# Predicting Failures in Smart Human-Centric EcoSystems

Anonymous Author(s)

## Abstract

Smart cities, smart grids, and in general *Smart human-centric EcoSystems* (*SES*) emerge from the co-existence of many systems that operate according to independently owned specifications, and evolve over time. *SES* may fail despite the correct behavior of the systems that comprise the *SES*. Fully testing *SES* on testbed to prevent failures in production is impossible, and scenarios that may lead to catastrophic *SES* failures are unavoidable.

In this paper we frame the core issues of *SES* failures, and propose *Smart human-centric Ecosystem Monitoring*, *SEM*, an approach that predicts *SES* failures to enable corrective actions for mitigating the catastrophic effects of failures. *SEM* identifies failure-prone scenarios from the reconstruction error of *SES* indicators, that is, metric values that *SEM* collects from *SES* at constant frequency. *SEM* computes the reconstruction error with a suitably trained denoising autoencoder combined with a Transformer architecture. The results of experimenting with a peer-to-peer ride-sharing ecosystem operating in San Francisco confirm that *SEM* can effectively predict *SES* failures early enough to activate preventing actions, and indicate the generalizability of *SEM* with continual learning.

## 1 Introduction

In this paper, we present a technique to predict failures in *Smart human-centric EcoSystems (*SES*)*, the important class of systems that El Moussa *et al.* characterize as *'multifaceted systems that emerge from the composition of independently-operated and autonomous systems with smart functionalities and that evolve over time*' [45]. *SES* are ecosystems composed of systems that operate autonomously, according to requirements defined by independent entities, and no central ownership, and consequently with no complete specifications of *SES* as a whole. *SES* have been studied since the late nineties as *virtual systems of systems* [42], *ultra-large scale systems* [49], *large-scale complex IT systems* [60], and *cyber-physical systems* [19]. *SES* are widely spread in many domains. Smart cities [28] smart grids [3], global cyber-infrastructures for healthcare [61], e-markets, e-commerce platforms [25], telecommunication infrastructures [30], and peer-to-peer ride-sharing systems [2, 40, 50] are relevant examples of *SES*.

The absence of central ownership and complete specifications challenges the classic notion of correctness defined as compliance with specifications. El Moussa *et al.* capture the intuitive notion of *'expected behavior of* SES *'* with the concept of SES *health* that they define *'along three non-orthogonal dimensions, hardiness, consistency, and resilience'*, where *'hardiness indicates the strength of a* SES *in terms of amount and quality of the provided services', 'consistency captures the ability of the* SES *to exhibit a stable behavior also in the presence of sudden and unpredictable events', 'resilience captures the ability of the* SES *to react to unavoidable crisis and go back to an acceptable behavior'* [45].

El Moussa *et al.* define SES *failures* as unacceptable degradations of *SES* health. Sommerville *et al.* observe that *SES* failures are unavoidable, and may occur even when all comprising systems behave according to their specifications [60], as well illustrated with *'Flash Crash'*, the U.S. equity markets crash of May 6, 2010, an exceptional stock market crash that caused a temporary loss of about 800 billion US dollars of market value, the cause of which cannot be attributed to a software bug, but rather to a combination of interactions that implicitly led to an unforeseeable scenario [23].

El Moussa *et al.* 's paper suggests a correlation between *SES* failures and anomalies of the energy of *SES* indicators, that is, quantifiable elements that capture significant aspects of the *SES*, and provides a simple example that illustrates the hypothesis.

In this paper, we refine the preliminary El Moussa *et al.* 's definitions of *SES* failures and propose a new approach to monitor *SES* health and predict *SES* failures. Early predicting *SES* failures can trigger fast automatic reactions to limit the impact of the failures, and offer timely information to the operators of the systems that comprise *SES* to activate recovery actions.

The main contributions of this paper are:

(1) We characterize *SES* failures as outstanding variations of some indexes that capture the ultimate perception of *SES* health, in analogy with the well-known indexes that characterize the status of complex systems, like the *Flash Crash* market value.
(2) We provide some relevant examples of *SES* failures, by referring to the case study of a ride-sharing *SES* that we use as a running example in this paper.
(3) We propose *SEM*, a *Smart human-centric Ecosystem Monitor* to detect health anomalies from the reconstruction error that we compute with a denoising autoencoder [64] combined with Transformer architecture [63]. Health anomalies signal degradations before otherwise unavoidable *SES* failures, and trigger recovery actions on any subset of systems in the *SES*.
(4) We present a prototype implementation of *SEM*, and discuss the results of a set of experiments that confirm the correlation between high reconstruction error of the autoencoder and *SES* health degradations that lead to *SES* failures if not handled. *SEM* predicts all failing scenarios half an hour on average and at least ten minutes before the failures, in all our experiments.
(5) We describe a digital mirror of a ride-sharing *SES* that we use for experimentally evaluating *SEM*, and that we offer to the community in a replication package to reproduce our experiments and comparatively evaluate new approaches.
(6) We report the results of experimenting with continual learning to generalize *SEM* in the presence of substantial variations in the dynamics of the *SES*. Our experiments indicate that *SEM* maintains its predictive abilities in the presence of significant evolutions of the *SES*, without requiring complete retraining.

The paper is organized as follows. Section 2 presents a precise definition of *SES* failures. Section 3 overviews the main contribution of this paper, *SEM*, an approach to monitor *SES* health and predict *SES* failures. Section 4 presents *SEM*, a Transformer-based anomaly detection approach to detect health anomalies in *SES*. Section 5 proposes *RS-Digital-Mirror*, the digital mirror of a peer-to-peer ride-sharing *SES* that we use in our experiments. Section 6 describes

the main research questions that we address in this paper, presents a set of experimental results on *RS-Digital-Mirror* that confirm the validity of the research hypotheses, and the generalizability of *SEM* with continual learning. Section 7 overviews the related work. Section 8 summarizes the results presented in this paper and indicates new research directions.

## 2 *SES* Failures

Studies of *SES* agree that failures are unavoidable (*'We can expect that, with ULS systems, unusual situations and boundary conditions will occur often enough that something will always be failing'* [49]), may occur independently from the correctness of the comprising systems (*'The cause of the Flash Crash cannot be attributed to a software bug, but rather to a combination of interactions that implicitly led to an unforeseeable scenario' [23]*), and escape classic approaches to correctness (*'it is impossible to specify the correctness of a* SES *in the presence of contradicting requirements of the autonomous systems'* [45]).

Manikas *et al.* introduce the concept of health to characterize the intuitive concept of *wellness* of software ecosystems [43], and El Moussa *et al.* characterize *SES* health according to three non-orthogonal dimensions, hardiness, consistency and resilience, and briefly introduce the concept of *SES* failure as unacceptable degradation of *SES* health [45]. We observe a significant difference between *SES* health anomalies and *SES* failures, analogous to the distinction in financial markets between anomalous market conditions, which can precipitate unforeseen profits or losses, and outright market failures, such as the *'Flash crash'*. We illustrate the differences between *SES* health anomalies and *SES* failures with a simple but representative example that we use as running example in the paper, a peer-to-peer ride-sharing *SES*.

A peer-to-peer ride-sharing *SES* emerges and evolves from the composition of autonomously owned and operated ride-managers (like *Uber* [13] and *Lyft* [34]), payment services [14], smart cars [4], smart GPS systems [12], and traffic alert systems [6], all interacting with passengers and drivers. The systems comprising the ride-sharing *SES* are owned by different entities, operate autonomously according to their own goals, and interact implicitly and unconsciously through shared resources and hidden actions with unavoidable contradictions and failures.

Unforeseen events like announcements of particularly long-lasting major underground failures or gigantic flash mobs may suddenly change the distribution of ride requests and duration of the rides, leading to temporary reductions of *SES* health, that is the amount and quality of the provided services. While in the many common situations observed so far in the field ride-sharing *SES* self-balance, an anomalous presence of *greedy drivers*, that is, drivers who decline invitations waiting for further increases of fares[1], can trigger exceptional amounts of declines, enormous increases in fares and delays, massive amounts of passengers giving up, and ultimately a *SES* failure, that is a giant percentage of unsatisfied requests. It is important to notice that these *SES* failures are due to implicit interactions and intrinsic contradictions of systems that behave correctly, according to their specification and goals, and people, who use the systems according to their needs.

[1]The trend of declining requests to increase fares is a hypothesis of eu.usatoday experts

Consistently with well-known complex systems, like financial markets, we characterize *SES* failures as sudden degradations of some indexes that quantify perceivable unacceptable degradations of *SES* behavior. The approach that we propose in this paper is parametric with respect to the indexes, which depend on the application domain, and capture some relevant aspects of the *SES*, similarly to stock exchange indexes that capture the overall market behavior. A drop of the stock exchange indexes below some thresholds are sometimes devastating market crashes[2]. Similarly, an increase of the *SES* indexes above some thresholds are *SES* failures.

In our ride-sharing *SES* example, we refer to a $failed\_requests$ index that captures the relevant global aspects of *SES*, and that we define as the ratio between the ride requests that are not served over all ride requests:

$$failed\_requests = \frac{rides\_not\_served}{requested\_rides} \tag{1}$$

where the value of $requested\_rides$ is the number of requests for rides, and the value of $rides\_not\_served$ is the number of rides ultimately not fulfilled by drivers, both computed with a rolling average over time windows of fixed length, five minutes in our experiments. Thus, the value of $failed\_requests$ captures the lack of service as the percentage of ride requests that are not served.

