Mondragon Unibertsitatea Biblioteka Biblioteca

biblioteka@mondragon.edu

This is an Accepted Manuscript version of the following article, accepted for publication in:

J. Galarraga, A. A. Marcos, S. Ali, G. Sagardui and M. Arratibel, "Genetic Algorithm-based Testing of Industrial Elevators under Passenger Uncertainty," 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2021, pp. 353-358

DOI: https://doi.org/10.1109/ISSREW53611.2021.00101.

© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Genetic Algorithm-based Testing of Industrial Elevators under Passenger Uncertainty

Joritz Galarraga*, Aitor Arrieta Marcos†, Shaukat Ali*,
Goiuria Sagardui† and Maite Arratibel ‡
Simula Research Laboratory*, Mondragon University †, Orona‡
* jorgalaro@gmail.com, shaukat@simula.no, †{aarrieta,gsagardui}@mondragon.edu,
‡marratibel@orona-group.com

***Abstract*—Elevators, as other cyber-physical systems, need to deal with uncertainty during their operation due to several factors such as passengers and hardware. Such uncertainties could affect the quality of service promised by elevators and in the worst case lead to safety hazards. Thus, it is important that elevators are extensively tested by considering uncertainty during their development to ensure their safety in operation. To this end, we present an uncertainty testing methodology supported with a tool to test industrial dispatching systems at the Software-in-the-Loop (SiL) test level. In particular, we focus on uncertainties in passenger data and employ a Genetic Algorithm (GA) with specifically designed genetic operators to significantly reduce the quality of service of elevators, thus aiming to find uncertain situations that are difficult to extract by users. An initial experiment with an industrial dispatcher revealed that the GA significantly decreased the quality of service as compared to not considering uncertainties. The results can be used to further improve the implementation of dispatching algorithms to handle various uncertainties.**

***Index Terms*—elevators, genetic algorithms, quality of service, uncertainty, passenger data, software in the loop simulation.**

I. Introduction

Uncertainty is a major concern in the context of elevators, which is contributed by several aspects such as due to passengers behaviors and hardware. The effect of these uncertainties on elevators can range from simple annoyance to its passengers to safety hazards that can potentially harm its passengers. Thus, we need software testing tools to support testing elevators behaviors under uncertainties.

This paper presents a Genetic Algorithm-based Uncertainty Testing methodology for Elevators (GAUTE) specifically for an industrial dispatcher provided by our industrial partner – Orona– a world-leader company in developing elevators. The dispatcher is integrated inside a commercial simulation tool called Elevate that has the capability to simulate the hardware and the environment of elevators.

In the past, several approaches have proposed testing methods for systems of elevators [1], [2]. As for testing dispatching algorithms, Ayerdi et al., proposed an approach based on metamorphic testing [3] whereas Arrieta et al., used machine-learning algorithms [4]. Both of these works tackle the test oracle problem in the same context as our's, but the uncertainty at which a system of elevators is exposed to is not considered.

Our tool extends the industrial dispatcher with a Genetic Algorithm (GA) to support testing by focusing on uncertainties in passenger data such as how much a passenger weighs and how long s/he will take to get in or out of an elevator. The GA is guided through Quality of Service (QoS) metrics (e.g., waiting times) to decrease the QoS of elevators and subsequently finding unforeseen situations.

The main contributions of this paper can be summarized as follows:

- We present GAUTE, a tool-supported methodology based on a GA to test elevators dispatching algorithms under uncertainty. We propose a domain-specific problem representation for GA and genetic operators (i.e., parents selection, mutation. and crossover operators) that tackle the problem.
- We present a detailed architecture of the tool and its integration with the industrial dispatcher and the Elevate–SiL software testing tool.
- We provide a preliminary evaluation of the approach with an industrial dispatching algorithm provided by Orona.
- We provide the lessons learned from applying the tool, and the next steps towards the transfer of the tool to practitioners.

