Simulation-Based Data Generation for On-Demand Bus-as-a-Service Analytics

Vânia Sousa[1,2], Martim Santos[1], Bruno Silva[1], Yanelys Llerena[2], Tiago Sá[3], António C. Vieira[1(✉)], and Maribel Y. Santos[1]

[1] ALGORITMI Research Centre, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal
avieira@dsi.uminho.pt

[2] CCG/ZGDV Institute, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal

[3] Caetanobus - Fabricação de Carroçarias S.A, Vila Nova de Gaia, Portugal

Abstract. On-demand mobility systems, such as bus services, are gaining importance due to their role in sustainable urban transport. Their flexible and context-specific nature poses challenges for planning, optimization, and performance evaluation, particularly due to the scarcity of related operational data needed to support data-driven decision-making. This work is motivated by a real-world Bus-as-a-Service (BaaS) project, where the lack of available data limited the development of analytical tools. To address this gap, this paper proposes an artifact for the generation of event data (i.e., operational data) related to an on-demand bus service. Thus, the main contributions of this paper include the artifact tailored for on-demand bus services, but also the methodology that was followed for this instantiation. The purpose is that such methodology can be replicated in other cases to generate synthetic data. Without access to real data and without the possibility of gathering data from an existing on-demand bus service, the first step consisted of gathering customers' requirements and analyzing similar business models, resulting in the definition of an Entity-Relationship Diagram that allowed us to identify the data that needed to be generated by the artifact. This artifact simulates the required operational processes, capturing critical elements such as passenger requests, vehicle routing, and other service constraints. The results show that this artifact produces valid simulations for multiple scenarios and enables the generation of structured datasets that meet predefined analytical requirements. Thus, the obtained datasets can be consumed by data-driven analytics.

Keywords: Data analytics · Event data · On-demand bus service · Discrete-event simulation · Synthetic data generation

1 Introduction

Process simulation provides a structured approach to capturing the operational dynamics of complex systems, enabling a comprehensive representation of workflows, decision logic, and business rules [9]. Furthermore, simulation also enables

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16759, pp. 327–343, 2027.
https://doi.org/10.1007/978-3-032-30518-3_21

the analysis of complex and dynamic domains, such as healthcare, manufacturing, or transportation [10,13]. Thus, simulation offers not only an understanding of how workflows and systems function but also serve as a foundation for further analysis.

Apart from the aforementioned benefits, simulation also allows one to generate synthetic data regarding events that occur within the emulated environment, which are the result of the modeled processes' dynamic execution [9,11,13]. This capability can be valuable, particularly for domains where data is scarce, inconsistent, costly to obtain, or nonexistent. In fact, this is even more useful when combined with the capacity of simulation to test what-if scenarios, meaning such artifacts can be used to generate synthetic data, even of hypothetical scenarios.

The motivation for this work stems from a real-world project integrated into a Bus-as-a-Service (BaaS) platform that needs to support analytics for on-demand bus systems. One of the main goals of the project is to allow analytics, even though no data can be used since the platform is still under development.

On-demand bus services, a subset of demand-responsive transport, operate without fixed routes or schedules, dynamically adjusting to passenger requests. These systems aim to combine the efficiency of mass transit with the flexibility of ride-hailing, offering a more sustainable and accessible alternative to private car usage. By harnessing digital technologies such as route optimization algorithms and mobile platforms for booking and coordination, these services support the digital transition in urban mobility. At the same time, they contribute to the green transition by reducing unnecessary mileage, lowering emissions, and encouraging shared travel. Typically, users submit trip requests through an app. Based on aggregated demand, tailored routes are planned that group compatible journeys, enabling buses to operate with higher occupancy and reduced environmental impact compared to traditional fixed-route systems.

Due to the absence of data that supports analytics for this domain, there is a need for an artifact that allows such data to be generated and analyzed. In light of this, the objective of this paper is to propose an artifact designed to simulate the operational dynamics of on-demand bus services, with the main purpose of generating synthetic data pertaining to the simulated events. Furthermore, this paper also proposes the methodology that was followed to establish the requirements of the data that needed to be generated, so that applications in other domains can also instantiate it in their projects. Regarding the main contributions of this paper, they include the generated datasets suitable for downstream analytical applications, including data for multiple scenarios. These datasets include operational variables such as waiting times, demand, service coverage, and resource utilization, all of which are essential for informed decision-making in as on-demand bus service analytics.

This paper is structured as follows. Section 2 analyzes related works in the literature. Section 3 describes the proposed methodological approach that was followed to develop the herein proposed artifact. Section 4 describes the main development activities that were conducted, namely the data requirements elicitation approach, the development of the simulation model, and its validation.

Section 5 explores the generated datasets in descriptive analytics. Section 6 discusses the main conclusions, limitations, and future research directions.