In general, the impact of percentages varies with the values, and the percentage of failed requests is not an exception: The relevance of the percentage of failed requests depends on the demand of rides. A degradation in the percentage of failed requests during periods of high demand, like at peak hours, is more severe than during off-peak times, like at night. We adjust the $failed\_requests$ index to different demands with a logarithmic factor that takes into account the average service requests that we observed during training, as commonly done when considering values with a high variance. When the number of requested rides ($requested\_rides$) is below a threshold that we fix as the average service requests that we observed during training ($avg\_rides\_requested$), we multiply the index value ($failed\_requests$) by a logarithmic factor $y$ (between 0 and 1) that we compute as the solution of the function $y = a \times \ln(requested\_rides + b) + c$, where the coefficients $a$ and $c$ are the solution of the following system of linear equations:

$$\begin{cases} a \times \ln(b) + c = 0 \\ a \times \ln(avg\_rides\_requested + b) + c = 1 \\ b = 0.1 \end{cases} \tag{2}$$

$b = 0.1$ to ensure the function is defined with $requested\_rides = 0$[3].

The value of the index $failed\_requests$ well captures the *hardiness* of the *SES*, since the quality of the ride-sharing *SES* dramatically decreases when the percentage of failed requests dramatically grows. The standard deviation of the index $failed\_requests$ well captures the *consistency* of the *SES*, since the stability of the ride-sharing *SES* dramatically decreases when the standard deviation of failed requests dramatically grows. The divergence distance of the moving average of $failed\_requests$ over the last minute (short-term) with respect to the moving average over the last thirty minutes (long-term) well captures the *resilience* of the ride-sharing *SES*,

[2]Like the *Black Friday* of 1929, a drop of 23% of the Dow Jones in two days https://www.irishtimes.com/business/personal-finance/black-friday-anniversary

[3]$requested\_rides$ is a Natural number

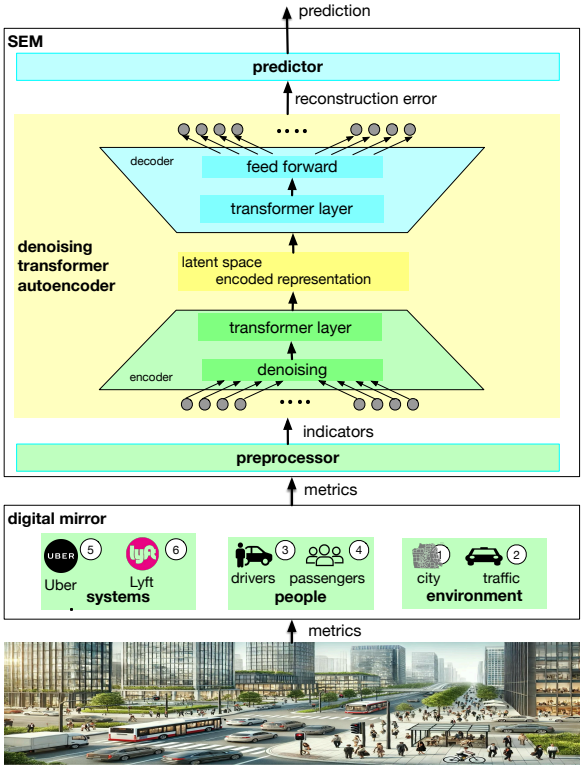


**Figure 1: *SEM* overview**

since the density of failed requests dramatically increases, thus indicating the difficulty of the *SES* to go back to a normal behavior.

We experience a failure of the *SES* (an *unacceptable degradation of SES health* [45]), when all of the three dimensions, hardiness, consistency and resilience, remarkably deteriorate, that is, the strength (hardiness), the stability (consistency) and the ability of the *SES* to go back to an acceptable behavior (resilience) all together vanish. We detect a failure when (i) the value of the index $failed\_requests$ (hardiness) exceeds the $99^{th}$ percentile of the distribution observed during the training period, (ii) the standard deviation of the index $failed\_requests$ (stability) exceeds the $99^{th}$ percentile of the distribution observed during the training period, and (iii) the difference between the short-term and long-term trends of the index $failed\_requests$ (resilience) monotonically grows.

We experience a temporary anomaly of the *SES* when either one or two, but not all three dimensions among hardiness, consistency and resilience deteriorate, that is, any subset of the healthiness of the *SES* temporary degrades, due to an exceptional but not disruptive event. The persistency of at least an aspect of the healthiness guarantees that the *SES* will return to a normal behavior. For instance, a degradation of hardiness and consistency, with a good value of resilience, in a ride-sharing *SES* may be due to an anomalous peak of requests and a consequent temporary delay in the services, before the return to a normal behavior.

## 3 *SEM* Overview

In this paper, we propose *SEM Smart human-centric Ecosystem Monitoring* to detect anomalies of the *SES*, predict *SES* failures, and enable short- and long-term recovery actions. For example, *SEM* may predict a major failure of a peer-to-peer ride-sharing *SES* due to inappropriate greedy drivers' behavior and trigger short- and long-term preventing actions. In the short term the city authority can temporarily suspend the service of the ride managers to prevent an escalation of fares, or single ride managers can force a temporary flat rate for the rides. In the long term, the developers of the peer-to-peer ride-sharing services may define new policies to compute the fares (the *surge multiplier* described later in this section) to prevent the spread of greedy behaviors.

Figure 1 shows the core components of *SEM* and its interface with the *SES*. *SEM* predicts failures from metrics that it collects from the systems, the environment elements, and the human behavior in the *SES*. The lack of central ownership may not allow to monitor metrics of all systems in the *SES*. The digital mirror between *SEM* and the *SES* completes the metrics collected from the *SES*, and allows to experiment with scenarios hardly reproducible in the *SES* itself, due to their disruptive effects.

A digital mirror includes (i) simulators of the environment to experiment with scenarios hardly reproducible in the *SES*, for instance a major alarm in the underground that causes a peak of ride requests, (ii) models of the human behavior, to experiment with not-yet-observed human behavior, for instance a sudden explosion of 'greedy drivers', and (iii) replicas of the software systems in the *SES*, to gather metrics that are not directly available from the *SES*, for example the fares of *Lyft*, a ride-sharing system that does not share the internals. A digital mirror differs from digital twins [35, 44], since it may omit data irrelevant for *SES* monitoring and testing, and includes models of human behavior relevant for testing.

We define *SEM* in Section 4 and the digital mirror we implemented for the experiments in Section 5.

## 4 *SEM*

Recent work demonstrates the effectiveness of deep learning, and in particular autoencoders to detect anomalies and predict failures [48, 54, 56, 68], thus we build *SEM* with an autoencoder core. An autoencoder is an artificial neural network composed of an encoder and a decoder. The encoder compresses the input data into a *latent compressed representation* with a lower dimensionality. The decoder reconstructs the original input data from the compressed representation of the encoder. An autoencoder is trained on a set of input data, to minimize the difference between the original input and the reconstructed output, also known as *reconstruction error*. The autoencoder learns to build a compact representation of the input data (the *latent space*) that captures the underlying structure, retains meaningful features, and ignores noise and irrelevant information. The reconstruction error indicates the degree of correspondence of the input with respect to the training set: High reconstruction errors spot largely anomalous inputs.

We experimented with LSTM and Transformer autoencoder, and chose a Transformer autoencoder as the core technology of *SEM*. A Transformer autoencoder pairs two transformer layers, and operates in parallel with self-attention and positional encoding mechanisms, to capture long-range dependencies between elements in a sequence. Figure 1 shows the main component of *SEM*: the *Preprocessor*, the *Denoising Transformer Autoencoder* and the *Predictor*.

**Table 1: Sample metrics from the ride-sharing *SES***

| *Metric* | *Description* |
|---|---|
| Rejected Rides | number of rides that drivers reject |
| Active passengers | number of active passengers |
| Moving drivers | number of drivers moving to a different TAZ |
| Idle drivers | number of active drivers waiting for a ride request |
| Avg surge multiplier | average value of the surge multiplier for each request |
| Avg expected price | average expected price for each ride |
| Avg real price | average final real price for each ride |
| Avg speed | average speed, computed for each ride |
| Avg ride time | average time to complete a ride after pickup |
| Avg meeting time | average time of the driver to meet the passenger |
| Avg ride length | average length of each ride after pickup |
| Avg meeting length | average length of each meeting route |

The *Pre-processor* converts the input metrics time series into the indicators time series for the autoencoder: (i) It filters out useless metrics; (ii) It computes the rolling average of a time interval (300 timestamps, that is, 5 minutes, in our experiments), to minimize the outliers and reduce the noise within the multivariate time series; (iii) It differs consecutive pairs of metric values, for the Transformer autoencoder that works on differences; (iv) It normalizes the indicators between 0 and 1; (v) It builds floating windows corresponding to 20 timestamps (20 seconds in our experiments).

The metrics depend on the domain. Table 1 reports a subset of the metrics that we collect from the ride-sharing *SES* with a granularity of a second. The interested readers can find the complete set of metrics in the replication package. The ride-sharing metrics refer to several aspects of the rides, including the status, duration, distance, and price, and offer a snapshot of the status of the *SES*. The table reports the metrics with essential explanations that clarify the metrics. The *TAZ* are the Traffic Analysis Zones that partition the city. The *Avg surge multiplier* is a factor between 1 and 5 that *Uber* uses to adjust the base ride fares that are computed from the distance and duration of the journey [15]. The value of the *surge multiplier* increases in response to a growing rate of unfulfilled requests (that is, requests that drivers do not accept), and decreases in the presence of a high concentration of available drivers with respect to passengers requesting a ride in the TAZ.

The *Denoising Transformer Autoencoder* computes the reconstruction error of the time series of indicators from the *Pre-processor*. Figure 1 shows the two layers of the *Denoising Transformer Autoencoder* that we trained with metrics collected over several hours of common operation of the *SES*. A *Denoising Transformer Autoencoder* adds Gaussian noise to the input data for training the autoencoder to learn how to ignore irrelevant variations and focus on the underlying structure of data [64].

The *Predictor* signals a failure if the reconstruction error remains over the $99^{\text{th}}$ percentile of the distribution computed during training for at least 78 seconds, that is, the maximum amount of consecutive seconds from our training data where the reconstruction error is above the $99^{\text{th}}$ percentile. We choose the $99^{\text{th}}$ percentile threshold as it is the most common choice in the literature [9, 26].