The structure of the paper is as follows: Industrial context is discussed in Section II. The tool supported methodology, i.e., GAUTE, is presented in Section III, whereas validation in Section IV. We provide lessons learned in Section V and related works in Section VI. Conclusions and future works are presented in Section VII.

II. Industrial Context

Elevators are complex Cyber-Physical Systems (CPSs) that aim at transporting passengers in a building safely and by providing the best QoS as possible. To this end, several computing components interact among them and with the physical environment of the system. A typical installation of elevator systems is shown in Figure 1, which consists of a set of elevators (e.g., three as shown in Figure 1) installed in a building with a set of floors (e.g., N floors in Figure 1). On each floor, there is a control panel that passengers use to

This work is supported by the Adeptness project, Grant agreement ID: 871319, Horizon 2020.

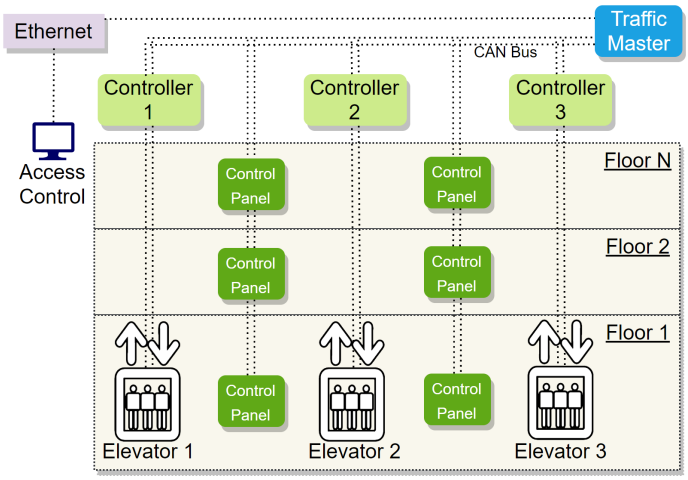

Fig. 1: Elevator Installation Architecture

call for the elevators. Each elevator has a dedicated controller responsible for controlling the movements of its elevator. All the control panels, elevators, and controllers are connected via Controller Area Network (CAN) bus to *Traffic Master*. The traffic master is in charge of controlling the passengers' flow. The core software component of the traffic master of an elevator is its dispatching algorithm, coined as "dispatcher". This algorithm aims at assigning an elevator to each passenger call by following certain criteria (e.g., reducing the passenger average waiting time, reducing energy consumption). The traffic master is also connected via an Ethernet connection to a dedicated computer system responsible for access control, if required, e.g., to control the access to lifts on the first floor.

Orona is one of the largest elevators companies in Europe. It has a large suite of dispatching algorithms to accommodate the needs of different kinds of buildings (e.g., the dispatcher of a system of elevators for a hospital has different needs to one of a hotel). As any other complex software system, their dispatching algorithms requires extensive maintenance. This is to deal with different aspects, such as, hardware obsolescence, new functionality, legislative changes and bug corrections [5]. Orona has a well established process to test and validate their dispatching algorithms. We refer the reader to [5], [3] for a detailed explanation of the dispatching testing process. This process relies on simulation-based testing at different test levels. At the Software-in-the-Loop (SiL) test level, Orona uses Elevate, a domain-specific simulation tool, to test their dispatchers. A test case in the context of Orona for testing their dispatching algorithms includes (1) the building configuration and (2) a passenger file. The former is related to the physical layer of the system of elevators, and encompasses information like the number of floors of the building, number of elevators, dynamic information of the elevators (e.g., maximum speed, acceleration, jerk), etc. The latter refers to a file with information of the passengers arriving at a floor and calling for an elevator in order to travel to another floor. For each of these passengers, different information is provided, including their arrival time, their weights, their up-loading and downloading time. In this paper, we focus on these passenger attributes and consequently, we provide their definitions below for an understanding:

- *Arrival time:* Time at which a passenger arrives at a floor and call for an elevator.
- *Arrival floor:* The floor number at which the passenger arrives.
- *Destination floor:* The floor at which the passenger wants to go.
- *Mass:* The weight of a passenger in kilograms (Kgs).
- *Capacity factor:* A percentage value, based on which, a passenger decides whether to take the arrived elevator.
- *Loading Time and Unloading Time*: The times a passenger takes to get in or get out of an elevator.

