Patterns of Applied Control for Public Health Measures on Transportation Services under Epidemic

Kenneth Johnson
Auckland University of Technology
Auckland, New Zealand
kenneth.johnson@aut.ac.nz

Samaneh Madanian
Auckland University of Technology
Auckland, New Zealand
sam.madanian@aut.ac.nz

Catia Trubiani
Gran Sasso Science Institute
L'Aquila, Italy
catia.trubiani@gssi.it

ABSTRACT

The recent trend of uncontrolled spread of infectious diseases has resulted in severe disruption to society on a worldwide scale. One of the causes is represented by public transportation services that contribute to the spread of an epidemic, which has consequences for human health and the economy. Public health organisations call for support from scientists to identify the main criticalities and take countermeasures on time. This paper aims to support public authorities by exploiting the capability of self-adaptive systems (SaS) to autonomously modify their behaviour when subject to changes in their environment. We inherit from the literature patterns that integrate distributed and central control of SaS, and we demonstrate the effectiveness of these design patterns when deciding public health measures applied to transportation services during an epidemic. Our novel methodology consists of modelling and analysing control action types that describe a unique interaction between a central controller and a distributed controlled transportation system to get a desired adaptation. We rely on probabilistic model checking to provide formal guarantees on the expected number of infections determining the epidemic evolution. Experimental results show that our technique is adequate to counteract the epidemic scenarios, thus supporting public health authorities in monitoring the status of transportation services and making informed decisions.

KEYWORDS

Self-adaptive systems, epidemic, distributed control, central control, probabilistic model checking, patterns

ACM Reference Format:
Kenneth Johnson, Samaneh Madanian, and Catia Trubiani. 2024. Patterns of Applied Control for Public Health Measures on Transportation Services under Epidemic. In *19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '24), April 15–16, 2024, Lisbon, AA, Portugal.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3643915.3644091

1 INTRODUCTION

An *epidemic* is the rapid spread of an infectious disease infecting a large number of people within a population in a short amount of time [19]. Public health measures [1, 3] aim to mitigate or reduce the impact of infectious diseases, through different approaches including isolation of those infected, quarantine of exposed individuals, promoting personal protective equipment usage, and social distancing. However, when the population is highly mobile the spread of an epidemic is worsened [11]. Public health measures must therefore act on public transportation services, e.g., by identifying the connections that are most at risk (due to a high number of infections) and limiting travellers' usage of such areas [7].

Modelling and simulation of the spread of diseases [2, 5, 22] is a key factor for deciding public health countermeasures [10] that are effective against future outbreaks or epidemics. The research community recently proposed some self-adaptive healthcare policies [12, 20, 25] that have shown the benefits of integrating adaptation to develop measures used during evolving public health emergencies. In this context, the trade-off is between the restrictiveness of the measures and projected health-care benefits. However, to the best of our knowledge, there is currently little work in the direction of providing formal guarantees on quantifying the impact of public health measures. To this end, we foresee the capability of self-adaptive systems (SaS) as a paramount occasion in this domain, since SaS show the unique characteristic of being autonomous when adapting to changes in their environments [26], hence sensing epidemic scenarios and reacting accordingly.

Our work takes inspiration from the literature, specifically we make use of a catalogue of five patterns (i.e., command, constraint, isolation, influence, and pseudo-emergence) that have been defined in [15] and show the peculiar characteristics of combining distributed and central control in SaS. Kroher et al. [15] demonstrate that the patterns provide the foundation for selecting the appropriate action types for a certain situation as needed, and this constitutes the motivation for our work. In this paper, we exploit these patterns of applied control to provide formal guarantees while evaluating the evolution of an epidemic, in the context of public transportation services. More specifically, we model the aforementioned five patterns and we evaluate their effectiveness on the *Public Transportation (PT)* system that is formulated as SaS under an epidemic scenario.

Summarising, our main contributions are:

(i) formulation of the PT system under epidemic as a SaS
(ii) development of formal models for five design patterns that are assessed in the literature as relevant to control SaS [15];
(iii) adoption of probabilistic model checking analysis technique to support public health authorities in quantitatively understanding the evolution of epidemic scenarios,

SEAMS '24, April 15–16, 2024, Lisbon, AA, Portugal

ACM ISBN 979-8-4007-0585-4/24/04…$15.00
https://doi.org/10.1145/3643915.3644091

(iv) an open-source implementation of the PT system[1],
(v) experimental results that quantitatively compare the impact of the five patterns along with a scalability evaluation, thus demonstrating the effectiveness of our approach.

The rest of the paper is organised as follows. Section 2 presents the PT system and clarifies its connection with SaS. Sections 3, 4 and 5 illustrate the main technicalities when modelling and analysing an epidemic's evolution. Section 6 focuses on quantifying the impact of control action types patterns [14], together with some scalability considerations. Section 7 discusses threats to validity and the main limitations of our approach. Section 8 argues on state-of-the-art approaches that are more closely related. Section 9 provides concluding remarks and outlines future research directions.

2 THE PUBLIC TRANSPORTATION SYSTEM

Figure 1 provides a birds-eye view of the system we use as our motivating scenario and shows the distinct characteristics of self-adaptive systems. We foresee a *Control Unit* that manages the *Public Transportation* (PT) that represents the *Managed System*. PT includes a set of *entities* that are connected through relations of *interaction* or *impact*. The overall goal of the system is to reduce the risk of infection for the population, and the control unit can trigger adaptation at different levels of granularity, depending on the involved entities.

First, the *Transportation Network N* entity represents the various commuting options available within an urban area, servicing a population with fixed routes. We represent the transportation network mathematically as a directed graph $G = (V, E)$ comprising vertices from the set $V = \{v_1, \ldots, v_n\}$ and directed edges from the set $E = V \times V$ of pairs of vertices, such that $(v_i, v_i) \notin E$, for all $v_i \in V$. Figure 2 shows a simple example of a transportation network, comprising seven vertices $v_1, \ldots, v_7$ with interconnecting bi-directional lines representing travel routes, labelled with the distance between the vertices. These lines are short-hand for two weighted, directed edges between vertices such that $(v_i, v_j) \in E$ if, and only if, $(v_j, v_i) \in E$. For convenience, we define a network *location* l of G as either a vertex or edge in G. In symbols, $l \in V \cup E$, which we write as $l \in G$.