2 Related Work

The scarcity of data in certain domains presents a challenge across multiple disciplines, including Healthcare, Artificial Intelligence (AI), and Social Sciences [4,5,9]. Such a lack of sufficient and high-quality data may severely limit the ability to conduct research, develop solutions, and make informed decisions. To address this challenge, the generation of synthetic data has gained significant attention. Among the most prominent approaches are those based on AI and Machine Learning (ML), which have demonstrated remarkable success in creating datasets that mimic the statistical properties of real-world data [1]. These methods are particularly effective in domains where large volumes of data are available, such as computer vision, natural language processing, and fraud detection. However, they often fall short when there is no data from which to learn patterns, as well as in capturing the dynamic, event-driven nature of many real-world systems [2,13]. These limitations have spurred the exploration of alternative methods, such as simulation, which offer unique advantages for generating synthetic and event-based data in data-scarce domains.

Riess [7] proposed a Python-based simulator that randomizes Business Process Model and Notation-style process models and provides rich temporal parameters, such as durations or inter-arrival times. However, it lacks operational details such as resource pools, capacity constraints, queues, and waiting times. Consequently, while control-flow and timing constraints can be effectively modeled, the tool is less effective in analyzing—and hence generating such data—related to resource utilization, bottlenecks, or service-level phenomena.

Skydanienko et al. [8] addressed the limitations of using real-life event logs to evaluate process mining algorithms, highlighting issues such as data privacy, lack of control over log parameters, and difficulty in isolating specific algorithmic behaviors. The author introduced a tool that generates artificial event logs, supporting constraints over both control flow and data. This tool tackles key challenges in declarative log generation by efficiently producing diverse traces using the Alloy model checker, enabling more robust and scalable algorithm evaluations. While resources may be included as static attributes (e.g., performed by), there is no token-based or queue-based dynamic monitoring. Thus, it cannot model phenomena such as workload or waiting time.

Andrews et al. [2] addressed the challenges of generating high-quality event logs for process mining from relational databases, highlighting common data quality issues and the lack of systematic, automated support for log extraction. It introduces a semi-automated, quality-informed tool that assesses source data, supports mapping to event log attributes, and generates logs suitable for process mining. The work aimed to lay the groundwork for a broader process data quality lifecycle, covering detection, repair, and tracking of quality issues. For this purpose, this work introduced an interactive loop to inject noise until

fitness/precision targets are met. The tool focuses on the semi-automated extraction of event logs from relational databases, guided by data quality dimensions and database constraints, rather than injecting synthetic logging errors. Furthermore, the tool does not simulate process executions, as its purpose is to support systematic event log generation from static data.

Li et al. [6] proposed a tool that learns from real logs using a Generative Adversarial Network with a transformer generator to output synthetic event data. While the training data may contain resource identifiers, the tool is agnostic in terms of considering capacity constraints, such as queue lengths, or resource policies, which are crucial to maintaining quality data in terms of being capable of representing attributes relevant to understanding the dynamic nature of the system, such as queue length or service levels.

As the above analysis of related works shows, despite AI- and ML-based approaches having achieved notable success in replicating the statistical properties of real-world data, they often rely on large training datasets and are often limited in capturing dynamic, event-driven behaviors inherent to many operational systems. Alternative approaches, such as process-oriented simulation, are capable of generating event-based data under controlled and configurable conditions. However, they frequently lack support for operational constructs such as resource contention, queues, and service dynamics, which are essential to accurately model a given system. Furthermore, such tools focus primarily on control flow and data constraints or on extracting logs from static relational sources, without simulating the execution of processes. These limitations highlight a gap in current solutions, i.e., the need for a synthetic data generator that can simulate realistic operational dynamics, including timing phenomena, queuing monitoring, and resource utilization, in order to support data-driven experimentation and analysis in contexts where real data is unavailable or insufficient. The simulation artifact proposed in this paper aims to address this gap. Nevertheless, to identify the required data it is necessary to follow specific steps, which are detailed in the next section.

3 Methodology for Synthetic Data Generation

This section describes the methodology that was adopted to develop the herein proposed artifact for data generation of an on-demand bus system. The main phases that comprise this methodology are illustrated in Fig. 1.

As the figure shows, the first step consisted of gathering the requirements in terms of data that the simulation should be able to generate. For this purpose, two aspects should be stressed. First, there is no data to gather information regarding how the system should operate. Second, as such a system is not yet operating, it is also not possible to gather data by alternative means, e.g., by field observations, interviews, business process models' analysis, or other techniques. Therefore, while traditional requirements elicitation for simulation models comprises some of the mentioned techniques, other means may need to be pondered.

