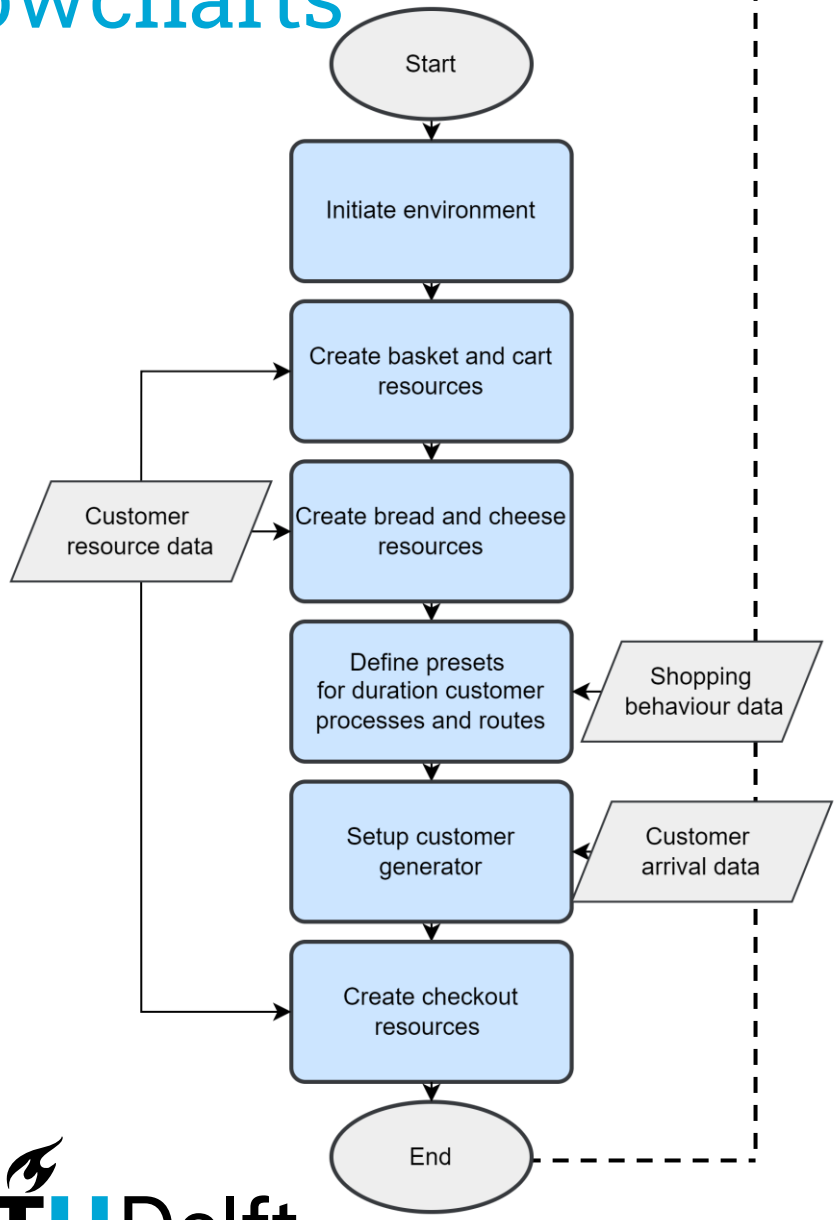
.