Second, *Population* is the entity that collectively represents the users of public transportation within the city. They *directly interact* (depicted in Figure 1 as a bi-directional line) with the Transportation Network, since they travel from a source to a destination location. The Transportation Network provides the *journey planner* to support travellers in planning their journey.

Third, the *Epidemic* entity represents the level of infection to which the population is exposed in case of close exposures. The relationship between the population and the epidemic is managed by a central node, namely *Travel Risk Balance* to denote that the travellers are subject to a certain evolution of the infectious disease. After an incubation period, exposure to the disease causes a susceptible traveller to become infected, and capable of spreading the contagion to other susceptible travellers. The evolution of the epidemic follows the *Susceptible-Exposed-Infectious-Recovered (SEIR)* model [2, 5] that will be detailed later, see Section 3.

[1] A Java implementation of the Control Unit and entities of the Public Transportation SaS, including all models and experimental results are publicly available [13].

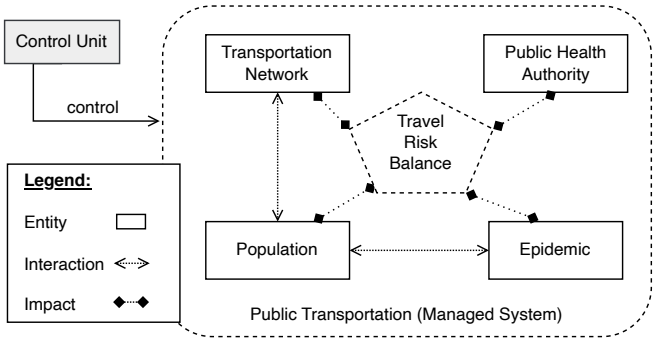

Figure 1: Public Transportation as a Self-Adaptive System

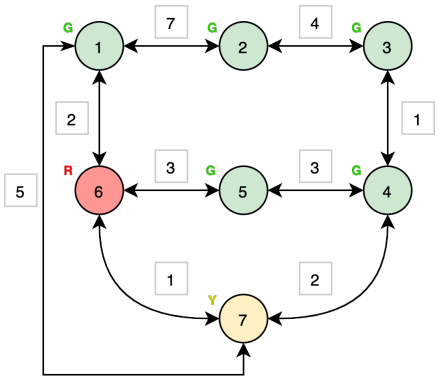

Figure 2: Transportation Network Exemplary Example

Fourth, the *Public Health Authority* (PHA) entity performs epidemic surveillance, i.e., it is in charge of tracking exposures [23]. Using the SEIR model, PHA performs an analysis to determine suitable advice to control the evolution of an epidemic. To simplify decision-making, PHA specifies categories of exposure risk, i.e., `R: RED` indicates a high risk of exposure, `Y: YELLOW` moderate risk, and `G: GREEN` means low risk. Each vertex of the travel network N is labelled with R-Y-G depending on the majority of travellers commuting in such a node and showing a certain exposure risk. For instance, in Figure 2, v_6 is `RED`, v_7 is `YELLOW`, and all other vertices are `GREEN`.

Entities in the PT system observe their environment via monitored variables and react via controlled variables [14]. Table 1 presents the *monitored* and *controlled* parameters for each entity of the PT system. Formally, we model each entity as a tuple $\sigma = (p_1, \ldots, p_m)$ of parameter values in the product set $\mathbf{P} = P_1 \times \cdots \times P_m$, where P_i is the type of data stored in parameter p_i for $1 \leq i \leq m$. Both monitored and controlled parameters are observable; e.g. there is an operation $get_i : \mathbf{P} \to P_i$, such that $get_i(\sigma) = p_i$, the i^{th} parameter of the tuple σ. If p_i of σ is a controlled parameter, then there also exists an operation $set_i : \mathbf{P} \times P_i \to \mathbf{P}$ that allows p_i to be set to new values by the entity. In symbols, $set_i(\sigma, v)(p_j) = v$, if $i = j$ and $\sigma(p_j)$ otherwise. When clear from the context, we write $set_i(\sigma, v)$ in programming notation $p_i := v$.

Entity	Monitored Variables	Controlled Variables
Transportation Network	occupancy	capacity journey planner
Population	infection status	compliance location journey
Epidemic	none	none
Public Health Authority	SEIR model	recommendation prevention

Table 1: Public Transportation – Entity Variables

Population Entity Parameters. Let T be the set of all travellers on the transportation network. We model the traveller entity $t \in T$ by the tuple $\sigma_t = (s, c, l, j)$ such that

- $s \in M$, represents the current infection status as a SEIR model compartment $M = \{S, E, I, R, Q\}$, more details are provided in Section 3,
- $c \in \mathbb{B}$, represents the compliance status determining adherence to public health authority recommendations,
- $l \in G$ is the current location of the traveller,
- $j \in Path$ is the current journey of the traveller, represented by a finite path of the graph G.

All parameters have set operations, but parameter s is restricted such that we may only set $s := Q$ to represent a traveller in quarantine. We name this set operation as $setQuarantine$. We fix $c := true$ for our work which means that all travellers comply with public health authority recommendations at all times.

Transportation Network Entity Parameters. We model the transportation network entity by the tuple $\sigma_N = (G, T_l, k_l, test_l, mode)$ such that G is the network graph. For any location $l \in G$ we specify the parameters

- occupancy, $T_l \subseteq T$, the set of travellers whose current location is l,
- capacity, $k_l \in \mathbb{N}$, sets the limit to a number k on the number of travellers allowed at l at any one time,
- $test_l \in \mathbb{B}$, is $true$ if a testing station is set up at location l and $false$ otherwise,
- journey planner, i.e., the $mode \in \{\textbf{short}, \textbf{safe}\}$, and it denotes the operational modes of the journey planner.