## 5 Ride-Sharing *SES* Digital Mirror

Figure 1 outlines the main components of the *RS-Digital-Mirror*, the digital mirror of a peer-to-peer ride-sharing *SES* that we developed to validate our approach.

The *RS-Digital-Mirror* simulates the aspects of the smart city that are relevant for the ride-sharing *SES* (the city plan (1) in Figure 1 and the traffic (2)), models the human behavior that impacts the ride-sharing systems (drivers (3) and passengers (4)), and replicates the ride-sharing systems in the *SES* (*Ride-sharing*). The *Ride-sharing* subsystem is a parametric module that can be instantiated into different service providers, by specifying *base fare*, *service fee*, *cost per mile*, *cost per minute*, *maximum ride length*, *maximum driver search area*, and *maximum driver shift time*. In our experiments we instantiated *Uber* ((5)) and *Lyft* ((6)), the two competing ride-sharing systems widely available in San Francisco.

We fed the digital mirror with publicly available data for the city of San Francisco, *Uber* and *Lyft*, to simulate the behavior of *SEM* with an actual *SES*.

**City** (1) The *RS-Digital-Mirror* simulates a city with the *Simulation of Urban MObility simulator, SUMO* [41], and its APIs: *TraCI* [67] and *sumolib*. We implemented a *Map Parser* that imports both the road layout and the characteristics (ways and points of interest) of a city from *OpenStreetMap OSM* [32] into SUMO. We partitioned the city into *Traffic Analysis Zones* (TAZs), the smallest units on the map for analyzing city traffic, to support the dynamic distribution of passengers and drivers across different districts. We instantiated the city with San Francisco, since both historic and current detailed data of the traffic of San Francisco are publicly available.

**Traffic** (2) The *RS-Digital-Mirror* simulates the *Traffic* by generating cars according to the average hourly traffic of the city, and by assigning a route to each generated car. The cars exit the system at the end of their route. For our experiments we used the average hourly data observed at all intersections within the city of San Francisco over 5 workdays.[4]

**Drivers** (3) The *RS-Digital-Mirror* models the *Drivers* by generating new drivers in the city at each timestamp. The *Drivers* module distributes drivers according to both the average hourly data of the Transportation Authority of San Francisco[5] and *Uber Movement*.[6]

Each driver is affiliated to a ride-sharing service (either *Uber* or *Lyft* in our experiments), and moves on a route within the city (San Francisco). The *RS-Digital-Mirror* dynamically adjusts the initial route assigned to the driver, to model the reactions of the driver to the distribution of ride requests. Once the driver accepts a ride request, the *RS-Digital-Mirror* modifies the route to reach the pickup location. When the driver reaches the pickup location, the *RS-Digital-Mirror* updates the ride to the destination of the passenger. The *RS-Digital-Mirror* monitors time, speed and distance traveled, and computes the offered fare at the time of the request and the final fare at the end of the ride, by including any surge pricing. The drivers either accept or decline requests depending on both the offered fare and the personality of the driver. The *RS-Digital-Mirror* uses a simple model of the drivers' personality that we defined according to Kooti *et al.* 's and Cook *et al.* 's studies [24, 37]. The model abstracts the personality of the drivers with the probability of accepting a ride request depending on the revenue for the drivers. In our *RS-Digital-Mirror*, we define three different personalities for drivers: (i) *greedy driver:* high probability of drivers accepting requests when the *surge multiplier* is high, (ii) *hurry driver:* high

[4]Data publicly available at data.sfgov.org/Transportation, at the time of writing
[5]https://www.sfcta.org/projects/tncs-today-2017
[6]https://www.uber.com/newsroom/introducing-uber-movement. *Uber Movement* was discontinued in October 2023, in our experiments we use the data that we gathered prior up to the closure of the website, and that are available in our replication package

probability of drivers accepting rides even when the *surge multiplier* is low, (iii) *normal driver:* moderate probability of accepting rides, in between greedy and hurry. For instance, hurry drivers accept unprofitable rides (*surge multiplier* between 1 and 1.2) with 90% probability, since hurry drivers tend to accept most of the rides despite the fares, while greedy drivers accept unprofitable rides with 30% probability, since greedy drivers tend to maximize the profit. In our experiments we use a standard distribution of 21% 'hurry' drivers, 24% 'greedy' drivers and 55% 'normal' drivers, by referring to the data available from the Transportation Authority of Chicago[7] and the studies of Ashkrof *et al.* [5] and Cook *et al.* [24].

**Passengers** (4) The *RS-Digital-Mirror* models *Passengers* by generating new passengers in the city at each timestamp. The *Passengers* module distributes new passengers requesting rides according to the data of the Transportation Authority of San Francisco and *Uber Movement*, as in the case of drivers.

Each passenger enters the *SES* with a request for a ride to a ride-sharing provider (either *Uber* or *Lyft*, in our experiments). Passengers' requests are distributed among *Uber* and *Lyft* according to the current market share[8], 75% for *Uber* as first choice, and 25% for *Lyft*. After the initial request, passengers wait for an offer, and either accept the offer (and complete the ride) or decline the offer and either forward the same request to the other ride-sharing provider or exit the *SES*. The passengers either accept or decline offers depending on personality and fare (*surge multiplier*). Similarly to the drivers, the *RS-Digital-Mirror* abstracts passengers' personality with the probability of accepting/declining an offer, and considers three types of personalities: (i) *greedy passenger:* high probability of passengers accepting rides when the *surge multiplier* is low, (ii) *hurry passenger:* high probability of passengers accepting rides even when the *surge multiplier* is unfavorable (high), (iii) *normal passenger:* moderate probability of passengers accepting rides, in between the greedy and hurry ones. In our experiments we use a standard distribution of 37% 'hurry' drivers, 18% 'greedy' drivers and 45% 'normal' passengers, by referring to the data of Uber's users from statista.com, and the studies of Ashkrof *et al.* [5] Cook *et al.* [24], and Kooti *et al.* [37].

**Uber** (5) The *RS-Digital-Mirror* replicates *Uber*, by instantiating a *Ride-sharing* module parametric with respect to: *base fare*, *service fee*, *cost per mile*, *cost per minute*, *maximum ride time*, *maximum drivers available*, and *maximum driver shift time*, with both publicly available data[9][10] and the study of Chen *et al.* [15]. *Uber* publicly shares[11] the factors that determine the pricing algorithm, which include the base fare, the service time, the miles driven, and the *surge multiplier*. The *surge multiplier* is a factor that adjusts the final price based on several dynamic factors, such as the time of day, weather conditions, the ratio between drivers and passengers, and traffic conditions. We implemented the pricing algorithm by referring to the information that is publicly available at sport.publictravels.com.

**Lyft** (6) The internals of *Lyft* are not publicly available. The *RS-Digital-Mirror* substitutes *Lyft* with a Random Forest [11] that we trained on a publicly available dataset[12] which includes data about prices, distances and *surge multiplier* of the city of Boston, an urban area with characteristics comparable with San Francisco.

The *RS-Digital-Mirror* dynamically generates passengers, *Uber* and *Lyft* drivers, and traffic cars at every second within each hour, according to the publicly available data that are hourly sampled.

Table 2 lists the relevant parameters to configure the *RS-Digital-Mirror* with the default values that characterize the *normal* scenario, that is, the routine in San Francisco, as available from previous studies [15]. The full list with a detailed explanation of the parameters is available in the replication package.

**Table 2: Main configurable parameters of *RS-Digital-Mirror***

| *Parameters* | *Description* | *Common values* |
|---|---|---|
| TAZ_involved | Zones of SF involved | All (25 zones in total) |
| driver_generation_policy | Number of hourly drivers added to each TAZ | Historical data from SF authorities |
| passenger_generation_policy | Number of hourly requests generated per TAZ | Historical data from SF authorities |
| traffic_generation_policy | Number of cars added to each TAZ per hour | Historical data from SF authorities |
| traffic_speed_policy | Cars' speed in each road | Speed limits of roads |
| driver_personality_policy | Prob. of driver's personality | Hurry 21%, Greedy 24%, Normal 55% |
| passenger_personality_policy | Prob. of passenger's personality | Hurry 37%, Greedy 18%, Normal 45% |
| market_share | Uber, Lyft market | Uber 75%, Lyft 25% |
| surge_multiplier_policy | Surge multiplier algorithms | Uber and Lyft |
| time_update_surge_multiplier | Update interval for surge multiplier | 5 minutes |
| route_length_distribution | Prob. a passenger requests a ride of a given length | Short 36%, Normal 22%, Long 18%, Extreme 24% |
| work_shift_policy | Prob. of end of shift of each driver at each timestamp | Exponentially growing, up to 3% after 4 hours |

## 6 Experimental Validation

In this section we discuss the results of our experimental evaluation of *SEM*. Section 6.1 presents the research questions that we address with our experiments. Section 6.2 introduces the scenarios that we use to validate *SEM*. Section 6.3 describes the experimental setting. Section 6.4 explores the generalizability of the approach through continual learning. Section 6.5 discusses the experimental results. Section 6.6 acknowledges the threats to the validity of the results.

### 6.1 Research Questions

Our evaluation addresses three main research questions (RQs):

RQ1 *Does* SEM *correctly reflect the healthiness of* SES*?* Question *RQ1* investigates if the reconstruction error that the autoencoder computes on the series of indicators offers a consistent viewpoint of the *SES* healthiness, in terms of the value and stability of the computed reconstruction error. A consistently low reconstruction error in normal execution conditions indicates a correct representation of the *SES* health.

RQ2 *Does* SEM *successfully predict* SES *failures?* Question *RQ2* quantifies the usefulness of *SEM* predictions in terms of both the ability to predict a failure before its occurrence and the interval between the *SEM* prediction and the actual failure.

RQ3 *Can* SEM *effectively generalize to significant* SES *variations?* Question *RQ3* assess the generalizability of *SEM*, by studying the robustness of *SEM* with respect to variations of the *SES*.