The passenger files are typically full-day traffic profiles, which are either (1) real traffic data, which needs to be extracted from the building by installing certain monitors or (2) theoretical traffic data, which are based on extensive elevator dispatching studies [6]. However, in both cases, certain assumptions are performed. For instance, the weight of all the passengers is set to 75 Kgs, which is considered a standard average weight in Europe for simulation.

Testing focuses on finding faults in addition to issues with QoS, which are measured with different attributes. In this paper, we focus on waiting time (WT) associated with each passenger, i.e., the time the dispatcher took it to serve a passenger. Its measurement starts when a passenger presses a button to call an elevator until an elevator arrives on the floor. For a set of n passengers, the Average Waiting Time (AWT) is simply the average of their WTs. This metric is employed because it has been demonstrated that the perception of whether a system of elevators is working properly or not for a passenger is related to the time this passenger needs to wait, rather than other metrics (e.g., time spent by the passenger traveling inside the elevator) [6].

III. Tool Supported Methodology

This section describes our tool supported methodology. Section III-A presents the overall architecture, whereas the methodology is presented in Section III-B.

A. Overall Architecture

GAUTE's overall architecture is shown in Figure 2. It is built on top of Elevate[1]– a commercial simulation software providing the facilities to test elevator software (e.g., dispatcher algorithms) with hardware and environment simulated as discussed in Section II. Within Elevate, one can implement and deploy their own dispatcher algorithm, which is responsible for optimal scheduling of passengers arriving at different floors. Orona provided the dispatcher implementation in C++ and is deployed as .dll inside Elevate.

We implemented GAUTE to support testing Orona's dispatcher under various uncertainties, in particular related to passengers. During the simulation, GAUTE introduces passengers

[1]https://peters-research.com/index.php/elevate/

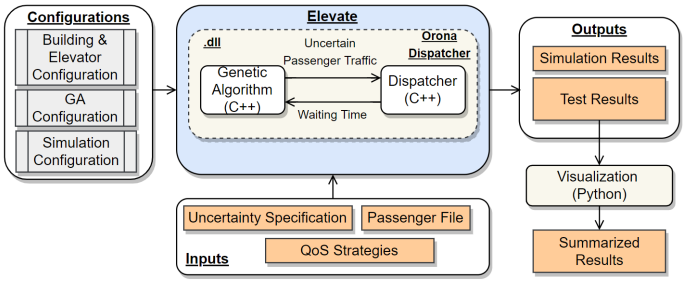

Fig. 2: Tool Supported Methodology

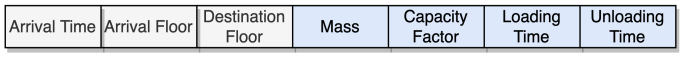

Fig. 3: Individual Representation

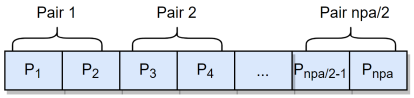

Fig. 4: Pairs

with varying passenger attributes as described in Section II, i.e., mass, capacity factor, loading time, and unloading time. This is in contrast to a typical practice, where a fixed value for each of these attributes is used based on existing standards, and guides (e.g., [7]). The rationale is that in the real operation, it is uncertain, e.g., at which time a passenger will arrive and on which floor, and how much he/she will weigh. GAUTE focuses on providing a support to study uncertainties in mass, capacity factor, loading time, and unloading time of each passenger. The uncertainty specifications of each of these attributes are provided by a test engineer in the form of realistic ranges. For example, in Europe, the recommended average mass is 75KGs, thus a test engineer can specify an interval for mass, e.g., 70KGs to 80KGs.