The interaction between the Transportation Network and the Population is due to the strict relationship these two entities show. In particular, the capacity k_l, testing station $test_l$ and $mode$ parameters have set operations which we name as $setCapacity$, $setTesting$ and $setMode$, for $l \in G$.

Epidemic Entity Parameters. While there are no parameters for this entity, there is an interaction with the Population due to regulating the infection among travellers. We model this as the operation $infect : T \to T$ such that $infect(t)$ infects traveller $t \in T$ if they are susceptible. Otherwise, the infection status of t is unmodified.

Public Health Authority Entity Parameters. We model this entity as the tuple $\sigma_{PHA} = (prev, M, rec)$, such that

- $prev$ is a scalar measuring effectiveness of preventing infections, where $prev = 1.0$ is the least and $prev = 0.0$ the most effective preventative measures,
- M is the SEIR infection model, formed from observing traveller locations and infection status,
- $rec : G \to \mathbb{P}$ is the *recommendation* obtained from SEIR analysis of M.

The $prev$ prevention parameter has the set operation named $setPrev$, and the rec recommendation has the get operation named $getRec$.

3 PUBLIC HEALTH AUTHORITY

The role of the *Public Health Authority* is to monitor the risk of infection for the population using the transportation network. PHA models the spread of an epidemic using the *Susceptible-Exposed-Infectious-Recovered (SEIR)* model [2, 5] that partitions the population of n travellers into compartments. At every point in time, each traveller in the population is in exactly one compartment with the following meanings: S means a traveller is *Susceptible* to catch the disease, E means a traveller has been *Exposed* to the disease and is in the incubation period. The traveller is infected but does not yet transmit the disease to others; travellers in compartment I are *Infectious* and spread the disease in case of close exposure, while travellers in R have recovered from the infection and will no longer be infected by the disease.

The SEIR compartments correspond to traveller t's disease parameter $s \in \sigma_t$, as they transition from S to E when they become infectious, E to I when they go through the disease's incubation period, and from I to R when they transition from infectious to a recovered state. We extend the four standard SEIR compartments by adding the *quarantine* compartment Q which corresponds to travellers placed in isolation during their infectious period, hence not spreading the disease to others.

Following the stochastic SEIR model developed in [5], let $s(t)$, $e(t)$, $i(t)$, $r(t)$ and $q(t)$ be differentiable functions denoting the ratio of travellers belonging to compartments S, E, I, R and Q, respectively, for a given time instant $t \geq 0$ such that $s(t) + e(t) + i(t) + r(t) + q(t) = 1$. The transition of the population through the compartments is defined by the ordinary differential equations listed in (1) and (2), modified from [5] whereby

$$\frac{\mathrm{d}s}{\mathrm{d}t} = -\beta i s, \qquad \frac{\mathrm{d}e}{\mathrm{d}t} = \beta i s - \epsilon e, \qquad \frac{\mathrm{d}i}{\mathrm{d}t} = \epsilon e - \chi i - \gamma(1-\chi)i \tag{1}$$

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \chi i \qquad \frac{\mathrm{d}r}{\mathrm{d}t} = \gamma(1-\chi)i. \tag{2}$$

The β parameter is the contact rate, given as the mean of a Poisson distribution to determine the number of close contacts between susceptible and infectious travellers, $\frac{1}{\epsilon}$ is the disease's incubation period, given as the mean of an exponential distribution, $\frac{1}{\gamma}$ is the infectious period, given as the mean of an exponential distribution, χ, the ratio of quarantined travellers and $\frac{1}{\rho}$ the quarantine period. In our work, we assume that travellers stay in quarantine until their infectious period ends, after which they transition to R.

3.1 Formal Verification of the SEIR Model

PHA makes use of the PRISM high-level modelling language [17] to express the SEIR compartment mechanism as a module following

the construction in [21] and presented in Listing 1. Line 1 clarifies that it is a *Continuous-Time Markov Chain (CTMC)* model. The state space is a discretisation from the real-typed variables of SEIR to the product of integer-typed variables `s`, `e`, `i`, and `r` in the range `[0..N]` declared in Lines 11 to 14, where `N` is a constant integer-typed value. These variables encode current numbers of travellers in the corresponding compartments of S, E, I, and R, initialised with data values `s0`, `i0`, `e0`, and `r0`, obtained from epidemic monitoring activities. To keep the model simple, we do not include the Q compartment. Transitions between compartments are given in Lines 16, 17, and 18 and are defined by guarded commands that determine the rates at which values of the module's variables change, specifically they present the following form:

$$[action]\ guard -> r_1 : update_1 + \cdots + r_n : update_n \quad (3)$$

where the command is labelled `action` and *guard* is a Boolean-typed expression over the model variables. If the guard is satisfied then $update_1, \ldots, update_n$ change the model variables according to positive value rates $r_1, \ldots, r_n$, which are input parameters to exponential distributions.

```
1  ctmc
2  const double beta = 1.3;
3  const double epsilon = 1/3.21;
4  const double gamma = 1/2.27;
5  const int N = 100;const double L;const int k;
6  const double prev = 1.0;
7  const int s0=99;const int e0=0;
8  const int i0=1; const int r0=0;
9
10 module seir
11  s: [0..N] init s0;
12  e: [0..N] init e0;
13  i: [0..N] init i0;
14  r: [0..N] init r0;
15
16  [ex] (s>0&i>0&e<N) -> beta*s*i*prev/N:(s'=s-1)&(e'=e+1);
17  [in] (e>0&i<N)     -> epsilon*e :(e'=e-1)&(i'=i+1);
18  [re] (i>0&r<N)     -> gamma*i:(i'=i-1)&(r'=r+1);
19 endmodule
```

Listing 1: Prism Module of SEIR Compartment Model

3.2 Analysing Exposure Risk at a Vertex

To determine the exposure of being infected in vertex v, PHA initialises the constants, see Lines 2 to 8 in Listing 1. First, it is necessary to set the SEIR model parameters to appropriate values. For instance, we can set the SEIR model parameters to values estimated in [5] for modelling the COVID-19 pandemic. The social distancing factor parameter is set to `dist=1.0` meaning that there is no social distancing measure in place to inhibit exposure. Second, it is necessary to quantify disease states of travellers within the *local* of v: either at vertex v or any incoming edge of v (e.g., travellers currently en route to shortly arrive). For example, we suppose there are `N=100` travellers local to v, with one infectious case, i.e., `i0=1`, and all other compartment data values are set to zero.