The second step consisted of developing the simulation model. This can be done with any commercial tool (e.g., Simio [12,13]) or by programming the

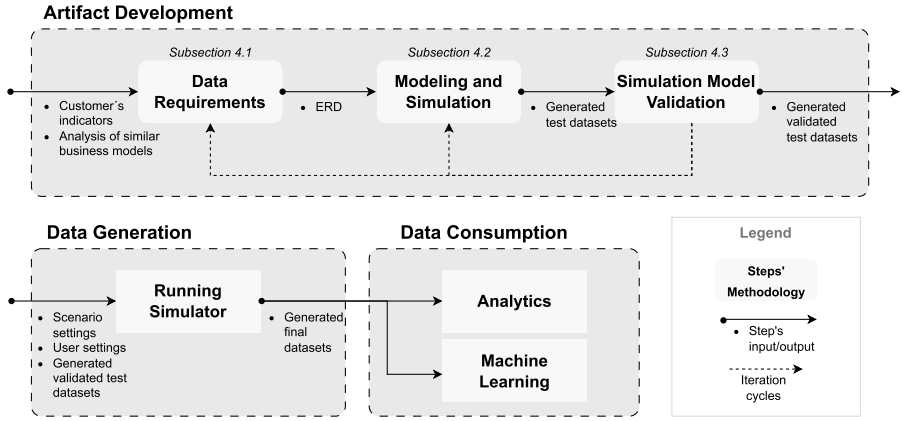

Fig. 1. Methodology followed for the development of the proposed artifact.

simulator (e.g., with Python). The purpose would be to develop models that replicate the behavior of business processes and allow their dynamic simulation, properly accounting for the evolution of resources, queues, and overall system state throughout time. By doing so, it should be possible to generate event-based data related to the events that occur during the simulation.

Having developed the simulation model, it is ready to generate data. However, before considering these data ready to be consumed, the simulation model should be validated, as it could still contain inconsistencies that could be producing data without the required quality. Traditional simulation studies emphasize the importance of simulation model validation, suggesting several approaches, such as statistical analysis, visualization of resulting animation, and comparison of obtained and expected outputs, among others [3,13]. Notwithstanding, due to the importance of obtaining data consistent with the operations of the intended system, alternative means can be pondered, namely, the creation and analysis of visualizations obtained from the generated data. Such visualization could allow one to identify operational inconsistencies, e.g., bus stops without passengers or demand higher than expected. Furthermore, to ensure that the withdrawn conclusions do not apply only to the scenario being tested, it is also important to adapt the validation strategy for multiple scenarios that were previously established. This ensures a higher confidence in the robustness of the simulation model to produce reliable and hence valid synthetic datasets.

Once the simulation model is validated, so are the generated datasets ready to be consumed in the respective layer. On this note, data can be consumed by ML applications, descriptive analytics, or any other analytical approach (e.g., AI, optimization). To validate the proposed methodology and the resulting artifact, synthetically generated datasets will be used for descriptive analytics.

4 Data Generation for On-Demand Bus-as-a-Service Analytics

This section describes the work that was conducted to instantiate the methodology described in the previous section. Concretely, first, the requirements are detailed, which consist of the proposal of the Entity-Relationship Model (ERD) for this system, thus establishing the main concepts that the simulation must contain, as well as the respective data. The next subsection describes how the simulation model was developed. The last subsection covers the validation process that was followed to validate the simulation model.

4.1 Data Requirements

This section describes how the main requirements, in terms of data that needed to be generated, were gathered. As per the methodology illustrated in Fig. 1, first, the requirements in terms of indicators that the project's customers identified as most relevant are gathered. In other words, the generated data should allow such indicators to be analyzed. However, per se, this would not allow one to understand how the on-demand bus system should operate. As such, this approach is complemented by benchmarking other similar business models (e.g., VitrusBus[1], a bus-on-demand service operating in Guimarães, Portugal). Such analysis, along with a careful validation with the project stakeholders, allowed us to identify relevant indicators, as well as specific attributes required to compute such indicators. Table 1 summarizes this mapping, between indicators and attributes, showing how the simulation model will enable the computation of each indicator required by the project's stakeholders.

The decomposition of relevant business indicators into the necessary data attributes, listed in Table 1, allowed us to organize such data in an ERD, as depicted in Fig. 2. As can be seen, six entities were considered, namely `Customer`, `Stop`, `Booking`, `Planned_Route`, `Bus`, and `Trip`.

This also allowed us to extract some relevant insights, such as events that need to be simulated, e.g., customer needs to be able to book trips, or buses need to perform specific trips, which depletes their battery. In fact, a customer can make one or more bookings, each of which includes a requested origin and destination stop, as well as the desired date and time. Each booking is associated with a planned route that aggregates similar requests from different customers. In turn, a planned route can be executed multiple times as separate trips. Each trip represents the operational realization of a planned route and includes actual data, such as execution time and distance traveled. Stops are defined as the physical locations for boarding and alighting, and each booking specifies a bus stop of an origin and another of a destination. The cardinalities between entities reflect these relationships: a one-to-many relationship exists between customers and bookings, bookings and planned routes, and planned routes and trips. Additionally, each booking has two many-to-one relationships with the stop entity: one

[1] https://vitrusambiente.pt/vitrusbus/.

Table 1. Mapping between the indicators given by the project's customer and the identified attributes.