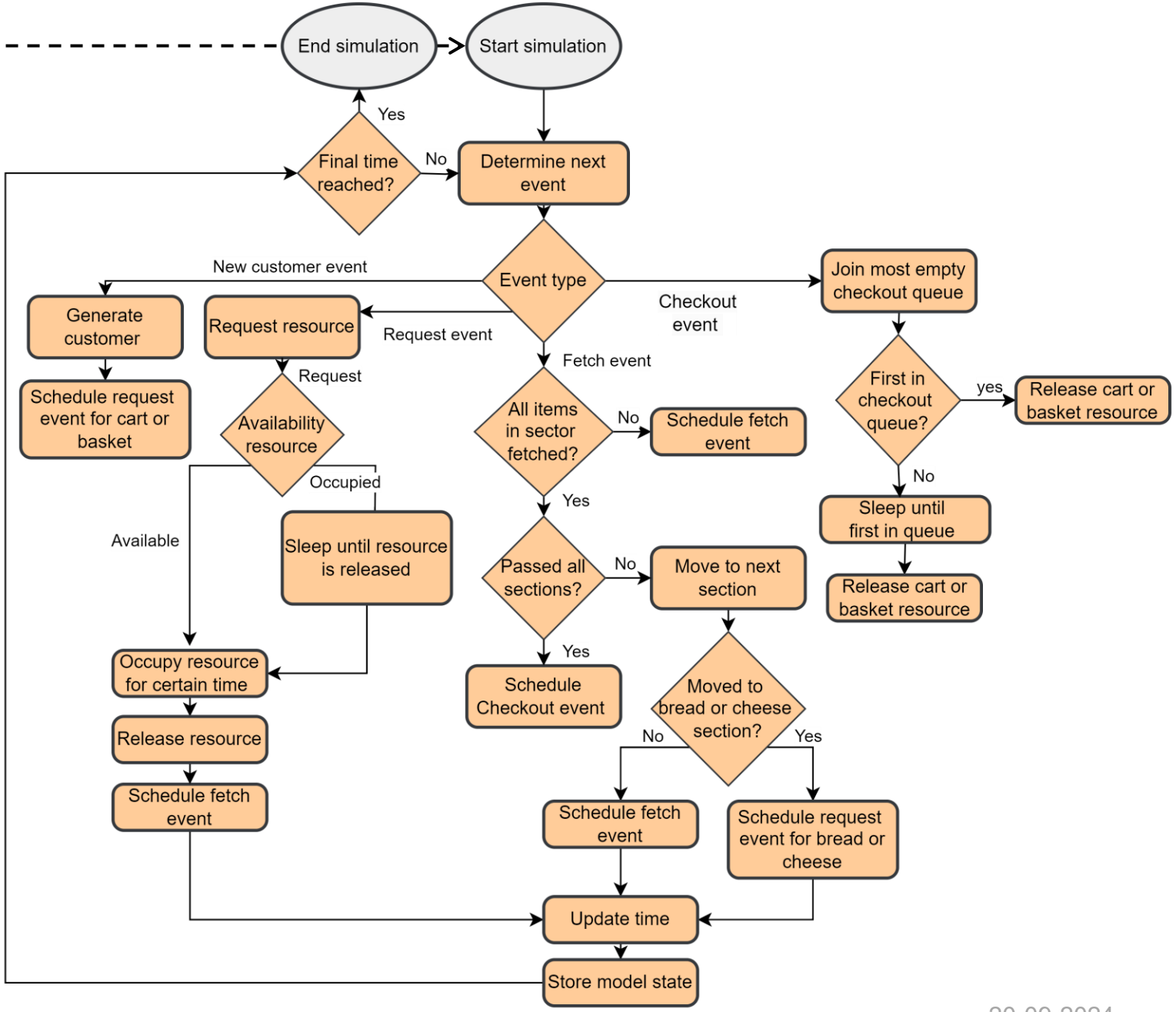
Creating a flowchart with an ABM perspective proved difficult. Where with ABM one iterate for the agent scheduler and explain with flowchart the process from waking up and until sleeping again. Doing that now with this assignment and a DES model would not make sense because there is no behavior that the customer expresses. It is more an entity which processes through series of events before it is complete. So, making a flowchart becomes easier when iterating through the various events defined instead of iterating of all the behavior.

The model has four types of main events:

1. New customer event → A new customer is generated and schedules a request event.
2. Request event → When a customer wants a resource (basket, product or clerk), the customer requests this. The availability is checked, and the resource is released. Lastly, a fetch event is scheduled.
3. Fetch event → It is checked if all items in the current department are fetched. If this is the case, the customer moves to the next section. If the next section is de bread or cheese section, they request a resource event, otherwise a fetch event is scheduled. If a customer has all the items, a checkout event is scheduled.
4. Checkout event → A customer will sleep until they are the first in the queue. When they wake up, they release their release their cart or basket.

Flowcharts





TU Delft

Results of the simulation

- Unfortunately, we did not figure out how to monitor the data with the different queues and put it in one dataset. Therefore, we were not able to log the different waiting times per department and plot them

We were able to log the individual times of one customer, which are displayed in figure 1. There, it is possible to see that a customer got a cart and is picking the fruit and vegetables.

In addition, we have data for the bread clerks and cheese&diary clerks during 1 run. For example the average stay the customers are at the bread_clerk is 62 seconds, and the average of the row length is 2.8 customers.

```
Customer's Action Log for customer customer.500:
At time 22262.595062173812, customer: Entered cart/basket queue for Resource (carts)
At time 22262.595062173812, customer: Got Resource (carts)
At time 22262.595062173812, customer: Getting fruit_and_vegetables
At time 22263.963632728737, customer: Got fruit_and_vegetables
At time 22264.2389613604, customer: Got fruit_and_vegetables
At time 22266.339846991545, customer: Got fruit_and_vegetables
At time 22266.912460812255, customer: Got fruit_and_vegetables
```

Figure 1: Logging of 1 customer

Length of stay in claimers of bread_clerks	entries	1036
	mean	61.957
	std.deviation	61.374
	minimum	0.026
	median	42.612
	90% percentile	149.426
	95% percentile	185.281
	maximum	422.143
Length of claimers of bread_clerks	duration	45125.333
	mean	2.852
	std.deviation	1.414
	minimum	0
	median	4
	90% percentile	4
	95% percentile	4
	maximum	4

Figure 2: Logging of bread_clerks

Description Experiments

- Since we were not able to create the dataset with the different waiting times, we decided to still do a small experiment by determining the best number of checkouts to improve the average shopping time. This is done by keeping all values constant, expect the number of checkouts. The model has been run with using 1,2,3,4,5,6,7,8,9,10 checkouts using 100 replications for each number of checkouts.