[7] https://data.cityofchicago.org/Transportation-Network-Providers-Drivers. We refer to the data of Chicago since we have no access to the data of San Francisco, and the sizes of the urban areas of San Francisco and Chicago are comparable

[8] https://secondmeasure.com/datapoints/rideshare-industry-overview/

[9] https://www.businessinsider.com/guides/tech/how-far-can-uber-take-you

[10] https://www.uber.com/en-GB/newsroom/introducing-new-driver-hours-policy/

[11] https://help.uber.com/riders/article/how-are-fares-calculated

[12] https://www.kaggle.com/datasets/ravi72munde/uber-lyft-cab-prices

## 6.2 Scenarios

We evaluate *SEM* in seven critical scenarios reported in the news and literature: *Underground alarm*[13], *Flash mob*[14], *Wildcat strike*[15], *Long rides*[16], *Greedy drivers*[17], *Boycott Uber*[18], and *Budget passengers*[19]. We summarize the scenarios by indicating the parameters that change with respect to the common values reported in Table 2.

**Underground alarm** *(TAZ_involved = City center (6 TAZ), passenger_generation_policy +100%, traffic_generation_policy +50%)* A sudden alarm in an underground station causes the evacuation of the station with an abrupt and chaotic explosion of ride requests in the area. The amount of new requests depends on the crowd in the station and the duration of the closure of the station. We implement the immediate surge in passenger volume, as a 100% increase in ride requests and a 50% increment of traffic in six contiguous high-traffic areas in the city center, for 40 minutes after the alarm.

**Flash mob** *(TAZ_involved = Central area (11 TAZ), traffic_speed_policy -90% in the City center and -80% in the remaining TAZs)* An unexpected flash mob blocks the city center with effects on the neighbor TAZs. The traffic worsen, and pickup and drop-off intervals increase. We implement a flash mob in the city center with a 90% decrease of the average speed in the streets of the city center (directly affected by the flash mob), and an 80% decrease in the streets of the neighbor TAZs (indirectly affected by the flash mob).

**Wildcat strike** *(driver_generation_policy -50%, work_shift_policy -50% in 30 minutes)* A sudden wildcat strike of drivers dramatically reduces the availability of drivers. We implement a wildcat strike with a decrease of 50% drivers who enter the *SES* after the beginning of the strike, and an increase of 50% drivers who terminate the work shift for 30 minutes after the beginning of the strike.

**Long rides** *(route_length_distribution = ⟨1%, 4%, 10%, 85%⟩)* In 2010, the eruption of the Eyjafjallajökull volcano paralyzed European air traffic [57], quickly saturating the train system and the car rentals. After few days, people with non-negotiable commitments considered taxi and ride-sharing for unusually long rides. We implement the unusual length of the ride requests by changing the distribution of the length of the rides requests from ⟨36%, 22%, 18%, 24%⟩ in normal conditions to ⟨1%, 4%, 10%, 85%⟩ in *long rides* conditions for $\langle Short, Normal, Long, Extreme \rangle$ lengths of ride requests.

**Greedy drivers** *(driver_personality_policy = ⟨5%, 15%, 80%⟩)* A sudden increase of 'greedy drivers', who decline ride requests also with standard fares, when perceiving an increasing number of ride requests, by following the experience that both *Uber* and *Lyft* increase fares when the number of pending requests increases. We implement the unusual increment of greediness by (i) changing the distribution of new drivers who enter the *SES* from ⟨21%, 55%, 24%⟩ in normal conditions to ⟨5%, 15%, 80%⟩ in *greedy drivers* conditions for $\langle hurry, normal, greedy \rangle$ drivers, and (ii) dynamically recomputing the drivers' ride acceptance rate by subtracting the *Greediness* that we compute as:

$$\text{Greediness} = \left( \frac{ratio_{300} - ratio_{1800}}{\text{surge\_multiplier}} \right) \quad (3)$$

at every minute, where $ratio_{300}$ and $ratio_{1800}$ are the average ratios of pending passengers to available drivers over the past 5 and 30 minutes (300 and 1,800 seconds), respectively. The difference between the growth of the drivers-passengers ratio in the short term ($ratio_{300}$, last five minutes) with respect to the longer period ($ratio_{1800}$, 30 minutes) captures the intuition that a high increase of average ratio in the last few minutes indicates some exceptional conditions that may incentivize the spread of greediness. A high difference in the two averages, adjusted with the *surge multiplier* (at the denominator) reduces the likelihood of acceptance of the drivers. The adjustment with the *surge multiplier* reduces the effect when the prices are already high.

**Boycott Uber** *(passenger_generation_policy -40%)* In late January 2017, *GrabYourWallet* invited passengers to boycott *Uber* against the decision of *Uber* not to join the protests against the US Government Executive Order 13769. As a result, approximately 200,000 users deleted the *Uber* mobile app. We implement the critical protest with a decrease of 40% passengers (that is, requests) who enter the *SES* after the beginning of the boycott.

**Budget passengers** *(passenger_personality_policy = ⟨5%, 15%, 80%⟩)* Passengers with limited budget might discard ride offers, when fares are particularly high. A peak of *budget passengers*, like at the end of a student event, may dramatically affect the dynamics of fares and rides. We implement a peak of *budget passengers* by shifting from a distribution of new passengers who enter the *SES* with a prevalence of *normal passengers* (⟨21%, 55%, 24%⟩) to a peak of *greedy passengers* (⟨5%, 15%, 80%⟩).

## 6.3 Experimental Setting

The effectiveness of an autoencoder largely depends on the amount of training data. It is best practice to train an autoencoder with a large amount of data collected in normal conditions, that is, without any evident anomalous or erroneous behavior [7]. We trained the autoencoder with data from 24 consecutive hours of execution of the *RS-Digital-Mirror* in normal conditions. We collected all data by executing the *RS-Digital-Mirror* on a Debian machine hosted on Google Cloud, with 128Gb of RAM.

We executed the *RS-Digital-Mirror* for 27 consecutive hours of traffic, with the setting of the parameters indicated in Table 2 that we derived from data collected from production, as discussed in Section 5. We collected 37 raw metrics every second for a total of 3,596,400 values for a time series of 97,200 timestamps. We pruned the time series of metrics by removing both the first two hours and the last hour of observations to obtain a clean observation of 24 consecutive hours without startup and shutdown perturbations.

We executed the seven scenarios presented above, as well as 11 additional scenarios that we obtained by combining them, for a total of 18 scenarios, as discussed later in Section 6.5. We executed each of the 18 scenarios for five hours. We started each execution at 9:00am of a working day that we obtained as the average workload of five consecutive working days from the publicly available data of the Transportation Authority of San Francisco. We injected the event at noon, after three hours of execution in normal conditions,

[13] https://www.cbsnews.com/news/new-york-city-train-collision-cause-ntsb-report/
[14] https://www.nytimes.com/2023/11/16/world/san-francisco-bay-bridge-protest
[15] https://www.cbsnews.com/news/uber-strike-new-york-city-pay-raises
[16] https://bestreferraldriver.com/long-distance-uber
[17] https://dl.acm.org/doi/pdf/10.1145/3468264.3473137
[18] https://en.wikipedia.org/wiki/Controversies_surrounding_Uber
[19] https://www.thefamuanonline.com/uber-lyft-increasing-prices-affect-students

and terminated the execution after 2 hours with no failure, or at the *SES* failure occurrence, otherwise. We pruned the time series of metrics by removing both the first two hours and the last 30 minutes of observations to obtain a 'clean' observation without startup and shutdown perturbations. This setting gives us 2.5 hours of 'clean' execution to study the effects of each scenario.

As discussed in Section 5, the *RS-Digital-Mirror* runs on SUMO, which executes on a single core with obvious performance limitations. The execution of an hour of real time on the Debian machine requires over 17 hours of CPU time on average. The collection of data for both training and experiments discussed below required over 1,200 CPU hours (50 CPU days).

We identified the combination of hyperparameters for the *Denoising Transformer Autoencoder* by training it on noisy data and validating it on the denoised version of the same training set. Training on noisy inputs while validating on the denoised training set ensures that the model focuses on learning robust representations of normal behavior, improving anomaly detection performance [64]. We used a grid search strategy over 192 combinations, with the Mean Squared Error (MSE) as the loss function for the evaluation.

Finally, we trained the model with the entire dataset and the optimal hyperparameter configuration found with the hyperparameters tuning. The final model is a *Denoising Transformer Autoencoder* with 1 Transformer layer for both the encoder and the decoder and a final Fully-Connected (FC) layer. The input is the multivariate time series of the indicators. We generated non-overlapping windowings of 20 timestamps (that is, 20 seconds), thus obtaining an input shape of (32, 20, 37) with a batch size of 32. We trained the model on the entire dataset for 500 epochs with a learning rate of 0.0001 and *Adam* optimizer [52], and referred to the Mean Squared Error (MSE) as loss function. We computed the reconstruction error on the training set with the average normalized Mean Absolute Error (MAE) across all the 37 indicators.

We trained the autoencoder on a MacBook Pro M3 Max with 36Gb of RAM. We completed the model tuning with grid search in three days and the training of the autoencoder in half an hour.

**Table 3: Grid search hyperparameter tuning**

| Hyperparameters | Value range* |
|---|---|
| *Window size* | [10, **20**] |
| *N layers encoder* | [**1**, 2] |
| *N layers decoder* | [**1**, 2] |
| *Number of attention heads* | [4, **8**] |
| *Embedding model dimension* | [64, **128**] |
| *Feed-forward layer dimension* | [128, **256**, 512] |
| *Learning rate* | [**0.0001**, 0.001] |
| *Dropout rate* | [0, **0.2**] |
| *Batch size* | [**32**, 64] |

*chosen configuration in **bold**

## 6.4 Continual Learning

*SES* evolve over time with changes in the behavior of the systems that comprise the *SES*. We evaluated the generalizability of *SEM* to emerging and evolving scenarios, by dynamically introducing *Flat*, a new ride provider, in the ride-sharing *SES*. *Flat* offers rides at a *flat* rate that depends only on the ride distance and not the duration of the ride. We implemented *Flat* by referring to the San Francisco cab fares[20], by excluding duration, and with no *surge multiplier*. We experimented with a shift of passengers from *Uber: 75%, Lyft: 25%* to *Uber: 65%, Lyft: 20%, Flat: 15%*, with passengers who may decide to change provider depending on the actual fare. The fare of *Flat* is slightly higher than the base fares of *Uber* and *Lyft*, and becomes competitive when the conditions lead the *surge multiplier* to increase the fares. We experimented with the new provider in the normal scenario and in the eight scenarios in which passengers and drivers are most likely to change behavior due to their personality, thus with highest impact on the dynamics of the *SES* (*Greedy drivers*, *Budget passengers*, and their combinations).