To analyze the risk of disease exposure at vertex $v \in V$, PHA verifies the *Continuous Stochastic Logic (CSL)* [4] property

$$\texttt{P=?[F < k (i >= L)]} \quad (4)$$

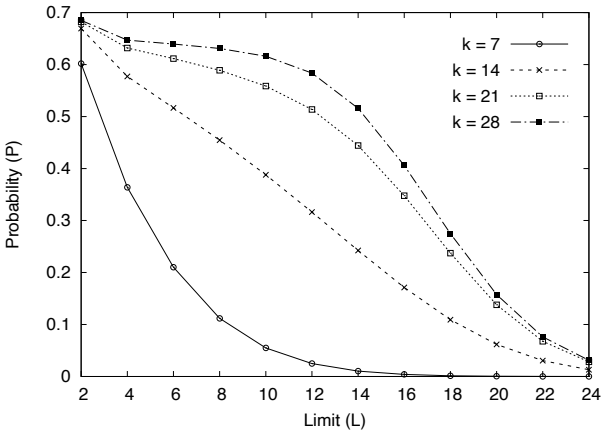

Figure 3: Probability of the number of infection cases at vertex v exceeding the safety limit threshold `L` after `k` days

against the *CTMC* model expressed by the `SEIR` module. Property (4) is formed from the `P` operator to return the probability value of reaching a model state in less than `k` steps, such that the number of infectious travellers at v will reach or exceed the threshold limit `L`.

Figure 3 shows the results of the PRISM experiments we ran following [21] to verify Property (4) on `SEIR`, i.e., predicting the probability of the number of infection cases meeting or exceeding the threshold of `L` after `k` days. We ran experiments ranging the safety threshold limit `L` from 2 to 24 infectious cases and `k` over 7 days (1 week) to 28 days (4 weeks). For example, from these results, we observe that the probability of having more than 10 infection cases in 7 days is lower than 0.1, whereby it increases during longer observation windows: 0.4 after two weeks, and up to 0.65 after four weeks. These results support PHA in early intervention to manage the outbreak situation. Figure 3 shows the probability of the threshold limit being hit is considerably lower after 7 days compared to 14, 21, or 28 days. Thus, we fix `k = 7` and the value `L` is fixed according to current public health measures. Verification of Property (4) against the `SEIR` module of vertex v yields the probability value p_v, which measures the *exposure risk* at v.

3.3 Health Recommendations

To simplify decision-making using verification results obtained by PRISM, PHA uses $C = \{Green, Yellow, Red\}$ to categorise levels of exposure risk. Formally, the categories in C represent a set of unit interval partitions such that $Green = [0, 0]$, $Yellow = (0, \theta]$, and $Red = (\theta, 1]$, for $0 < \theta < 1$. Given the category $c \in C$, we define corresponding mappings $c_\theta : [0, 1] \to [0, 1]$ such that $c_\theta(p) = c$, is the constant canonical probability value representing the range of exposure risk in each category. The number of categories can be modified to enable more or less fine-grained measurements of exposure risk.

Formally, we define the public health authority's *recommendation* as a function $rec : G \to \mathbb{P}$ of vertices and edges of the graph to probability values such that

$$rec(v) = c_\theta(p_v) \quad (5)$$

is the canonical value for $v \in V$, and $rec(e) = rec(v)$, for each incoming edge $e \in E$ of v.

4 THE TRANSPORTATION NETWORK

The *Transportation Network* entity provides the *journey planning* to the population, i.e., indications on the path to follow from the source to the destination vertices. Formally, the journey planning is represented by a mapping $jour : V \times V \times mode \rightarrow Path$ such that computing of $jour(v_1, v_n, mode)$ uses the network's graph representation $G = (V, E)$ to compute a journey between a source vertex v_1 to a destination vertex v_n as a finite sequence $\omega = (v_1, v_2, \ldots, v_n)$ of vertices in V such that there exists an edge e_i in E connecting v_i to v_{i+1}, for $1 \leq i < n$. We call the sequence ω a *path* between vertices v_1 and v_n. Let $Path$ be the set of all finite paths of G.

4.1 Journey Planner Modes

The journey planner has two *modes* of operation, which determine the properties of the returned $Path$. If $\omega = jour(v_1, v_n, mode)$ then

- when $mode =$ **short**, $jour$ returns the shortest path between v_1 and v_n using Dijkstra's Algorithm, with weighted edges according to distances between locations,
- when $mode =$ **safe**, $jour$ returns the guaranteed safest path between v_1 and v_n, based on PHA recommendation.

While the shortest path can be computed directly from G, the safest path is instead derived via probabilistic model checking verification. PRISM modelling language is adopted as the formalism of choice, and the model is synthesised from the transportation network graph representation G.

```
1  mdp
2  const int v1 = 1;const int v2 = 2;const int v3 = 3;
3  const int v4 = 4;const int v5 = 5;const int v6 = 6;
4  const int v7 = 7;
5
6  const int src = v4;const int dst = v1;
7  const int succ = 8;const int fail = 9;
8
9  const double RED = 0.10;const double YEL = 0.05;
10 const double GRE = 0.0;
11
12 module Network
13  l : [1..fail] init src;
14
15  //vertex v6 is labelled RED
16  [r56] (l=v5)&(l!=dst) -> 1-RED:(l'=v6) + RED:(l'=fail);
17  [r76] (l=v7)&(l!=dst) -> 1-RED:(l'=v6) + RED:(l'=fail);
18  [r16] (l=v1)&(l!=dst) -> 1-RED:(l'=v6) + RED:(l'=fail);
19
20  //vertex v7 is labelled YELLOW
21  [r17] (l=v1)&(l!=dst) -> 1-YEL:(l'=v7) + YEL:(l'=fail);
22  [r47] (l=v4)&(l!=dst) -> 1-YEL:(l'=v7) + YEL:(l'=fail);
23  [r67] (l=v6)&(l!=dst) -> 1-YEL:(l'=v7) + YEL:(l'=fail);
24
25  //vertices labelled GREEN (omitted for brevity)
26  [r21] (l=v2)&(l!=dst)-> 1-GRE:(l'=v1) + GRE:(l'=fail);
27  [r43] (l=v4)&(l!=dst)-> 1-GRE:(l'=v3) + GRE:(l'=fail);
28  [r32] (l=v3)&(l!=dst)-> 1-GRE:(l'=v2) + GRE:(l'=fail);
29  ...
30  [succ] (l=dst)  -> (l'=succ);
31  [fail] (l=fail) -> true;
32 endmodule
```