We implemented a GA inside the dispatcher that controls the generation of the next *np* of passengers based on the current *AWT* (details in the next section). We chose the GA since it is one of the most commonly used search algorithms in the software engineering domain. Nonetheless, other search algorithms can be employed in the future.

As shown in Figure 2, Elevate together with the dispatcher including GA tests the dispatcher under uncertain attributes of passengers. It takes as inputs the following configurations: (1) a building and elevator configuration (e.g., number of elevators, their sizes and capacities, number of floors in a building); (2) configuration of a GA such as population size; (3) simulation configurations (e.g., time slide between simulation calculations). Also, it takes input uncertainty specifications, a passenger file, and QoS strategies (described in the next section). A simulation is then started, which keeps inserting the passengers following the passenger file, but varying the attributes of each passenger as decided by GA based on current AWT value. Once the simulation is completed, a set of simulation reports are produced by Elevate, in addition to a *Test Results* file, which is produced by GAUTE. This file contains all the detailed information such as AWT in each generation, what were the chosen values of attributes of passengers by GA and so on.

We also implemented a *Visualization* component in Python that takes input test results and produces visualization of results (e.g., graphs) to analyze how the AWT was increased with GA in each generation.

B. Methodology

We aim to test a dispatcher with uncertain attributes of a passenger with the ultimate aim of increasing QoS attributes (e.g., AWT) with a GA, consequently leading to bad QoS. To this end, we need to encode our problem in the GA.

1) Individual Representation: First, we need to provide representation of each individual, which in our case is one passenger. A representation of the individual is shown in Figure 3 having seven attributes. The first attributes are used as it is from the original traffic profile for the first generation, and for the following generations as close as possible to the original traffic profile to ensure that the traffic remains as realistic as possible. The rest of the four attributes, i.e., mass (*m*), capacity factor (*cf*), loading time (*lt*), and unloading time (*ut*), are generated by a GA according to the uncertainty specification provided by a test engineer. To be more specific, for a passenger *i*, these attributes will be referred as: m_i, cf_i, lt_i, and ut_i. Each individual after simulation will have an associated waiting time (WT), i.e., the amount of the time the individual had to wait before being served by an elevator. The waiting time associated with each individual represents its fitness, i.e., higher the waiting time and higher fitness. We consider a higher waiting value to be better since we aim to test a dispatcher under uncertainties and a higher waiting time means that the dispatcher's QoS is not good.

2) Parents Selection, Crossover, and Mutation Operators: GA is guided by a set of search operators, i.e., parents selection, crossover operator, and mutation operator. In terms of parents selection, we employed a simple selection criteria, where we pick $npa/2$ number of parents with the highest WTs, where npa is the population size. The parents are sorted according to their WTs as shown in Figure 4. Assuming P_1 has the highest WT, whereas P_{npa} has the lowest WT. We form the pairs (Pair 1, Pair 2, .. Pair npa/2) as shown in the figure. Then, we select the first $(npa/2)/2$ pairs of parents, i.e., Pair 1, Pair 2, .., Pair (npa/2)/2.

On each pair of parents (e.g., *Parent 1* and *Parent 2* in Figure 5), we apply our crossover operator as shown in Figure 5 to produce four children (*Child 1* to *Child 4*). To do crossover, we randomly chose 2 crossover points (e.g., cp_1 and cp_2) shown in the figure out of three possible crossover points (note that a crossover point is unique). We then swap the attributes to the right side of a crossover point to produce two children. In our example shown in Figure 5, when we have chosen cp_1 and cp_2, it resulted into four children as shown in the figure.