We executed each scenario with the same conditions of the previous experiments, from 9:00am to 2:00pm, for a total of 5 hours, with the event injected at noon for two consecutive hours or up to a *SES* failure. We pruned the time series of metrics by removing both the first two hours and the last 30 minutes of observations.

We produced a new baseline by enriching the 24 hours of data from the *SES* with two providers with additional three hours of data from the *SES* with three providers in the normal scenario. The limited additional training allows continual learning to integrate new knowledge incrementally while striving to retain previously acquired information [66], thus reducing the risk of *catastrophic forgetting* [36]. Continual learning allows *SEM* to adapt to evolving conditions without the need for a whole retraining. We implemented continual learning with *Memory-Aware Synapses* (MAS) [1] to ensure that significant parameters remain stable while allowing less critical parameters to adjust to new data. We implemented MAS in the *Denoising Transformer Autoencoder* model by adding a MAS loss term to the original loss function:

$$\mathcal{L}_{\text{MAS}} = \mathcal{L}_{\text{MSE}} + \lambda \sum_i \Omega_i (\theta_i - \theta_i^*)^2 \tag{4}$$

where:

- $\mathcal{L}_{\text{MSE}}$ is the standard task loss
- $\lambda$ is a regularization hyperparameter that controls the strength of memory retention
- $\Omega_i$ represents the importance measure for parameter $\theta_i$, computed based on the sensitivity of the output function to $\theta_i$
- $\theta_i$ is the current parameter value
- $\theta_i^*$ is the parameter value from the previous task

This loss function ensures that important parameters are constrained to remain close to their previously learned values, while less crucial ones can adapt to new data.

We trained *SEM* with the MAS loss term on the new normal scenario, which includes the new flat-rate provider, for 100 epochs with a learning rate of 0.00001, a batch size of 32 and $\lambda = 1$. We updated the reconstruction error on the training set with the average normalized Mean Absolute Error (MAE) across all the 37 indicators. We updated the *Predictor* to signal a failure alert if the reconstruction error remains over the 99$^{\text{th}}$ percentile of the distribution computed during training for at least 75 seconds, the maximum amount of consecutive seconds from the new training data where the reconstruction error is above the 99$^{\text{th}}$ percentile.

[20] https://www.sfmta.com/getting-around/taxi/taxi-fares

**Table 4: Failures and Predictions in different scenarios**

| Scenario | Failure | Prediction | Time after injection (s) | Time before failure (s) |
|---|---|---|---|---|
| Underground alarm (UA) | NO (H,C) | NO | - | - |
| Flash mob (FM) | NO (H,C) | NO | - | - |
| Wildcat strike (WS) | NO (H,C) | NO | - | - |
| Long rides (LR) | NO | NO | - | - |
| Greedy drivers (GD) | NO (C) | NO | - | - |
| Budget passengers (BP) | NO | NO | - | - |
| Boycott Uber (BU) | NO | NO | - | - |
| GD + UA | YES | YES | 1,528 | 781 |
| GD + FM | YES | YES | 348 | 1,667 |
| GD + WS | YES | YES | 1,401 | 2,371 |
| GD + LR | YES | YES | 1,190 | 2,649 |
| LR + UA | YES | YES | 1,043 | 1,344 |
| LR + FM | YES | YES | 411 | 2,378 |
| LR + WS | YES | YES | 1,337 | 573 |
| BP + GD | NO (H,C) | NO | - | - |
| BP + WS | NO (H,C) | NO | - | - |
| BU + FM | YES | YES | 1,795 | 2,390 |
| BU + WS | YES | YES | 2,199 | 1,641 |

*H* and *C* in column Failure indicate hardiness and consistency over threshold, respectively.

## 6.5 Experimental Results

In this section we present the results of our experiments with the *RS-Digital-Mirror* that answer the research questions.

*RQ1: Does* SEM *correctly reflect the healthiness of* SES*?*
RQ1 investigates the correlation between *SES* healthiness and the reconstruction error that the autoencoder computes on the series of indicators. A consistently low reconstruction error in normal execution conditions indicates a correct model of the *SES* healthiness.

We computed the reconstruction error with data from 24 consecutive hours (86,400 timestamps from 0:00am to 11:59pm of a working day), data that we used for training. The values of the reconstruction error range between 0.043 and 0.334. The reconstruction error computed with data from the execution of all scenarios before the injection of the event is consistently low, within the $6^{th}$ and the $84^{th}$ percentile of the training error, thus confirming the values of the training data. The consistently low values of the reconstruction error confirm a good measure of the *SES* healthiness.

***RQ1 Findings****:* SEM *well represents the healthiness of* SES*: The reconstruction error that the autoencoder computes on the series of indicators collected in normal execution conditions well reflects the* SES *healthiness, in terms of value and stability.*

*RQ2: Does* SEM *successfully predict* SES *failures?*
RQ2 investigates the correlation between *SES* failures and the reconstruction error of the autoencoder. Persistent high values of the reconstruction error indicate critical anomalies, that is, anomalies that lead to *SES* failures without corrective actions.

We computed the reconstruction error with the data from the execution of the seven scenarios presented in Section 6.2: *Underground alarm*, *Flash mob*, *Wildcat strike*, *Long rides*, *Greedy drivers*, *Boycott Uber*, and *Budget passengers*, and pairwise combinations for a total of 18 scenarios.

Table 4 shows (i) the failures (column *Failure*), (ii) the predictions (column *Prediction*), (iii) the interval between the injection and the prediction (column *Time after injection*), and (iv) between the prediction and the failure (column *Time before failure*) observed, for the seven single scenarios (top of the table) and the pairwise combinations (bottom of the table). The *SES* does not fail in any of the seven single scenarios and in two pairwise scenarios (*Budget passengers+Greedy drivers* and *Budget passengers+Wildcat strike*) (*NO* in column *Failure*). It fails in the other pairwise combinations (*YES* in column *Failure*).

In the presence of *Underground alarm*, *Flash mob*, *Wildcat strike* and the two pairwise combinations that do not fail, the *SES* experiences a degradation of both hardiness and consistency (indicated with (H, C) in column *Failure*), while resilience remains below the threshold. Thus, the *SES* recovers after a temporary degradation. In the *Greedy drivers* scenario the *SES* experiences a degradation of consistency ((C) in column *Failure*), while hardiness and resilience remain below the threshold. In the *Long rides*, *Boycott Uber* and *Budget passengers* scenarios hardiness, consistency and resilience remain below the threshold. In the failing scenarios all hardiness, consistency and resilience go out of range.

*SEM* predicts a failure (reconstruction over the $99^{th}$ percentile for more than 78 consecutive timestamps) for all (pairwise) scenarios that fail (*YES* in column *Prediction*), and does not predict a failure for all scenarios that do not fail (*NO* in column *Prediction*). *SEM* predicts a failure no later than 2,199 seconds after the injection of the scenario (the beginning of the anomalous conditions) (column *Time after Injection*) and at least 573 seconds before the actual failure (column *Time before failure*). Thus *SEM* predicts failures only in the presence of anomalies that lead to failure, early enough to allow for immediate preventing actions.

Figure 2 shows the diagrams of hardiness, consistency and resilience as the percentage of failed over total requests (diagrams *Indexes of Healthiness* in the figure) and the mean normalized reconstruction error over time (diagrams *MAE*) for four representative scenarios: *Greedy drivers*, *Underground alarm*, *Greedy drivers+Budget passengers*, and *Greedy drivers+Underground alarm*. Each diagram reports the values computed in the 9,000 timestamps (150 minutes, 11:00am - 1:30pm) of the experiments with each scenario. The vertical blue line indicates the timestamp we injected the event (at noon, timestamp 3,600) in all scenarios. The diagram visualizes that hardiness is out of range when the percentage of failed requests is over the threshold (horizontal yellow line). It visualizes that resilience is out of range with a purple color of the percentage of failed requests, and does not visualize consistency out of range, being consistency a standard deviation.

The *Indexes of Healthiness* diagrams of *Greedy drivers*, *Underground alarm*, and *Greedy drivers+Budget passengers* scenarios indicate that the *SES* does not fail in these scenarios: only consistency is out of range in the *Greedy drivers* scenario (vertical black bar), and both consistency and hardiness (vertical red bar) are out of range for the other two scenarios. The *Indexes of Healthiness* diagram of the *Greedy drivers+Underground alarm* scenario shows that hardiness, consistency and resilience go out of range at timestamps 4,571 (vertical red bar), 4,569 (vertical black bar) and 5,909 (vertical purple bar), respectively.