ID	Indicator Description	Relevant Attributes
I1	Total number of bookings	Booking.Booking_ID
I2	Total number of customers	Customer.Customer_ID
I3	Average trip distance	Trip.Distance
I4	Booking trends by date	Booking.Booking_Timestamp
I5	Average trip duration	Trip.Actual_Start_Time, Trip.Actual_End_Time
I6	Main customers (by number of bookings)	Booking.Customer_ID, Customer.Name
I7	Times with highest volume of bookings	Booking.Requested_Time, Booking.Requested_Date
I8	Total active customers	Customer.Customer_ID, Booking.Booking_Timestamp
I9	Total number of trips	Trip.Trip_ID, Bus.Bus_ID
I10	Total battery consumption	Trip_Energy.Consumption_KWh
I11	Energy consumption by route	Trip_Energy.Consumption_KWh, Planned_Route.Route_ID
I12	Route usage map (Origin → Destination)	Planned_Route.Planned_Route_ID, Route_Stop.Stop_ID, Trip.Trip_ID
I13	Bus Utilization and Energy Consumption Analysis	Bus.Bus_ID, Trip.Trip_ID, Trip.Passengers_Onboard, Trip_Energy.Consumption_KWh
I14	Hourly Battery Consumption by Bus	Bus.Bus_ID, Battery_State.Level_Percent, Battery_State.Timestamp
I15	Daily evolution of bus trips	Bus.Bus_ID, Trip.Trip_ID, Trip.Date
I16	Customer segmentation by booking frequency	Customer.Customer_ID, Customer.Name, Booking.Count
I17	Bookings vs travels over time	Booking.Booking_ID, Booking.Booking_Timestamp, Trip.Trip_ID

for the origin and one for the destination stop. It should also be noted that, in addition to the data attributes identified during the requirements gathering phase, additional ones can be generated and calculated as a result of the dynamic execution of the simulation model. Some examples include the requests' waiting times, bus utilization rates, the number of passengers queuing to board the bus per stop, among others.

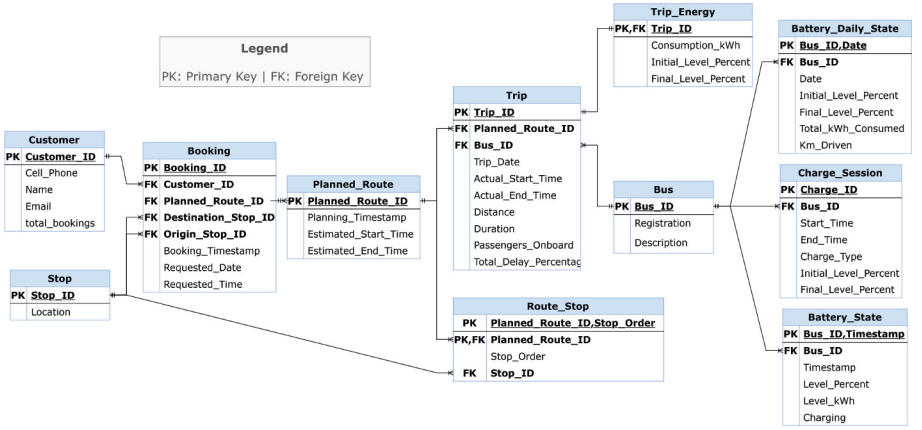

Fig. 2. Relationship Diagram associated with the elicited data requirements for the simulation-based generator.

4.2 Modeling and Simulation

This subsection describes the development of the simulation model that was used to develop the on-demand bus system. The simulation model was implemented in `Python` version 3.12.4[2], namely using the `SimPy` library version 4.1.1[3]. In addition, other libraries were also used, from which the following can be highlighted. `NetworkX`[4] was used to generate the graph that specifies the locations of the bus stops, as well as the distance between them and the different routes that allow buses to travel within the simulation model. In turn, `Faker`[5] was used to generate static contextual data, such as customer names or their contact details. Algorithm 1 provides an overview of the simulation model implemented, by presenting the respective pseudo-code. Initially, the simulator reads a set of parameters from user inputs. While several were implemented, the ones listed in Table 2 can be highlighted.

When the simulation begins, some elements need to be created. First, a list of stops, as well as the respective complete graph. This graph contains a central point to be used as a dockyard from which all buses leave and return after finishing a route. Thereafter, a bus fleet is also created. Finally, other data structures are also created, of which the set of requests can be emphasized, which will serve as a pool for all bus service requests. These requests will act as the main entities of this simulation model.

Afterwards, the main loop in the simulation can initiate, which runs until the simulation clock (specified by the `simtime` variable) reaches the user-defined simulation length parameter (`SIM_LENGTH`). At the start of each new day (every

[2] https://www.python.org/downloads/release/python-3124/.
[3] https://simpy.readthedocs.io/en/4.1.1/contents.html.
[4] https://networkx.org/.
[5] https://faker.readthedocs.io/en/master/.

Table 2. Examples of simulation parameters that can be edited by users.