The original plan was to use the EMA workbench to do the explorative analyses, but during the implementation effort was made to create separate .py files containing the Customer class and then import the class. This, however, made it impossible to directly sample from the time distribution. To solve this, we tried to create an environment class which contain attributes accessible for all customers (we based this on how it works with MESA). Here the problem that the customers need to have a parameter which gives the model class object which is not possible when using the Component Generator of Salabim. The solution to fix this was to add the class code in the model class and then set an attribute directly in the init of the customer class. The code that is handed in for grading is the old version where the replications and experiments are done using a double for loop and adding the values to a dictionary.

Results of the simulation experiment

- Saving the average through put in seconds per run in a list for one experiment and then putting this list into a dictionary where the keys are the number of checkouts used, we can make a boxplot depicting (sound like ChatGPt but isn't) how long it takes on average to walk through the supermarket. The small interquartile range indicate that all the replication result in similar times. Noticeable is that adding checkouts significantly decrease the throughput time until the third checkout. After the third checkout adding more does not result in faster throughput. An explanation could be that with fewer checkouts the checkouts form a bottleneck where customers accumulate.

It needs to be mentioned that in the x-as, 1 checkout should be added. So, zero in the graph is in the model 1 checkout. We did not have time to fix the layout, due to the long run time. That would be a lot of time for such a small change.