The *MAE* diagrams of *Greedy drivers*, *Underground alarm*, and *Greedy drivers+Budget passengers* scenarios show that the mean normalized reconstruction error does not exceed the $99^{th}$ percentile threshold for more than 78 timestamps, while it does exceed the threshold at time 5,128 (green vertical bar) for the *Greedy drivers+Underground alarm*, thus correctly predicting the failure. The

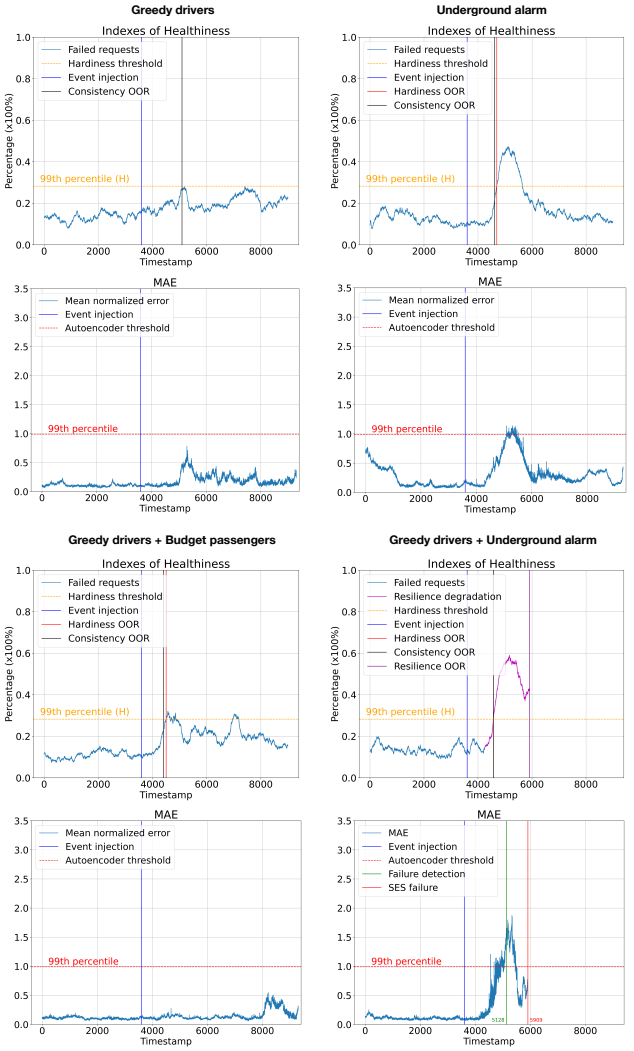


**Figure 2: Sample indexes of healthiness (Failed_requests) and mean normalized reconstruction error (MAE)**

distance between the green (timestamp 5,128) and the red (timestamp 5,909) vertical lines in the *MAE* diagram visualizes the interval (781 seconds, about 13 minutes) for acting to prevent, mitigate and accelerate the recovery from the failure.

The figure shows that the percentage of *failed_requests* grows significantly after the injection in the *Underground alarm* scenario, for about 20 minutes. The reconstruction error rises up to the threshold for less than 78 consecutive timestamps, and *SEM* does not predict any failure. The reconstruction error well reflects the *SES* healthiness also in the presence of anomalous yet not critical events, that is, even-some-time-strong events that perturb the *SES* behavior without leading to failures.

The interval between the failure alert and the actual failure (*Time before failure* in Table 4) spans from 573 timestamps (10 minutes) for the *Wildcat strike+Long rides* scenario, to 2,649 timestamps (44 minutes) for the *Long rides+Greedy drivers* combined scenario, with

**Table 5: Failures and Predictions in different scenarios**

| Scenario | Failure | Prediction | Time after injection (s) | Time before failure (s) |
|---|---|---|---|---|
| GD | YES | YES | 1,575 | 2,068 |
| BP | NO | NO | - | - |
| GD + UA | YES | YES | 1,295 | 1,029 |
| GD + FM | YES | YES | 293 | 2,166 |
| GD + WS | YES | YES | 2,736 | 793 |
| GD + LR | YES | YES | 1,841 | 2,569 |
| BP + GD | YES | YES | 1,982 | 759 |
| BP + WS | NO (R) | NO | - | - |

*R* in column Failure indicates resilience over threshold.

an average prediction time of 1,755 timestamps (29 minutes). A failure alert of half an hour before the actual failure allows for different (semi-) automatic recovery actions, for instance a temporary reward offered to loyal drivers to reduce the effect of greediness, and prevent the disruptive effect of the combination of two scenarios.

*SEM* predicts failures much before their occurrence, and can effectively distinguish between temporary anomalies and failures, while simple observations of the *SES* do not: The easy observations of underground alarms, flash mobs, and wildcat strikes cannot distinguish transient anomalies from incoming failures, due to the absence/presence of drivers' behavior that can be deduced once the failures occurs and cannot be observed directly on the *SES*.

*SEM* identifies anomalies in less than a minute on a MacBook, a time that does not impact on the failure predictions interval, making *SEM* a viable solution for signaling real-time failures in production.

***RQ2 Findings**: SEM can successfully predict SES failures. SEM correctly predicts the failure in all the 9 failing scenarios, with a time interval before the failure that allows for recovery actions.*

*RQ3: Can SEM effectively generalize to significant SES variations?* RQ3 investigates the ability of *SEM* to adapt to significant variations of the *SES*. A consistently low reconstruction error in normal execution conditions and persistent high values of the reconstruction error in failing execution conditions, when the *SES* undergoes a major change, indicates the ability of *SEM* to generalize.

We investigate the generalizability of *SEM*, by experimenting with *SEM* while introducing the new provider *Flat*, as discussed in Section 6.4. Table 5 shows (i) the failures (column *Failure*), (ii) the predictions (column *Prediction*), (iii) the interval between the injection and the prediction (column *Time after injection*), and (iv) between the prediction and the failure (column *Time before failure*), observed for eight scenarios while *Flat* enters the *SES*.

The results in the table confirm the ability of *SEM* to generalize to evolving settings. The additional data collected to incrementally re-train the autoencoder allow *SEM* to correctly predict all failures in the new setting, with no significant variation of the interval between prediction and failure. *SEM* correctly predicts failures that emerge when *Flat* enters the *SES* (rows *Greedy drivers* and *BP+GD*, in Table 5). We argue that the different *SES* behaviors may be due to the impact of flat-rate rides on the behavior of 'greedy' drivers, who may not be able to artificially push up the fares.

***RQ3 Findings**: SEM generalizes to significant variations of the SES. The reconstruction error consistently describes the healthiness of SES and predicts failures in the SES, without complete re-training.*

### 6.6 Threats to Validity

The main threat to the internal validity is the single case study: We executed the experiments with 18 scenarios in total. *SES* involve people and it is impossible to inject failures in real *SES*, like smart cities. Digital twins are complex proprietary systems. We run our experiments on a digital mirror that we implemented with publicly available components (*SUMO, TraCI, sumolib, OSM*), by referring to publicly available data (SFMTA, SFCTA, and *Uber Movement*). We offer the *RS-Digital-Mirror* in a package to replicate the experiments, confirm and refine the preliminary results reported in this paper, and comparatively evaluate new approaches. We carefully implemented the *RS-Digital-Mirror*, and systematically tested the components that we implemented with respect to the data we collected from systems in operation.

We are aware that the results may not be broadly generalizable. Nevertheless, our experiments with continual learning demonstrate the robustness of our solution to significant variations of *SES*.

The main threat to the external validity of our approach is the data availability and governance. Our study assumes that the necessary data can be collected, integrated, and synchronized from multiple sources; however, in real-world scenarios, this may not always be feasible. Issues such as data ownership, access restrictions, and interoperability between different systems could impact the applicability of our approach. In our study, we relied exclusively on open-source data, demonstrating that meaningful insights can still be derived from publicly available information.

## 7 Related Work

To the best of our knowledge, this is the first study on predicting failures in *Smart human-centric EcoSystems*. We overview the relevant papers about *SES* and the core anomaly detection approaches that inspired our work.

**SES Failures** We discuss the main work on *SES* in the introduction [10, 18, 20, 42, 43, 46, 47, 49, 51, 55, 58] where we define *SES*. Here we focus on the few approaches that consider *SES* failures.

Cioroaica *et al.* [19] present a flexible and scalable platform for simulating and visualizing complex interactions among the components of cyber physical systems, to ensure safety, reliability, and performance. The approach is specific to automotive *SES*. Subsequently, Cioroaica *et al.* [21] discuss digital twins [35] for evaluating trust in collaborative systems (*CES*) in real-time scenarios. They address the challenge of ensuring trustworthiness in open ecosystems where systems and services of varying origins collaborate [17, 20, 22].

Sommerville *et al.* [60] well exemplify the relevance and impact of *SES* failures with a clear and inspiring presentation of the *'Flash Crash'*, an exceptional stock market crash, due to subtle interactions among systems that correctly behave as expected.

Batool *et al.* [8] address challenges and potential failures in smart city ecosystems, by focusing on the integration of IoT and Artificial Neural Networks (ANNs) to create an intelligent ecosystem that optimizes smart city functions.

El Moussa *et al.* [45] define *SES* healthiness, intuitively characterize *SES* failures as unacceptable degradation of *SES* health, and argue the usefulness of free energy to predict *SES* failures.

**Anomaly Detection for Failure Prediction** Many recent approaches successfully exploit machine learning to detect anomalies in software [53, 69] and complex systems [27, 39, 59]. Yang *et al.* [69] propose a diffusion-based imputation method to predict failures in large-scale cloud systems like Microsoft 365. Chennappan *et al.* [53] introduce a software failure prediction technique using hybrid machine learning algorithms. The work of Zhao *et al.* [72] and Ran *et al.* [54] indicates excellent results with deep learning to identify complex patterns, with the availability of large datasets for training.

The lack of labeled data has shifted researchers' focus from supervised [33, 65] to semi-supervised and unsupervised methods that do not require labeled data for training. Sun *et al.* [62] exploit a graph autoencoder for data augmentation to address the limitation of scarce labeled data in microservice-based systems. Lee *et al.* [38] propose a semi-supervised cross-modal attention approach to identify system anomalies based on heterogeneous data. Several unsupervised approaches exploit autoencoders [31] to reveal anomalies in different domains, from fraud detection [56] to cloud-based systems [26, 69], sales [48] and air quality [68].

The most promising studies leverage the Transformer architecture [63] to capture long-range dependencies in time series data [70]. Zhang *et al.* [71] propose a Transformer-based variational autoencoder for detecting anomalies on several public real-world multivariate time series datasets. Chen *et al.* [16] train a Transformer autoencoder to learn the graph structure and model the anomaly information flow between network nodes in IoT systems. Fang *et al.* [29] propose a dual Transformer-based autoencoder architecture with adversarial learning, achieving state-of-the-art performance on several benchmark datasets.