Listing 2: Fragment of the Synthesised PRISM Module

4.2 Markov Chain Model of the Network

Listing 2 presents a fragment of the `Network` module synthesised from the example shown in Figure 2. Given the graph $G = (V, E)$, the state space `l : [1..fail]` is constituted of the vertices in V, the `succ` and `fail` states. Line 1 identifies the model as a *Markov Decision Process (MDP)*. Lines 2 to 4 are constants numbering vertices `v1` to `v7` with integer-typed values. Line 6 sets constants `src` and `dst`, i.e., the journey's source and destination vertices. The `succ` state represents the successful exposure-free travel, while `fail` represents the state in which travel has resulted in exposure.

MDP module has the same transition command syntax as shown in (3), however $r_1, \ldots, r_n$ are now probability values such that $r_i \in [0, 1]$ is the probability $update_i$ occurs when the guard condition is satisfied, and $\sum_{i=1}^{n} r_i = 1$. Each edge of the graph's structure is encoded by a module transition and the recommendation given by Equation (5) assigns the transition probabilities according to the risk category. The edge $(v_i, v_j) \in E$ is synthesised as the transition

`[rij] (l=`v_i`)&(l!=dst) -> 1 −` p_i`:(l'=`v_j`) +` p_i`:(l'=FAIL);`

with action `rij`, naming the route between vertices v_i and v_j. The guard is satisfied when the state variable `l` is equal to v_i, and is not the destination state `dst`. In this case, the probability of exposure-free travel to v_j is $1 - p_i$; the next state is given by the update command `l'=`v_j. Otherwise, with probability p_i, the next transition is the `FAIL` state as the travel has resulted in an exposure. The probabilities are supplied by PHA recommendation in Equation (5).

The MDP in Listing 2 comprises transitions extracted from our example in Figure 2. Since PHA has set vertex `v6` to high-risk, Lines 16-18 use the constant `RED`, set to `0.10` for transition probabilities of incoming edges to vertex v_6. Similarly, Lines 21-23 uses the constant `YEL`, set to `0.05` for transition probabilities of incoming edges to vertex v_7. The remaining edges are risk-free and use the constant `GRE` set to `0.0`. In this case, there is no possibility of infection, e.g., probability `1 = 1 − GRE` of transitioning to the next state. We omit most of these transitions for brevity. Lastly, Line 30 has transition with guard `l=dst`, meaning that the journey has ended in the destination state successfully, while Line 31 is a self-loop transition in the fail state.

4.3 Safest Journey Guarantee

Suppose that a traveller wishes to travel from source vertex v_4 to destination vertex v_1. We synthesised the corresponding model by setting `src = 4` and `dst = 1`. Note that the initial state of the MDP is set to `src`. To obtain the safest route, we verify the *Probabilistic Computation Tree Logic Formula (PCTL)* [9] property

$$\texttt{Pmax=?[F(l = succ)]} \tag{6}$$

against the MDP model expressed by the `Network` module. MDP shows a non-deterministic choice that must be resolved when there are multiple transitions available at a state. For example, when `l = v1` three transitions are available: `r16`, `r17`, and `r12` which correspond to potential routes a traveller may decide to take. Property (6) is a *reachability* property formed from the `Pmax` operator, which resolves non-determinism into a *strategy*, yielding the maximum probability value of reaching state `succ`. When Property (6) is verified against the synthesised MDP model, we obtain the guaranteed safest journey as the strategy (`v4`, `v3`, `v2`, `v1`). While this strategy

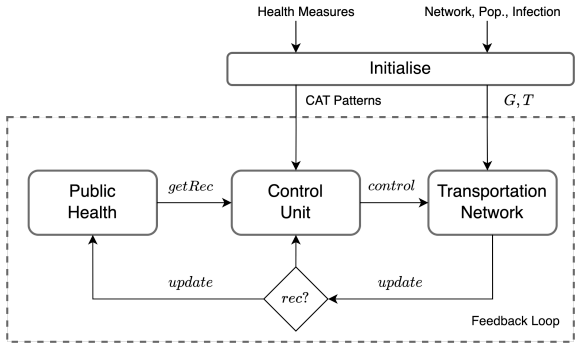

Figure 4: Feedback Loop Workflow

represents the safest path in G, it is not the shortest, but it avoids both the RED and YELLOW labelled vertices.

In summary, when traveller t requests a journey, the result $\omega = jour(source, dest, mode)$ is computed and *sent* to the traveller, such that the journey parameter j is set by the operation $setJourney(\omega)$.

5 PT SYSTEM FEEDBACK LOOP SIMULATION

We implement the self-adaptive feedback loop for the Public Transportation system as an extendable Java library, thus simulating adaptations for several epidemic scenarios. The *Feedback Loop* workflow is depicted as the dashed box in Figure 4, and comprises the control unit, the public health authority, and the transportation network. Each iteration of the loop represents a single day (i.e., stored in the counter variable day), and our simulation is used primarily to record the number of infection cases per day.