Symbol	Description
SIM_LENGTH	Duration of a simulation run in minutes
NUM_BUSES	Number of buses used in a simulation run
DEMAND	Expected average number of requests per hour
NUM_STOPS	Number of bus stations. Some stops are more popular
SCALE	Defines the area for the simulation
AT	Maximum time for next bus service in minutes

1440 min), the system plans routes for all pending received requests, with the *plan_routes_for_day(requests)* method. This planning is necessary in order to organize the received requests into future bus routes. This starts by filtering the unprocessed requests scheduled for the current day and sorting them by their scheduled time. Bus routes are then created by combining multiple requests with similar origins, destinations, and compatible schedules. These route groupings are defined by taking into account a few constraints, e.g., not exceeding the defined bus capacity and the maximum expected route duration of 4 h (this estimation can be altered by the user). This process also ensures that no routes are defined with less than two stops. When the planning is finalized, the `logs` function is used to save the required data for later export, in this case into CSV files.

Throughout the simulation, new requests are generated at discrete intervals in time, following an Exponential distribution with an average value related to the expected number of passengers per hour (specified by the `DEMAND` parameter). A randomly selected set of origin and destination bus stops is assigned to these requests, as well as a desired schedule for their bus travels and the associated number of passengers. Finally, such data is also saved with the `logs` function for later export.

When a route's scheduled departure time matches the current simulation time, the route is executed, i.e., the route is assigned to a bus, and it travels to the next bus stop at a randomly assigned speed, following a Uniform distribution with a minimum of 0.83 and a maximum of 1.17 km per minute. In its turn, the travel time is calculated by the *travelDuration(bus.speed, simtime)* method, which considered the speed assigned to the bus, as well as on the current simulation time, thereby allowing to account for the traffic intensity. Moreover, the bus battery is also consumed during each bus travel. To verify if the bus needs to charge, the *charge_check(bus, simtime)* method monitors the battery, as well as the current simulation time, to decide whether the bus should charge, i.e., in periods of higher demand, the battery charging policy changes. In order to allow the simulation clock to advance to the time in which each bus arrives at the intended bus station (next event), the `wait(event)` function mimics such behavior. At each stop, the simulation models (with the `delay()` method) pas-

senger boarding (0.03 min per passenger) and exiting (0.02 min per passenger) the bus before proceeding to the next stop.

Algorithm 1. Pseudocode of the bus simulation implemented in SimPy.

Inputs: SIM_LENGTH, NUM_BUSES, DEMAND, NUM_STOPS, SCALE, AT

Initialization:
    stops ← Create list of stops based on NUM_STOPS
    network ← Create the network of bus stops, considering SCALE
    buses ← Create set of buses based on NUM_BUSES
    requests ← Create empty set of bus requests
    routes ← Create empty set of routes *// including attributes, e.g., battery level*

while simtime < SIM_LENGTH **do**
    if wait(**simtime** % 1440 == 0) **then** *// Start of a new day*
        routes = plan_routes_for_day(requests) *// Planning for pending requests*
        logs(routes) *// Save data logs for later export*
    end if
    if wait(**simtime** == nextRequest) **then** *// Create new request*
        $request.scheduledTime \leftarrow$ **simtime** $+ \mathcal{U}(1440, AT)$
        $request.numPassengers \leftarrow$ if $\mathcal{U}(0,1) \leq 0.5$ then 1 else $\mathcal{U}2, 4$
        $interarrival \leftarrow \mathcal{E}(60/DEMAND)$ *// When to create next request*
        $request.destination \leftarrow$ random bus stop
        $requests.add(request)$
        $logs(request)$ *// Save data logs for later export*
        $nextRequest \leftarrow$ **simtime** $+ interarrival$
    end if
    for each (bus, route) in $(b, r)|b \in buses, r \in routes$ **do**
        if wait(**simtime** == scheduledTime) **then** *// Start new bus route*
            for each stop in route **do**
                $bus.speed \leftarrow \mathcal{U}(0.83, 1.17)$ km per minute
                $bus.travelTime \leftarrow travelDuration(bus.speed,$ **simtime**$)$
                bus.move($stop$) *// Bus travels to the next stop*
                $bus.consumeBattery()$ *// Dynamic consumption of battery*
                $charge_check(bus,$ **simtime**$)$ *// Verifies if the bus need to charge*
                wait(stop) *// Wait for bus to arrive at next stop*
                delay() *// Delay for each passenger boarding or exiting the bus*
                logs(route, bus) *// Save relevant route indicators*
            end for
        end if
    end for *// Go to time of next discrete event*
end while *// End simulation and save data logs*

Finally, relevant indicators, such as waiting time and travel durations, are calculated, and relevant data is saved for later export with the `logs` function. Since SimPy is a process-oriented, discrete-event simulation framework, the next event to simulate is determined based on the simulation time when the event was

scheduled. I.e., at this point, the simulation engine decides whether new routes should be planned, new requests generated, or bus routes modeled.

After completing all stops, the bus returns to the dockyard. The simulation clock advances in discrete time steps until reaching the end of the simulation, when all data events are saved to the respective files. This process captures the logic while handling stochastic demand and travel time variability. Therefore, between different simulation runs, different datasets may be generated. However, studying this stochasticity falls outside the scope of this research. The next subsection covers the validation process that was conducted.