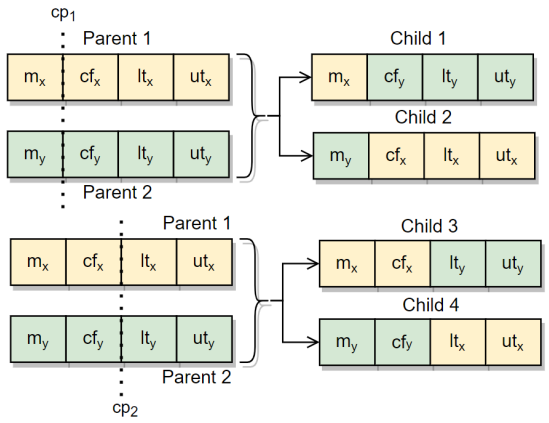

Fig. 5: Crossover Operator

Finally, we defined a mutation operator to mutate each child as follows. For each child, we select 1, 2, 3, or 4 of the attributes for mutation with a mutation probability mp. If the attribute is selected for mutation, we mutate its values by replacing its current value with a random value chosen from a predefined interval for each of the attribute. For instance, if m_x is selected, and assuming a predefined range of 70KG to 80KGs, we will pick a random value within this range and replace it with the current value of m_x.

Note that, in addition to the above-presented operators, other operators can be defined. We will investigate other possible operators and experiment with them in future works.

3) Fitness Function: Since a GA works by using fitness as guidance, we employed QoS attributes to guide the GA towards generating uncertain traffic passengers that lead to high values of QoS attributes (e.g., high AWTs), i.e., unacceptable quality. An AWT is simply calculated by taking the average WTs of npa individuals. There are different ways to acquire AWTs during simulation in each generation of a GA to guide the creation of next npa passengers. The strategy we implemented waits until all the last npa passengers have finished their trips to acquire exact AWT of the last np passengers. Due to this reason, the first generation of passengers (npa) has exactly the same values of arrival times as the profile, whereas the arrival times of passengers in the next generations are delayed. As a result, the traffic generated by GA is a bit different than the original traffic profile. Other strategies will be investigated in the future.

IV. Validation

We present the results of an initial validation on an industrial dispatcher provided by Orona. First, we present the experimental settings for SiL in Section IV-A, and GA in Section IV-B. Second, we present the results in Section IV-C. Third, we present threats to validity of our experiment in Section IV-D.

A. Settings for Software in the Loop Simulation

We picked one traffic generation pattern provided by Elevate with the standard settings provided by Elevate. We set the profile to generate traffic from 8 AM to 10 AM. Orona provided us with the dispatcher algorithm implemented in C++ that extended with our implementation of the GA as shown Figure 2. We used a default building configuration provided in Elevate, i.e., with eight floors. We used two elevators, with each having a maximum capacity of 1000 KGs and the rest were the default configurations in Elevate. For simulation, we also used the default settings except for number of repetitions for simulations that was set to 1. As a comparison baseline, we used the same profile but with the fixed values for mass, capacity factor, loading time, and unloading time, meaning that it had no uncertainty in these attributes.

B. GA Settings

We set the population size to 20, i.e., 20 passengers. To perform crossover, we picked two crossover points randomly out of the three possible crossover points that we have in our individual representation. For the mutation operator, we used a mutation probability of 5%, i.e., each of the four parameters (i.e., mass, capacity factor, loading time, and unloading time) has 5% chance to be selected for mutation. For each of these parameters, we chose the following ranges to pick a mutated value; 1) Mass: 65KGS to 85KS; 2) Capacity factor: 60 to 80; 3) Loading and Unloading times: 1 to 2 seconds.

To enable using the GA to follow the selected traffic profile from Elevate as closely as possible, we set the arrival time, arrival floor, and departure floor of each passenger as closely as possible to the one specified in the profile. The rest of the four parameters, i.e., mass, capacity factor, loading time, and unloading time, were controlled by GA to maximize the QoS attribute, which was AWT in this paper. Given the number of passengers in the chosen traffic profile, GA was evolved up to 63 generations. To determine whether GA manages to increase the AWT with uncertainty in parameters, we compared its AWT per generation with the one for the chosen profile. To account for randomness, we repeated GA 10 number of times to ensure that the results are not obtained by chance.