5.1 Feedback Loop Initialisation

Before the *Feedback Loop* starts, there exists an *Initialise* phase that is manually performed. It takes as input values for SEIR model parameters β, ϵ, and γ, the transportation network, a population of travellers and their infection status which are translated into the graph data structure G and collection of travellers T; forming entities σ_{PHA} and σ_N of the PT system. The *Initialise* phase translates health measures into CAT patterns that program the control unit's adaptation of the PT system.

For our experimentation (see Section 6), we fix the number of simulated days over which the *Feedback Loop* will run to 100. Moreover, we set one traveller to be infected with the disease. Initially, travellers are placed at a source vertex and given a destination vertex to travel to. We distribute travellers equally across the vertices.

5.2 Feedback Loop Simulation

After initialisation is completed, the simulation of the *Feedback Loop* starts. On day = 1, the PHA monitors the Transport Network and applies the SEIR analysis described in Section 3. Once completed, recommendations are sent to the control unit by the *getRec* command. The Control Unit invokes the *control* operation which conditionally applies CAT patterns to adapt the Transport Network to risks reported by the PHA. We will describe CAT patterns adaptations in detail in Section 6. The *update* determines the next state of all entities (described below) and is sent back to the Control Unit.

The diamond conditional labelled *rec*? is a test to decide if another recommendation from the PHA is needed. Recommendations from the PHA occur weekly, e.g., $rec? \equiv (day \mod 7) = 0$.

Algorithm 1 Update Simulation Parameters

```
1:  Method UPDATE()
2:      for traveller in Population do
3:          cell ← traveller disease status s
4:          if cell > 0 then
5:              cell ← cell − 1
6:              if cell = 0 then cell ← −1
7:              if cell = 100 then cell ← P_I
8:              if cell = 200 then cell ← −1
9:          if cell = 0 then
10:             n ← numberOfCloseContacts(l)
11:             contacts ← getTravellersAtLocation(l, n)
12:             if infected(contacts) then cell ← 100 + P_E
13:         if 0 < cell < 100 then
14:             if testStation(l) and (random value < χ) then
                cell ← 200 + P_Q
15:         l ← travel(l)
16:     day ← day + 1
```

The *update* method's implementation is given by Algorithm (1), which is based on and extends the Stochastic SEIR model developed in [5]. The algorithm runs once per day, iterating over all travellers $t \in T$ in the population (Line 2) and updating each traveller's location and disease state s, represented by an integer value *cell* in the range $[0, 300)$. If $cell = 0$ then t is in the susceptible compartment S. Lines 10-12 determine if t has been exposed by choosing n number of close contacts, by sampling a Poisson process with parameter β. If a traveller is found to be $infected$, then $cell \leftarrow 100 + P_E$, where P_E is sampled from an exponential distribution with parameter ϵ; determining the number of days t is in the *exposed* (s=E) compartment. We added a fixed small social distancing factor of 1.0×10^{-5}, i.e., the probability that t is adequately distanced from others thus avoiding exposure.

Lines 4 to 8 codify SEIR compartment behaviour. When $cell > 100$, it means t is passing through E, $(100 < cell < 200)$, I $(0 < cell < 100)$ and Q $(cell < 200)$. In this last case, a traveller may only go into quarantine if they are infectious and their location l has a test station, modelled by the $testStation(l)$ predicate. If so, then a random value is selected, if less than χ, then we set $cell \leftarrow 200 + P_Q$ where P_Q is a sample of the exponential distribution with parameter χ; the number of days t remains in quarantine (s=Q), and cannot infect any other traveller. If $cell = 200$ then Line 8 sets $cell \leftarrow -1$, and t is now in the recovered compartment (s=R) and cannot be re-infected.

Once the disease status of t is updated, Line 15 updates their location. Travellers in our simulation follow very simple behaviour. They begin at a source location and journey to their destination, based on the results of the journey planner. Once they reach their destination, they again consult the journey planner to return to the source. This behaviour models the most common use case of the network where a traveller journeys between home, workplace, and back again. Lastly, Line 16 increments the day counter variable.

6 CONTROL ACTION TYPES PATTERNS

The *Control Unit* (see Figure 1) supervises and manages the transportation system, its role is to provide adaptation plans to PT system entities. To this end, the Control Until relies on the designed *control action type (CAT) patterns* [14] which give a methodological approach to specifying how the managed system adapts. There are five CAT patterns that we use to express the self-adaptive behaviour of the PT system: *command*, *constraint*, *isolation*, *influence*, and *pseudo-emergence*. This section evaluates the effectiveness of these patterns in reducing the infection risk of travellers.

Signature	Σ_{CAT}
Imports	Σ_C, Σ_K
Operations	$command : Entity \times C \rightarrow Entity$ $constraint : Entity \times K \rightarrow Entity$ $isolation : Entity \times C \rightarrow Entity$ $influence : Entity \times Entity \times C \cup K \rightarrow Entity$ $pseudoEmergence : Entity \rightarrow Entity$

We specify CAT patterns by the Signature Σ_{CAT}, which comprises Σ_C for commands to be executed by entities, and Σ_K constraints to be applied to entities. Our pattern specifications are based on the entity interactions described by the diagrams in [14].

For each pattern, we present experimental results of simulations run over 100 days. Each simulation is instantiated with the transportation network in Figure 2 and a population of 100 travellers, with one traveller initially infected. Presented data series show the average number of *active infections* per day, over an average of ten simulation runs. The parameter values for SEIR compartment modelling are $\beta = 1.3$, $\epsilon = \frac{1}{3.21}$, $\gamma = \frac{1}{2.27}$, averaged from values obtained from the COVID-19 pandemic we inherit from [5].

6.1 Command

The *command* pattern establishes a direct communication between the Control Unit and a *Target* entity. In our system, the Public Health Authority issues measures impacting on the SEIR model.

6.1.1 Pattern Syntax. The syntax of the *command pattern* is of the form $command : Entity \times C \rightarrow Entity$ which takes as input the target $ta \in Entity$, and a command c listed in signature Σ_C such that the command pattern is the term $command(ta, c)$ invoked by the Control Unit, it sends c directly to ta to change the target's parameters. The *execute* operation named by Σ_C is the Control Unit's interface to send commands to a target entity.