4.3 Simulation Model Validation

This subsection describes the validation of the simulation model, which was carried out to ensure that the model effectively represents the domain, with the ultimate goal of ensuring that the generated data are realistic and useful for analysis. The validation process consisted of analyzing the visualizations obtained from the generated data in order to analyze the coherence of different test scenarios. As per the methodology proposed in Fig. 1, first, a set of scenarios is defined, which guides this validation process. For all scenarios, the following default parameter values were used: maximum antecedence of 1 week (AT) for the scheduling of requests, SCALE of 35 and 20 bus stops (NUM_STOPS); parameters defined in Table 2. In turn, for the defined scenarios, the following parameters are altered:

- **Scenario 1:** The demand parameter is set to 5 and the number of buses is set to 5. Obtained visualizations are depicted in Fig. 3.
- **Scenario 2:** The demand is increased to 20, and the number of buses is set to 10. Obtained visualizations are depicted in Fig. 4.

Both figures confirm that the generated data behaves as expected, exhibiting realistic patterns consistent with the modeled transport process.

The top-left graphs of both figures indicate that the route planning process operates efficiently, generating routes that service multiple requests simultaneously. While, indeed, a planning process was implemented which may considerably affect the operation of the entire simulation, this falls out of the scope of this paper. As expected, the number of requests served per route increases significantly in correlation with higher demand, since the number of buses remain unchanged from one scenario to the other. The top-middle graphs demonstrate that total travel time remains within a defined limit, due to the battery of the buses. As can be seen, under higher demand, the trend is for additional routes being required to satisfy the higher number of passengers with a higher variety of requests. The same can be concluded from the top-right graphs of both figures. In turn, the bottom-left graphs validate that the passenger capacity limit is respected. Furthermore, the increase in demand in scenario 2 shows a realistic increase in passenger load per bus. The bottom-middle graphs, depicting the distribution of distances traveled per bus, reveals a uniform workload across the

fleet. This balanced distribution further suggests that the scheduling algorithm optimizes resource allocation effectively, avoiding the over-utilization of specific vehicles. Finally, the bottom-right graphs illustrate the evolution of requests over time. The graphs differentiate between received, processed and rejected requests, since customers need to schedule their trips, which need to be processed, i.e., allocated to a bus and a route (as per the mentioned planning process).



Fig. 3. Results of scenario 1.

Regarding the rejected requests, this pertains to situations, such as requests that would originate routes consuming too much battery, or simply not enough bus capacity. In some of these situations, customers can simply wait on a queue, however, in some cases they are considered as rejected. The relatively low volume of rejected requests in scenario 1 underscores the effectiveness of the algorithm responsible for allocating the requests to bus routes, and, as expected, under higher demand this number also slightly increases. It is also possible to see that the system achieves a steady state, in what concerns the three variables in question, after roughly one week.

In conclusion, the comparative analysis of these scenarios demonstrates that the simulator is both robust and highly responsive to parameter adjustments. By maintaining operational constraints while achieving a stable steady state, the model proves to be a reliable tool for testing different strategies in on-demand bus transport, providing a realistic foundation for dataset generation and consequent analytics.



Fig. 4. Results of scenario 2.

5 Results

This section proposes to evaluate the effectiveness of the proposed methodology, the resulting artifact, and the datasets it generates by presenting a dashboard to allow descriptive analytics over the generated datasets. The dashboard allows the visualization of the indicators that had been previously identified during the requirements elicitation phase, since such indicators were deemed as relevant for the project's customers. As such, the dashboard was implemented as a proof-of-concept to demonstrate that the synthetic data can effectively support analytical tasks, particularly descriptive analytics. For this purpose, this section displays two different dashboards: one that integrates indicators related to customers and their respective requests (Fig. 5) and another that integrated indicators related to routes and buses (Fig. 6).

The developed dashboard comprises different visualizations, each corresponding to an indicator elicited during the requirements phase. The first visualization displays the total number of bookings (I1), providing an overview of the volume of service usage during the simulation period. The second shows the number of unique users (I2), reflecting the diversity of the user base. The third visualization presents the average distance per trip (I3), offering insights into travel behavior and typical route lengths. The fourth visualization highlights booking trends over time (I4), allowing the identification of usage peaks and seasonal patterns. The fifth graph focuses on the average duration of planned routes (I5), which supports the analysis of operational efficiency. The sixth chart lists the top customers by number of bookings (I6), helping to identify frequent users or institutional clients. Finally, the seventh visualization shows the distribution of bookings across time slots (I7), enabling the detection of peak hours and

demand concentration. In turn, Fig. 7 and Fig. 8 shows that it is also possible to drill down on a route that a user may select. Overall, the results provide empirical evidence that the adopted methodology leads to the generation of usable, structured data aligned with analytical needs.