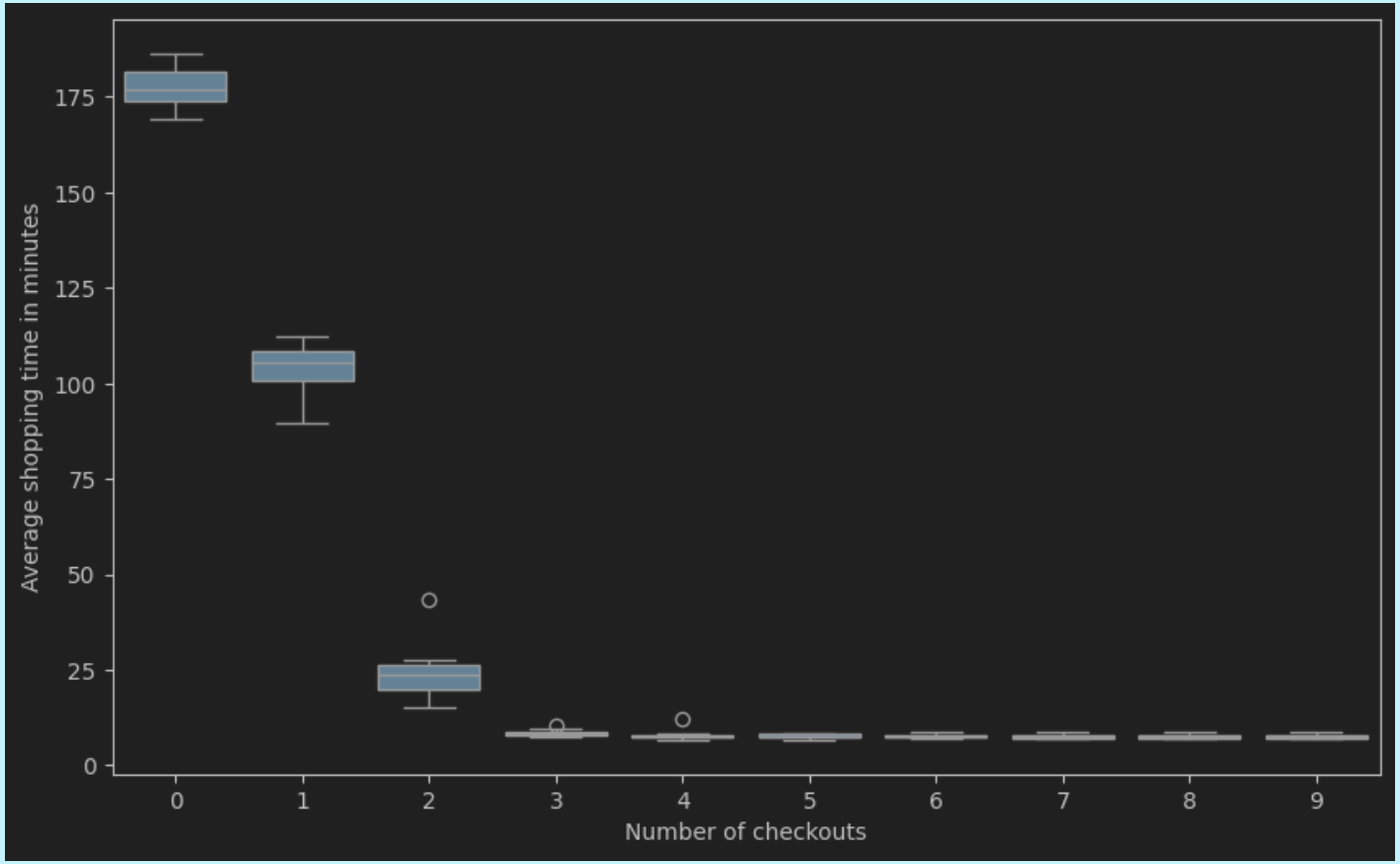


Figure 3: Average shopping time with different amounts of checkouts

Comparison Salabim, SD and MESA

Our group has experience making System Dynamics (SD) models in Vensim and Agent-Based models using the MESA python package. The main differences are given in the table 1.

Table 1: Differences between the simulation package

Aspect	SD using Vensim	DES using Salabim	ABM using MESA
Time	Continuous calculated state model per certain time intervals	Discrete, based on events	Discrete based on agent actions per step. Time steps have same interval
Level suitability	Macro-level	Micro-level	Micro-level
Interface	GUI, coding of equation	Python based coding	Python based coding
Components	Stocks, flows, feedback loops	Classes, Objects, resources, queues, events	Classes, Objects, Agent scheduler, enviroment

DES modelling is most definitely better suited for this assignment than SD modelling because of 2 reasons. Firstly, SD would be impossible to model customers as single entities. With SD one should have calculated the shopping duration using delays in flows. This would make it impossible to follow individual behavior of a shopping customer. Secondly, uses deterministic differential equations making it hard to introduce stochastic characteristics of shopping behavior. Not every customer needs the same amount of time to shop. The only advantages of SD is that it is implemented in Vensim which has a GUI with some useful tools for visualizing results with graphs and setup experiments. However, the use of GUI makes it harder to create specific things for example a 2D/3D visualization of movement through the supermarket.

Both DES and ABM could be used, but when also considering the used package Salabim seems more suitable for various reasons. Firstly, salabim includes ready to use components to model queues and resources making it very easy to implement the carts and baskets, cheese, bread and checkout parts of the assignments which are the biggest part of the assignment. Additionally, salabim has a component generator which makes it easy to create customers through out the simulation. Furthermore, ABM model is slower because it needs to activates all the agents every step while DES only activates entities when they need to change their state. Lastly, a big advantage of ABM is not applicable in this assignment. ABM lends itself better to model specific and frequent interactions between customers better which are not yet required in this assignment.

During modeling the supermarket it was, however, hard to step away from how models are created using MESA. The biggest change is that MESA defines a general model class that contains general environment data which can be easily accessed by agents. With Salabim there is no such class, so we made one of hour own (the version with a model class is in branch not the main). The hard thing then was setting the main model class object as an attribute for the customers. This was hard because the Component generator does not take parameter arguments. The solution was to define the customer class within the model class (Code does not contain this yet)

Description of what was easy and what was difficult

- At first we had two classes: customers and clerks. However, it was hard to work with two entities interacting via passivating and activating. We decided to make it easier by removing the checkout clerk as a separate entity and made them a resource. In general, the process interactions took some time to understand, and generally avoiding them for now made it much simpler to implement the model correctly. This approach does limit the extensibility of the model as we cannot easily add behaviour to the checkout clerk.
- It was difficult to see what was going wrong in the simulation. E.g. we had a bug in the checkout resulting in waiting times of an hour at the bread department. By adding a log we were able to identify the issues and resolve them.
- It was very easy to define resources, especially compared to what we were used to in MESA.
- The built in methods of the ComponentGenerator were also very nice as they made it trivial to create customers according to a distribution over time.
- Experimentation is harder compared to MESA batchrunner, as you need to define the experiments yourself and handle the monitor's ouputs.
- Lastly, we had some troubles with monitoring the data. It was harder to interpret/retrieve data with monitors in Salabim compared to e.g. MESA as the timestep is variable and the output is not a dataframe that we are used to. We did not have the time to figure the monitoring out, and now only have the simple timelogs.


TU Delft