Signature	Σ_C
Commands	$setMode(N, mode), setPrev(f)$
Operations	$execute : Entity \times C \rightarrow Entity$

6.1.2 Pattern Semantics. The target's interpretation of *execute* defines the semantics of the command pattern. If the command corresponds to a *set* operation of the target, then its parameter values change accordingly. Otherwise, the command is ignored by the target. In this sense, commands are simply *set* operations applied to the target by the Control Unit. Mathematically, let σ_{ta} be the parameters of the target entity ta and c is the command to set the i^{th} parameter to the value v; e.g. $c \equiv set_i(\sigma_{ta}, v)$. The *execute* operation is interpreted by ta according to the equation

$$execute_{ta}(\sigma_{ta}, c) = \begin{cases} set_i(\sigma_{ta}, v) & \text{if } set_i(v) \text{ defined on } ta \\ \sigma_{ta} & \text{otherwise.} \end{cases}$$

Formally, $[\![command(ta, c)]\!]_{ta}(\sigma_{ta}) = execute_{ta}(\sigma_{ta}, c)$.

By this definition, we have the following commands for the transport network entity:

- $command(N, setMode(N, mode := \{\textbf{safe}, \textbf{short}\}))$ sets the journey planner mode
- $command(PHA, setPrev(f))$, sets $prev := f$, such that the public health authority recommends preventative measures to factor f.

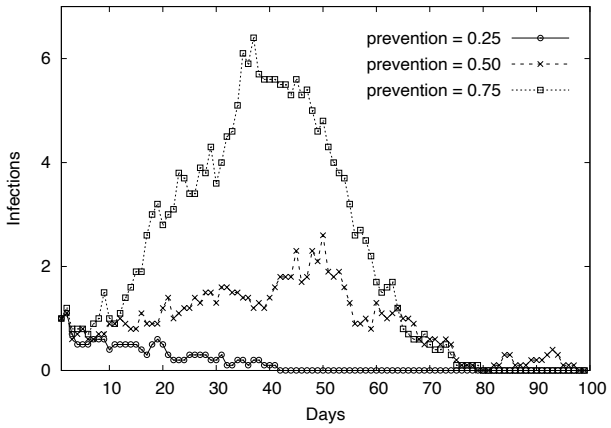

Figure 5: Command pattern

6.1.3 Pattern Adaptation Results. Figure 5 shows the application of the command pattern sent by the Control Unit to the Public Health Authority entity. This causes the PHA to issue health measures corresponding to the supplied effective factor. For instance, PHA can indicate wearing a properly fitted mask, frequent hand washing, and covering the mouth and nose with a bent elbow or tissue in case of cough. Mathematically, the prevention factor ($prev$) scales the behaviour SEIR parameter β; e.g., the parameter of the Poisson process controlling the number of contacts is given by $prev \cdot \beta$.

The command pattern $command(PHA, setPrev(0.75))$ therefore scales β by 0.75, representing mostly ineffective measures to prevent infection. This results in a sharp spike at approximately day 37 of the simulations, then quickly dropping off. The command pattern $command(PHA, setPrev(0.50))$ offers improved protection against infection and flattens the spike, causing the epidemic to last almost to the end of the 100 days. Lastly, the command pattern $command(PHA, setPrev(0.25))$ offers the best protection, resulting in very few infections and with the epidemic ending before day 46 of the simulation.

6.2 Constraint

The *constraint* pattern includes constraints that limit the scope of the behaviour of a *Target* entity. In our system, the Transportation Network limits the circulation of the Population.

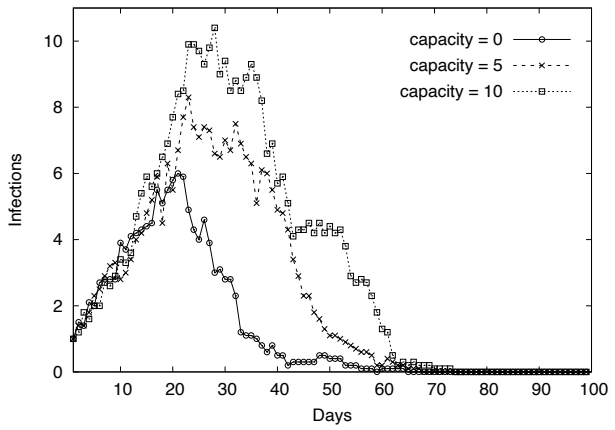

Figure 6: Constraint and Isolation patterns

6.2.1 Pattern Syntax. The syntax of the *constraint pattern* is of the form $constraint : Entity \times K \rightarrow Entity$ which takes as input the target $ta \in Entity$, and a constraint k listed in signature Σ_K such that the constraint pattern is the term $constraint(ta, k)$ invoked by the Control Unit which sends k directly to ta to change the target's parameters.

Signature	Σ_K
Constraints	$setCapacity(l, v)$, $setQuarantine(t)$
Operations	$apply : Entity \times K \rightarrow Entity$

Similarly to the command pattern, when a constraint is sent to a target entity, it is immediately applied and it restricts the target's behaviour. The *apply* operation named by Σ_K is the Control Unit's interface to send constraints to a target entity.

6.2.2 Pattern Semantics. The target's interpretation of *apply* defines the semantics of the constraint pattern. Mathematically, let σ_{ta} be the parameters of the target entity ta and k is the constraint to set the i^{th} parameter to the value v; e.g., $k \equiv set_i(\sigma_{ta}, v)$ acts as a restriction on the behaviour of entity ta. The *apply* operation is interpreted by ta according to the equation

$$apply_{ta}(\sigma_{ta}, k)(v) = \begin{cases} set_i(\sigma_{ta}, v) & \text{if } set_i \text{ defined on } ta \\ \sigma_{ta} & \text{otherwise.} \end{cases}$$

Formally, $[\![constraint(ta, k)]\!]_{ta}(\sigma_{ta}) = apply_{ta}(\sigma_{ta}, k)$.