## 8 Conclusion

In this paper we define *SES* failures, present a digital mirror of a *SES*, propose an effective approach to predict *SES* failures before their occurrence, and discuss a set of preliminary results that open relevant research directions towards robust *SES*.

We present an effective approach to predict *SES* failures early enough to activate preventing actions. We present the results of a set of experiments on the digital mirror of a ride-sharing *SES* operating in San Francisco. The experiments confirm the possibility of *SES* failures that may occur in the presence of combinations of unavoidable events and human behaviors, regardless of the correct behavior of the software systems, and indicate the ability of our approach to predict failures that cannot be revealed in advance with simple monitoring systems. The results presented in the paper open important research directions towards robust *SES*: The paper proposes a new way to deal with *SES* failures well introduced in the literature, but never addressed so far, and offers a relevant experimental ground to the community, to both study alternative approaches, and move from prediction to prevention with automatic corrective actions.

## 9 Data Availability

We provide *RS-Digital-Mirror*, together with a comprehensive usage guide, all scenario details, and scripts to use *SEM* in the replication package.[21]

[21] Replication package available at https://anonymous.4open.science/r/SES_failure_prediction

# References


[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory Aware Synapses: Learning What (not) to Forget. In *Computer Vision – ECCV 2018*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.). Springer International Publishing, Cham, 144–161.

[2] Andrew Amey, John Attanucci, and Rabi Mishalani. 2011. Real-time ridesharing: opportunities and challenges in using mobile phone technology to improve rideshare services. *Transportation Research Record* 2217, 1 (2011), 103–110.

[3] Mohsen Anvaari, Daniela S Cruzes, and Reidar Conradi. 2012. Smart Grid software applications as an ultra-large-scale system: Challenges for evolution. In *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*. IEEE, 1–6.

[4] Fabio Arena, Giovanni Pau, and Alessandro Severino. 2020. An overview on the current status and future perspectives of smart cars. *Infrastructures* 5, 7 (2020), 53.

[5] Peyman Ashkrof, Gonçalo Homem de Almeida Correia, Oded Cats, and Bart van Arem. 2020. Understanding Ride-sourcing Drivers' Behaviour and Preferences: Insights from Focus Groups Analysis. *Research in Transportation Business & Management* 37 (2020), 100516. https://doi.org/10.1016/j.rtbm.2020.100516

[6] CN Gireesh Babu, KT Chandrashekhara, Jhanvi Verma, and M Thungamani. 2021. Real time alert system to prevent car accident. In *2021 International Conference on Forensics, Analytics, Big Data, Security (FABS)*, Vol. 1. IEEE, 1–4.

[7] Pierre Baldi. 2011. Autoencoders, unsupervised learning and deep architectures. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27* (Washington, USA) *(UTLW'11)*. JMLR.org, 37–50.

[8] Tooba Batool, Sagheer Abbas, Yousef Alhwaiti, Muhammad Saleem, Munir Ahmad, Dr Asif, and Nouh Sabri. 2021. Intelligent Model Of Ecosystem For Smart Cities Using Artificial Neural Networks. *Intelligent Automation and Soft Computing* 30 (08 2021), 513–525. https://doi.org/10.32604/iasc.2021.018770

[9] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. 2019. Anomaly Detection Using Autoencoders in High Performance Computing Systems. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (07 2019), 9428–9433. https://doi.org/10.1609/aaai.v33i01.33019428

[10] Jan Bosch. 2009. From Software Product Lines to Software Ecosystems. In *Proceedings of the 13th International Software Product Line Conference* (San Francisco, California, USA) *(SPLC '09)*. Carnegie Mellon University, 111–119.

[11] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (oct 2001), 5–32. https://doi.org/10.1023/A:1010933404324

[12] Barry Brown and Eric Laurier. 2012. The normal natural troubles of driving with GPS. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1621–1630.

[13] Ryan Calo and Alex Rosenblat. 2017. The taking economy: Uber, information, and power. *Colum. L. Rev.* 117 (2017), 1623.

[14] Mahil Carr. 2007. Mobile payment systems and services: an introduction. In *Mobile Payment Forum*, Vol. 1. 1–12.

[15] Le Chen, Alan Mislove, and Christo Wilson. 2015. Peeking Beneath the Hood of Uber. In *Proceedings of the 2015 Internet Measurement Conference*. Association for Computing Machinery, 495–508. https://doi.org/10.1145/2815675.2815681

[16] Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. 2022. Learning Graph Structures With Transformer for Multivariate Time-Series Anomaly Detection in IoT. *IEEE Internet of Things Journal* 9, 12 (2022), 9179–9189. https://doi.org/10.1109/JIOT.2021.3100509

[17] Emilia Cioroaica, Stanislav Chren, Barbora Buhnova, Thomas Kuhn, and Dimitar Dimitrov. 2019. Towards Creation of a Reference Architecture for Trust-Based Digital Ecosystems. In *Proceedings of the 13th European Conference on Software Architecture - Volume 2*. Association for Computing Machinery, 273–276. https://doi.org/10.1145/3344948.3344973

[18] Emilia Cioroaica, Stanislav Chren, Barbora Buhnova, Thomas Kuhn, and Dimitar Dimitrov. 2020. Reference Architecture for Trust-Based Digital Ecosystems. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*. 266–273. https://doi.org/10.1109/ICSA-C50368.2020.00051

[19] Emilia Cioroaica, Thomas Kuhn, and Thomas Bauer. 2018. Prototyping Automotive Smart Ecosystems. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN, Luxembourg, June 25-28, 2018*. IEEE Computer Society, 255–262. https://doi.org/10.1109/DSN-W.2018.00072

[20] Emilia Cioroaica, Thomas Kuhn, and Barbora Buhnova. 2019. (Do Not) Trust in Ecosystems. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. 9–12. https://doi.org/10.1109/ICSE-NIER.2019.00011

[21] Emilia Cioroaica, Thomas Kuhn, and Dimitar Dimitrov. 2021. *Supporting the Creation of Digital Twins for CESs*. Springer International Publishing, 283–294. https://doi.org/10.1007/978-3-030-62136-0_14

[22] Emilia Cioroaica, Daniel Schneider, Hanna AlZughbi, Jan Reich, Rasmus Adler, and Tobias Braun. 2019. Predictive Runtime Simulation for Building Trust in Cooperative Autonomous Systems. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 86–89. https://doi.org/10.1109/DSN-W.2019.00024

[23] United States. Commodity Futures Trading Commission, United States. Securities, and Exchange Commission. 2010. *Preliminary Findings Regarding the Market Events of May 6, 2010: Report of the Staffs of the CFTC and SEC to the Joint Advisory Committee on Emerging Regulatory Issues*. U.S. Commodity Futures Trading Commission. https://books.google.ch/books?id=aCaSnQAACAAJ

[24] Cody Cook, Rebecca Diamond, Jonathan Hall, John List, and Paul Oyer. 2020. The Gender Earnings Gap in the Gig Economy: Evidence from over a Million Rideshare Drivers. *The Review of Economic Studies* 88 (11 2020). https://doi.org/10.1093/restud/rdaa081

[25] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review* 41, 6 (2007), 205–220.

[26] Giovanni Denaro, Noura El Moussa, Rahim Heydarov, Francesco Lomio, Mauro Pezzè, and Ketai Qiu. 2024. Predicting Failures of Autoscaling Distributed Applications. *Proc. ACM Softw. Eng.* 1, FSE, Article 87 (July 2024), 22 pages. https://doi.org/10.1145/3660794

[27] Maria Drakaki, Yannis Karnavas, Ioannis Tziafettas, Vasilis Linardos, and Panagiotis Tzionas. 2022. Machine Learning and Deep Learning Based Methods Toward Industry 4.0 Predictive Maintenance in Induction Motors: A State of the Art Survey. *Journal of Industrial Engineering and Management* 15 (02 2022), 31. https://doi.org/10.3926/jiem.3597

[28] Schahram Dustdar, Stefan Nastic, and Ognjen Scekic. 2017. *Smart Cities - The Internet of Things, People and Systems*. Springer. https://doi.org/10.1007/978-3-319-60030-7

[29] Yuchen Fang, Jiandong Xie, Yan Zhao, Lu Chen, Yunjun Gao, and Kai Zheng. 2024. Temporal-Frequency Masked Autoencoders for Time Series Anomaly Detection. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 1228–1241. https://doi.org/10.1109/ICDE60146.2024.00099

[30] Claudiu Farcas, Emilia Farcas, and Ingolf Krüger. 2008. Requirements for service composition in ultra-large scale software- intensive systems. In *Proceedings of the 15th Monterey conference on Foundations of Computer Software: future Trends and Techniques for Development*. Springer, 93–115.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[32] Mordechai Haklay and Patrick Weber. 2008. Openstreetmap: User-generated street maps. *IEEE Pervasive computing* 7, 4 (2008), 12–18.

[33] Watson Jia, Raj Mani Shukla, and Shamik Sengupta. 2019. Anomaly Detection using Supervised Learning and Multiple Statistical Methods. (2019), 1291–1297. https://doi.org/10.1109/ICMLA.2019.00211

[34] Shan Jiang, Le Chen, Alan Mislove, and Christo Wilson. 2018. On ridesharing competition and accessibility: Evidence from uber, lyft, and taxi. In *Proceedings of the 2018 World Wide Web Conference*. 863–872.