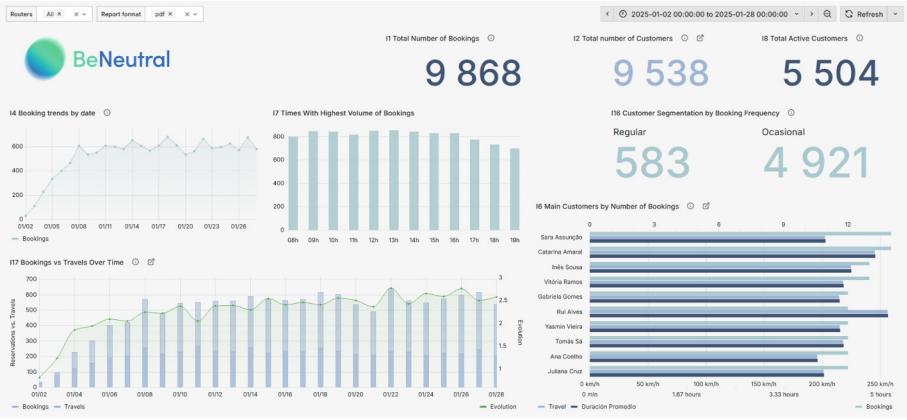

Fig. 5. Dashboard integrating indicators related to customers and their requests.

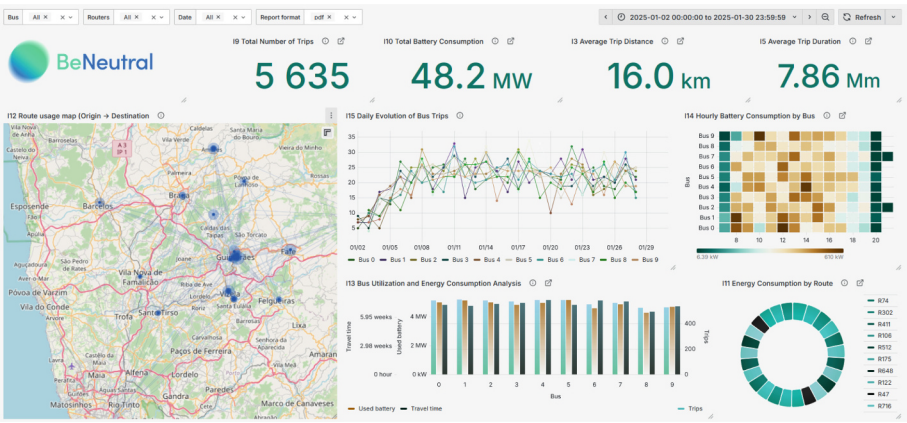

Fig. 6. Dashboard integrating indicators related to routes and buses.

The first dashboard (cf. Fig. 5) reveals a predominantly occasional user base, with regular customers representing a small fraction of the total. Booking trends show an initial ramp-up consistent with simulation cold-start effects, stabilising into a steady daily demand pattern thereafter. The hourly distribution suggests

a mild midday concentration without strongly pronounced peaks. The bookings versus travels chart highlights a persistent gap between reservations and completed trips that gradually narrows over time, reflecting the route planning mechanism adapting to accumulated demand.

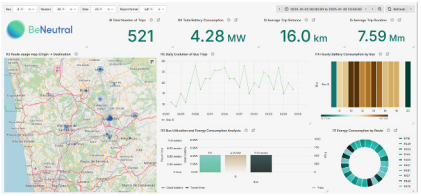

Fig. 7. Dashboard of Fig. 5 filtered by bus.

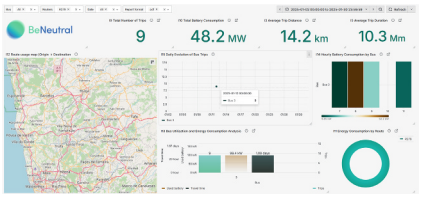

Fig. 8. Dashboard of Fig. 5 filtered by route.

The second dashboard (cf. Fig. 6, Fig. 7 and Fig. 8) shows a well-balanced fleet utilisation, with trip counts and energy consumption distributed evenly across buses throughout the simulation period. An initial stabilisation phase is visible in the first days of operation, after which each bus reaches a consistent activity level. The hourly battery consumption heatmap confirms that energy usage is concentrated within operational hours, with some buses exhibiting higher midday consumption. At the route level, a small subset of routes accounts for higher energy consumption, pointing to differences in route length or demand intensity. The filtered dashboards demonstrate the system's drill-down capability at both bus and route level. Individual bus metrics align closely with fleet averages, confirming consistent behavior across the fleet without systematic assignment bias. Route-level filtering exposes the heterogeneity of the network, with less frequent routes showing distinct distance and duration profiles compared to the overall average.

6 Conclusions

This work addressed the pressing challenge of data scarcity in emerging mobility systems, particularly in the context of on-demand bus services. In the absence of real-world operational data - critical for supporting analytics and decision-making - as well as a real operating system from which data could be gathered, this paper proposed and implemented an artifact capable of generating high-quality synthetic data. The artifact is based on process modeling and simulation, meaning that business processes are modeled and can be simulated, thus considering essential components and dynamics of on-demand bus systems.