C. Results and Discussion

Figure 6 shows the results of comparing GA with the results of not using GA, i.e., baseline profile with fixed values of uncertainty attributes (*No GA* in the figure). The x-axis shows the number of generations, whereas the y-axis shows the AWT in seconds. Note that for *No GA*, it is one AWT value, whereas for GA it is an average of 10 AWT values corresponding to the 10 independent runs.

As we can see from the figure that without GA, i.e., without uncertainty, AWT stays relatively low throughout the simulation. In contrast, with the introduction of uncertainties with GA, one can see that AWT continues to rise until the simulation is finished. The results showed that over the generations, GA managed to increase the AWT as compared with *No GA*, suggesting that uncertainties in the passenger data, i.e., in particular in mass, capacity factor, loading time, and unloading time, affect the AWT of the dispatcher.

To further assess the statistical significance of the results, we did a one-sample Wilcoxon test at the significance level of

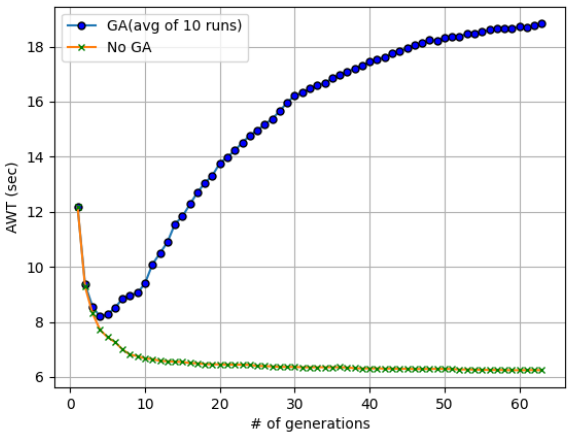

Fig. 6: Results of Studying the Impact of Uncertainties

0.05 to compare the AWT from 10 runs of GA (last generation) with 1 AWT value of *No GA*. The test revealed a p-value less than 0.05 suggesting that GAUTE leads to significantly higher AWT than the baseline. These results suggest that uncertainties in the studied uncertainty attributes lead to significantly higher AWT than without uncertainties.

In general, our results indicate that uncertainty plays a crucial role in affecting the QoS of elevators, and thus, it should be explicitly studied. GAUTE is such a testing framework that will allow our industrial partners to experiment with different types of uncertainties during testing.

D. Threats to Validity

Our current experiments have several threats to validity that we plan to address in the future. Examples of such threats include using one dispatcher algorithm, one profile, one set of configurations (building, simulation, GA), and only one search algorithm. Thus, to further increase the confidence in the results, we need experiments with variants of dispatcher algorithms, different traffic profiles, and configurations. Moreover, we experimented with one set of GA parameters (e.g., 5% mutation probability), and different settings may lead to better results. Thus, we aim to perform parameter tuning to find the best settings in the future work.

Randomness affects the results of search algorithms like GA; therefore, we repeated it ten times to deal with randomness. However, we need to run simulations an additional number of times to gain further confidence in the results. Finally, we performed a relevant statistical test to compare *GA* with *No GA* approach.

V. Lessons Learned

In this section, we present lessons learned and implications of our results from industrial and research perspectives.

A. Uncertainty as the First Class Entity

Our preliminary experimental results show that uncertainty in passenger data affects the AWT. Thus, we argue that uncertainties must be considered explicitly and as first-class entities during testing of elevators. Doing so will: (1) help understanding the relationships among various uncertainties and QoS attributes; (2) identify unforeseen situations such as too long waiting times; (3) improve dispatcher algorithms to identify and handle unforeseen situations; (4) building new self-healing and recovery mechanisms to deal with unforeseen situations.