[35] Franz-Josef Kahlen, Shannon Flumerfelt, and Anabela Alves. 2016. *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*. Springer. 1–327 pages. https://doi.org/10.1007/978-3-319-38756-7

[36] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114, 13 (mar 2017), 3521–3526. https://doi.org/10.1073/pnas.1611835114

[37] Farshad Kooti, Mihajlo Grbovic, Luca Maria Aiello, Nemanja Djuric, Vladan Radosavljevic, and Kristina Lerman. 2017. Analyzing Uber's Ride-Sharing Economy. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 574–582. https://doi.org/10.1145/3041021.3054194

[38] Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, Yongqiang Yang, and Michael R. Lyu. 2023. Heterogeneous Anomaly Detection for Software Systems via Semi-supervised Cross-modal Attention. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 1724–1736. https://doi.org/10.1109/ICSE48619.2023.00148

[39] Yaguo Lei, Bin Yang, Xinwei Jiang, Feng Jia, Naipeng Li, and Asoke K. Nandi. 2020. Applications of Machine Learning to Machine Fault Diagnosis: A Review and Roadmap. *Mechanical Systems and Signal Processing* 138 (2020), 106587. https://doi.org/10.1016/j.ymssp.2019.106587

[40] Michael W Levin, Kara M Kockelman, Stephen D Boyles, and Tianxin Li. 2017. A general framework for modeling shared autonomous vehicles with dynamic network-loading and dynamic ride-sharing application. *Computers, Environment and Urban Systems* 64 (2017), 373–383.

[41] Pablo Álvarez López, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie WieBner. 2018. Microscopic Traffic Simulation using SUMO. In *21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018*, Wei-Bin Zhang, Alexandre M. Bayen, Javier J. Sánchez Medina, and Matthew J. Barth (Eds.). IEEE, 2575–2582. https://doi.org/10.1109/ITSC.2018.8569938

[42] Mark W Maier. 1998. Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering* 1, 4

(1998), 267–284.

[43] Konstantinos Manikas and Klaus Hansen. 2013. Reviewing the Health of Software Ecosystems - A Conceptual Framework Proposal. *CEUR Workshop Proceedings* 987.

[44] Judith Michael, Istvan David, and Dominik Bork. 2024. Digital Twin Evolution for Sustainable Smart Ecosystems. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems* (Linz, Austria) *(MODELS Companion '24)*. Association for Computing Machinery, New York, NY, USA, 1061–1065. https://doi.org/10.1145/3652620.3688343

[45] Noura El Moussa, Davide Molinelli, Mauro Pezzè, and Martin Tappler. 2021. Health of smart ecosystems. In *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (Eds.). ACM, 1491–1494. https://doi.org/10.1145/3468264.3473137

[46] Cornelius Ncube. 2011. On the engineering of systems of systems: Key challenges for the Requirements Engineering community. In *2011 Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems*. 70–73. https://doi.org/10.1109/RESS.2011.6043923

[47] Cornelius Ncube, Soo Ling Lim, and Huseyin Dogan. 2013. Identifying top challenges for international research on requirements engineering for systems of systems engineering. In *2013 21st IEEE International Requirements Engineering Conference (RE)*. 342–344. https://doi.org/10.1109/RE.2013.6636746

[48] H.D. Nguyen, K.P. Tran, S. Thomassey, and M. Hamad. 2021. Forecasting and Anomaly Detection Approaches using LSTM and LSTM Autoencoder Techniques with the Applications in Supply Chain Management. *International Journal of Information Management* 57 (2021), 102282. https://doi.org/10.1016/j.ijinfomgt.2020.102282

[49] L. Northrop, Peter Feiler, R.P. Gabriel, John Goodenough, Rick Linger, Thomas Longstaff, Rick Kazman, M. Klein, Douglas Schmidt, Kevin Sullivan, and Kurt Wallnau. 2006. *Ultra-Large-Scale Systems - The Software Challenge of the Future*. Technical Report. Software Engineering Institute – Carnegie Mellon.

[50] Mehdi Nourinejad and Matthew J Roorda. 2016. Agent based model for dynamic ridesharing. *Transportation Research Part C: Emerging Technologies* 64 (2016), 117–132.

[51] Patricia Oberndorf and Carol Sledge. 2010. Evolution of a software engineer in a SoS system engineering world. In *2010 IEEE International Systems Conference*. 91–96. https://doi.org/10.1109/SYSTEMS.2010.5482478

[52] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). https://api.semanticscholar.org/CorpusID:6628106

[53] Chennappan R. and Vidyaa Thulasiraman. 2023. An automated software failure prediction technique using hybrid Machine learning algorithms. *Journal of Engineering Research* 11 (03 2023), 100002. https://doi.org/10.1016/j.jer.2023.100002

[54] Yongyi Ran, Xin Zhou, Pengfeng Lin, Yonggang Wen, and Ruilong Deng. 2019. A Survey of Predictive Maintenance: Systems, Purposes and Approaches. arXiv:1912.07383 [eess.SP]

[55] Damian Roca, Daniel Nemirovsky, Mario Nemirovsky, Rodolfo Milito, and Mateo Valero. 2016. Emergent Behaviors in the Internet of Things: The Ultimate Ultra-Large-Scale System. *IEEE Micro* 36, 6 (2016), 36–44. https://doi.org/10.1109/MM.2016.102

[56] Najmi Rosley, Gee-Kok Tong, Keng-Hoong Ng, Suraya Nurain Kalid, and Kok-Chin Khor. 2022. Autoencoders with Reconstruction Error and Dimensionality Reduction for Credit Card Fraud Detection. In *Proceedings of the International Conference on Computer, Information Technology and Intelligent Computing (CITIC 2022)*. Atlantis Press, 503–512. https://doi.org/10.2991/978-94-6463-094-7_40

[57] A. R. Schmitt and A. Kuenz. 2015. A reanalysis of aviation effects from volcano eruption of Eyjafjallajökull in 2010. In *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. 1B3–1–1B3–7. https://doi.org/10.1109/DASC.2015.7311335

[58] Klaus-Benedikt Schultis, Christoph Elsner, and Daniel Lohmann. 2013. Moving Towards Industrial Software Ecosystems: Are Our Software Architectures Fit for the Future?. In *Proceedings of the 4th International Workshop on Product LinE Approaches in Software Engineering (PLEASE 2013)*. IEEE, 9–12. https://doi.org/10.1109/PLEASE.2013.6608655

[59] Reza Shokrzad. 2023. 6 Pivotal Anomaly Detection Methods: From Foundations to 2023's Best Practices. *Medium* (2023). https://medium.com/@reza.shokrzad/6-pivotal-anomaly-detection-methods-from-foundations-to-2023s-best-practices-5f037b530ae6 Accessed: 2024-07-25.

[60] Ian Sommerville, Dave Cliff, Radu Calinescu, Justin Keen, Tim Kelly, Marta Kwiatkowska, John Mcdermid, and Richard Paige. 2012. Large-scale Complex IT Systems. *Commun. ACM* 55, 7 (July 2012), 71–77. http://doi.acm.org/10.1145/2209249.2209268

[61] Kevin Sullivan, William Knaus, and Richard Marks. 2010. An Ultra-Large-Scale Systems Approach to National-Scale Health Information Systems. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research* (Santa Fe, New Mexico, USA) *(FoSER '10)*. Association for Computing Machinery, New York, NY, USA, 365–368. https://doi.org/10.1145/1882362.1882436

[62] Yongqian Sun, Zihan Lin, Binpeng Shi, Shenglin Zhang, Shiyu Ma, Pengxiang Jin, Zhenyu Zhong, Lemeng Pan, Yicheng Guo, and Dan Pei. 2025. Interpretable Failure Localization for Microservice Systems Based on Graph Autoencoder. *ACM Trans. Softw. Eng. Methodol.* 34, 2, Article 52 (jan 2025), 28 pages. https://doi.org/10.1145/3695999

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[64] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning* (Helsinki, Finland) *(ICML '08)*. Association for Computing Machinery, New York, NY, USA, 1096–1103. https://doi.org/10.1145/1390156.1390294

[65] Jinjiang Wang, Peilun Fu, Laibin Zhang, Robert X. Gao, and Rui Zhao. 2019. Multilevel Information Fusion for Induction Motor Fault Diagnosis. *IEEE/ASME Transactions on Mechatronics* 24, 5 (2019), 2139–2150. https://doi.org/10.1109/TMECH.2019.2928967

[66] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE transactions on pattern analysis and machine intelligence* PP (02 2024). https://doi.org/10.1109/TPAMI.2024.3367329

[67] Axel Wegener, Michał Piórkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. 2008. TraCI: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium*. 155–163.

[68] Yuanyuan Wei, Julian Jang-Jaccard, Wen Xu, Fariza Sabrina, Seyit Camtepe, and Mikael Boulic. 2023. LSTM-Autoencoder based Anomaly Detection for Indoor Air Quality Time Series Data. *IEEE Sensors Journal* PP (02 2023). https://doi.org/10.1109/JSEN.2022.3230361

[69] Fangkai Yang, Wenjie Yin, Lu Wang, Tianci Li, Pu Zhao, Bo Liu, Paul Wang, Bo Qiao, Yudong Liu, Mårten Björkman, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Diffusion-Based Time Series Data Imputation for Cloud Failure Prediction at Microsoft 365. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (San Francisco, CA, USA) *(ESEC/FSE 2023)*. Association for Computing Machinery, New York, NY, USA, 2050–2055. https://doi.org/10.1145/3611643.3613866

[70] Zahra Zamanzadeh Darban, Geoffrey I. Webb, Shirui Pan, Charu Aggarwal, and Mahsa Salehi. 2024. Deep Learning for Time Series Anomaly Detection: A Survey. *ACM Comput. Surv.* 57, 1, Article 15 (oct 2024), 42 pages. https://doi.org/10.1145/3691338

[71] Hongwei Zhang, Yuanqing Xia, Tijin Yan, and Guiyang Liu. 2021. Unsupervised Anomaly Detection in Multivariate Time Series through Transformer-based Variational Autoencoder. In *2021 33rd Chinese Control and Decision Conference (CCDC)*. 281–286. https://doi.org/10.1109/CCDC52312.2021.9601669

[72] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X. Gao. 2019. Deep Learning and Its Applications to Machine Health Monitoring. *Mechanical Systems and Signal Processing* 115 (2019), 213–237.