The main contributions of this work comprise not only the proposed artifact but also the systematic methodology that guided its construction. Grounded on traditional simulation development methodologies, it extends such methodologies by considering essential steps when aiming to generate high-quality data,

especially without the possibility of gathering data. Namely, the data-driven requirements elicitation process fostered the project stakeholders, which, along with an analysis of similar business models, allowed us to identify attributes that the data should contain. The methodology also reckons that the validation of the simulation model should consider the generated datasets, as well as the assessment of the respective visualizations, which allow for a thorough examination of the operational details that occur throughout the simulation.

One of the advantages of the approach proposed and implemented in this paper lies in its independence from historical data. Unlike methods such as Machine Learning, or generative models (e.g., Large Language Models), which rely on large volumes of data, simulation enables the explicit modeling of processes based on operational rules and process logic. This is valuable in scenarios where data is unavailable or limited, as is the case in early-stage or conceptual systems. Moreover, simulation allows the configuration and evaluation of "what-if" scenarios.

In terms of future work, several main avenues can be pursued. First, key decision processes occurring during simulation runtime, such as charging scheduling, route planning, and fleet dispatching, could be refined through more sophisticated optimization strategies, replacing the current rule-based approaches with adaptive or learning-based mechanisms. Second, the integration of sustainability-oriented indicators, such as carbon emissions and energy efficiency metrics, would broaden the analytical scope of the framework beyond operational performance. Third, the data generated by the simulation should be exploited by machine learning applications, enabling the training of predictive models for demand forecasting, anomaly detection, or route optimization. Fourth, while the simulation produces reliable data, it still warrants the investigation of more flexible generative approaches, since any modification requires explicit rule re-implementation. Finally, the methodology proposed in this paper should also be tested in other cases that require synthetic data that can be generated by simulating process models, e.g., logistics or healthcare.

Acknowledgements. This work has been supported by the European Union under the NextGenerationEU, through a grant of the Portugal's Recovery and Resilience Plan (RRP) Partnership Agreement, within the scope of the project BE.NEUTRAL – Agenda da Mobilidade para a neutralidade carbónica das cidades (Project ref. nr. 35 - C644874240-00000016).

References

1. An, G., Cockrell, C.: Generating synthetic multidimensional molecular time series data for machine learning: considerations. Front. Syst. Biol. **3**, 1188009 (2023)
2. Andrews, R., van Dun, C.G., Wynn, M.T., Kratsch, W., Röglinger, M., ter Hofstede, A.H.: Quality-informed semi-automated event log generation for process mining. Decis. Support Syst. **132**, 113265 (2020)
3. Banks, J.: Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice. Wiley (1998)

4. Borovits, N., Bardelloni, G., Hashemi, H., Tulsiani, M., Tamburri, D.A., van den Heuvel, W.J.: Addressing data scarcity with synthetic data: a secure and GDPR-compliant cloud-based platform. ACM Trans. Softw. Eng. Methodol. (2025)
5. Chen, Y., Kempton, D.J., Ahmadzadeh, A., Wen, J., Ji, A., Angryk, R.A.: CGAN-based synthetic multivariate time-series generation: a solution to data scarcity in solar flare forecasting. Neural Comput. Appl. **34**(16), 13339–13353 (2022)
6. Li, K., Yang, S., Sullivan, T.M., Burd, R.S., Marsic, I.: ProcessGAN: generating privacy-preserving time-aware process data with conditional generative adversarial nets. ACM Trans. Knowl. Discov. Data **18**(9), 1–31 (2024)
7. Riess, M.: SynBPS: a parametric simulation framework for the generation of event-log data. SIMULATION **100**(8), 849–870 (2024)
8. Skydanienko, V., Di Francescomarino, C., Ghidini, C., Maggi, F.M.: A tool for generating event logs from multi-perspective declare models. In: Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018, vol. 2196, pp. 111–115 (2018)
9. Sousa, V., Vieira, C., Lavalle, A., Vieira, A., Santos, M.Y.: Integrating business and process analytics for enhanced data-driven decision-making. In: 43rd International Conference on Conceptual Modeling: ER Forum, Special Topics, Posters and Demos, ER-Companion 2024 (2024)
10. Taleb, M., Khalid, R., Ramli, R., Nawawi, M.K.M.: An integrated approach of discrete event simulation and a non-radial super efficiency data envelopment analysis for performance evaluation of an emergency department. Expert Syst. Appl. **220**, 119653 (2023)
11. Vieira, A., Dias, L., Pereira, G., Oliveira, J., Carvalho, M., Martins, P.: Automatic simulation models generation of warehouses with milk runs and pickers. In: 28th European Modeling and Simulation Symposium, EMSS, pp. 231–241 (2016)
12. Vieira, A., Dias, L., Pereira, G., Oliveira, J.: Comparison of Simio and arena simulation tools. In: 12th Annual Industrial Simulation Conference, ISC, pp. 5–13 (2014)
13. Vieira, A.A., Figueira, J.R., Fragoso, R.: A multi-objective simulation-based decision support tool for wine supply chain design and risk management under sustainability goals. Expert Syst. Appl. **232**, 120757 (2023)