B. Utilization of Real Operational Data

As part of our project, we can gain access to operational data of real installations. Such data can be used instead of the traffic profile from Elevate to further study uncertainties with more realistic data. Moreover, we set the ranges of uncertainty attributes based on the knowledge of domain experts rather than extracting such information from operational data. This was required since data about mass, capacity factor, loading, and unloading times are not collected for each individual passenger during the real operation.

C. Technology Transfer

The current tool-supported methodology version we report in this study is a prototypical implementation intended for research purposes. However, despite its prototypical implementation, based on our conversations with practitioners, we believe that the steps towards the transfer of the tool are straightforward. Specifically, we envision the following steps: (1) consider feedback taken from industrial practitioners and polish certain aspects of the tool (e.g., how to automatically tune GA parameters, how to include data from the real operation, and how to make analyses more detailed); (2) perform a more careful and complete empirical evaluation, by considering more aspects than the ones considered in this paper (e.g., more buildings, further experimental runs, statistical tests); (3) train engineers from Orona to use our tool; (4) establish a tool maintenance plan.

D. Research and Industrial Implications

In terms of industrial implications, GAUTE provides new insights on testing dispatcher algorithms in the presence of uncertainties. Thus, our developed tool can be used by Orona to test their various dispatcher algorithms with different traffic profiles to improve their quality and further study how well their dispatchers deal with uncertainties. Such a study can help Orona to improve their QoS further.

In terms of research, our work is preliminary; thus, it opens up several new directions for researchers to investigate. First, one can develop cost-effective genetic operators since our operators are an initial attempt to demonstrate the working of the methodology. Second, researchers can also investigate the development of specialized algorithms for testing dispatchers under uncertainties. Third, we only experimented with GA, and other existing algorithms such as multi-objective algorithms (e.g., NSGA-II) and machine learning algorithms (e.g., reinforcement learning) could also be tried.

E. Current Limitations

Currently, there are several limitations. First, we only studied four passenger parameters. However, other parameters such as arrival time, arrival floor, and destination floor could introduce uncertainty as well. Second, we only considered uncertainty in passenger data. However, uncertainty also exists in other aspects such as hardware, e.g., related to delays in opening and closing doors and other hardware errors. Third, the interactions between different uncertainties, i.e., among passenger data and hardware-related uncertainties, need to be studied together.

VI. Related Work

With the increase on the autonomy of CPSs, testing them has become a challenge. Therefore, in the last few years, the research community has devised novel approaches to testing CPSs from several perspectives, including test generation [8], test selection and prioritization [9] and debugging [10]. Unlike these approaches, the method we propose is focused on modifying a test input in the form of Passenger File with the goal of finding uncertain situations when testing an industrial elevator dispatching algorithm.

Techniques for testing systems of elevators and their subsystems has also been proposed in the past. Sagardui et al., proposed an automated technique for testing configurable embedded software in charge of controlling the opening and closing of the doors of elevators [2]. Ayerdi et al., focused on studying whether metamorphic testing was an appropriate technique to automate the execution of tests of the dispatching algorithm [3], whereas Arrieta et al. proposed a machine-learning based approach [4]. In these studies, however, the uncertainty at which the elevators are exposed to is not considered, which is the core aspect of our method.

Uncertainty testing in CPSs is essential to ensure that CPSs handle uncertain situations and do not fail. To this end, several uncertainty testing techniques have been developed for CPSs in various domains. For instance, Zhang et. al [11] developed an uncertainty testing technique called *UncerTest* that was applied to test CPSs in the logistics and sports domains. Another example is work by Shin et al. [12], who developed an uncertainty testing technique for satellites. Uncertainty testing is also of interest in self-driving cars [13]. In contrast, our focus is in the context of industrial elevators with a particular focus on passenger data in an industrial context. Thus, our representation of search problem and various operators are new and specific to the elevators domain.

VII. Conclusion and Future Work

We presented GAUTE, a tool supported methodology for uncertainty testing of industrial dispatcher developed by Orona. In particular, we focused on uncertainties in passenger data with Software in the Loop (SIL) configuration. We presented our domain-specific representation of the search problem followed by the development of new genetic operators. We also presented the overall architecture of the tool support. The methodology can be tailored with various GA operators depending on the requirements. We validated GAUTE with one traffic profile and a building configuration. Results showed that the GA could manage to decrease the quality of service compared to the chosen profile.

Our future work will be along the following lines. First, we will improve our experiments, e.g., by considering more dispatchers, profiles, search algorithms, and additional uncertainty parameters. Second, we will validate GAUTE with the help of practitioners from Orona to assess its applicability, e.g., by conducting questionnaires. Third, we would like to assess the feasibility of using this approach in the Hardware in the Loop (HiL) setting, which is a common practice in Orona.

References

[1] C. F. Nicolas, I. Ayestaran, I. Martinez, and P. Franco, "Model-based development of an fpga encoder simulator for real-time testing of elevator controllers," in *2016 IEEE 19th International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 2016, pp. 53–60.

[2] G. Sagardui, L. Etxeberria, J. A. Agirre, A. Arrieta, C. F. Nicolás, and J. M. Martin, "A configurable validation environment for refactored embedded software: an application to the vertical transport domain," in *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2017, pp. 16–19.

[3] J. Ayerdi, S. Segura, A. Arrieta, G. S. Arratibel, and M. Arratibel, "Qos-aware metamorphic testing: An elevation case study," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 104–114.

[4] A. Arrieta, J. Ayerdi, M. Illarramendi, A. Agirre, G. Sagardui, and M. Arratibel, "Using machine learning to build test oracles: an industrial case study on elevators dispatching algorithms," in *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*. IEEE, 2021, pp. 30–39.

[5] J. Ayerdi, A. Garciandia, A. Arrieta, W. Afzal, E. Enoiu, A. Agirre, G. Sagardui, M. Arratibel, and O. Sellin, "Towards a taxonomy for eliciting design-operation continuum requirements of cyber-physical systems," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 2020, pp. 280–290.

[6] G. Barney and L. Al-Sharif, *Elevator traffic handbook: theory and practice*. Routledge, 2015.

[7] CIBSE, "Transportation systems in buildings," *Cibse Pub. London*, 2010.

[8] S. Abbaspour Asadollah, R. Inam, and H. Hansson, "A survey on testing for cyber physical system," in *Testing Software and Systems*, K. El-Fakih, G. Barlas, and N. Yevtushenko, Eds. Cham: Springer International Publishing, 2015, pp. 194–207.

[9] U. Markiegi, "Test optimisation for highly-configurable cyber-physical systems," in *Proceedings of the 21st International Systems and Software Product Line Conference - Volume B*, ser. SPLC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 139–144. [Online]. Available: https://doi.org/10.1145/3109729.3109745

[10] M. Marra, E. G. Boix, S. Costiou, M. Kerboeuf, A. Plantec, G. Polito, and S. Ducasse, "Debugging cyber-physical systems with pharo: An experience report," in *Proceedings of the 12th Edition of the International Workshop on Smalltalk Technologies*, ser. IWST '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3139903.3139913

[11] M. Zhang, S. Ali, and T. Yue, "Uncertainty-wise test case generation and minimization for cyber-physical systems," *Journal of Systems and Software*, vol. 153, pp. 1–21, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121219300561

[12] S. Y. Shin, K. Chaouch, S. Nejati, M. Sabetzadeh, L. C. Briand, and F. Zimmer, "Uncertainty-aware specification and analysis for hardware-in-the-loop testing of cyber-physical systems," *Journal of Systems and Software*, vol. 171, p. 110813, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121220302132

[13] Z. Huang, M. Arief, H. Lam, and D. Zhao, "Evaluation uncertainty in data-driven self-driving testing," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1902–1907.