By this definition, we have the following constraints for the transportation network and population entities:

- $apply(\sigma_N, setCapacity(l, k))$, sets $k_l := v$, which constrains location l to allow only v travellers at any time.
- $apply(\sigma_t, setQuarantine(t))$, sets $s := Q$, which constrains traveller t in quarantine, preventing infections.

6.2.3 Pattern Adaptation Results. Figure 6 shows the application of the constraint pattern sent by the Control Unit to the transportation network and population entities. This causes the network to restrict the number of travellers allowed on a route at any given time. For example, this may mean restricting the occupancy of buses.

The adaption starts when at least 10 infection cases have been observed on the network at any time. This means that the disease was able to replicate through the network unhindered for the first few days of the simulation before the constraint pattern is activated. The Control Unit decides to restrict the capacity of the network by placing a constraint on the number of travellers at a location. Our simulation does not allow travellers to move to the next location in their journey if the maximum capacity is achieved.

Our analysis tests $constraint(N, setCapacity(k = \{10, 5\}))$, i.e., no more than 10 and 5 travellers are allowed through the edges of the transportation network. When $k = 10$, the results show the number of infections peaking between 19 and 37 days, before dropping low; with the epidemic lasting the full 100 days. When $k = 5$, the travellers are further restricted from moving around the network, thus the results show a lower number of daily infections.

6.3 Isolation

The *isolation* pattern allows the Control Unit to disable relationships between entities. In our system, we isolate the vertices of the transportation network, i.e., stopping all the connections.

6.3.1 Pattern Syntax. The syntax of the isolation pattern is of the form $isolation(r)$, which takes as input a relation $r \in R$ between system entities such that $R \subseteq Entity \times Entity$; the set of all entity relationships.

6.3.2 Pattern Semantics. The isolation pattern invokes the *disable* operation, disabling the relation $r \in R$. In symbols, $[\![isolation(r)]\!]_R = disable_R(r)$ such that the implementation of $disable_R$ is determined according to the relationship in R. This can involve simply setting a Boolean-typed relation parameter; $isEnabled := false$. For our work, we use the $setCapacity$ constraint in Σ_K to isolate routes between vertex locations in the transportation network, such that $isolation(e) = disable_E(e)$, for all edges $e \in E$ and $disable_E(e) \equiv setCapacity_e(k = 0)$.

6.3.3 Pattern Adaptation Results. Figure 6 shows the impact on the isolation pattern that implies disabling all edges of the transport network (i.e., the capacity is set to zero). In our simulation, all travellers are locked down into a vertex as any attempt to travel means that the capacity would be exceeded. The pattern is activated after 10 infection cases are detected, and results in a major reduction of infections per day.

6.4 Influence

The *influence* pattern enables the Control Unit to interact with a *Influencer* entity that affects the behaviour of a *Target* entity. In our system, the Transportation Network switches modes of the journey planner, and this impacts the Population.

6.4.1 Pattern Syntax. The syntax of the influence pattern is of the form $influence : Entity \times Entity \times C \cup K \rightarrow Entity$ which takes as input a target $ta \in Entity$ entity, an influencer $in \in Entity$, and either a command or constraint $a \in C \cup K$ such that the influence pattern is the term $influence(ta, in, a)$ invoked by the Control Unit, which sends the command or constraint a to in and changes its parameters. In turn, the behaviour of ta is influenced by information it receives from in.

6.4.2 Pattern Semantics. The influencer's interpretation of the command or constraint defines the semantics involving the changes

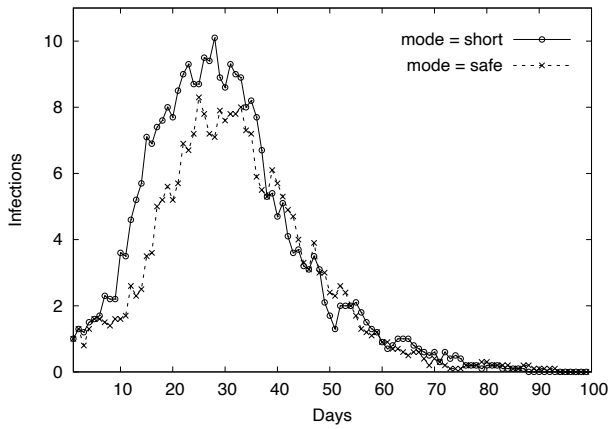

Figure 7: Influence pattern

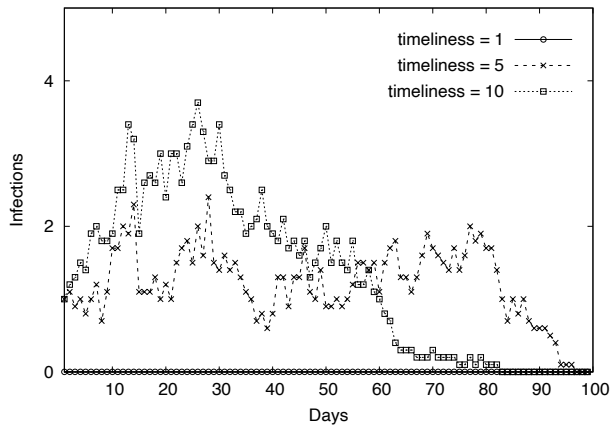

Figure 8: Pseudo-emergence pattern

of parameters. To specify the influence on the target, we define the operation $send_i : Entity \times Entity \times P_i \rightarrow Entity$ such that the equation $send_i(\sigma_{in}, \sigma_{ta}, v) = set_i(\sigma_{ta}, v)$ means that the influencer entity in sends the parameter value v to the target entity ta, setting $p_i := v$ using the corresponding set operation. To define the semantics of the influence pattern, we have the following cases

$$[\![influence(ta, in, a)]\!]_{in}\ (\sigma_{in}) = \begin{cases} execute_{in}(\sigma_{in}, a) & \text{if } a \in C \\ apply_{in}(\sigma_{in}, a) & \text{if } a \in K \end{cases}$$

to handle either commands in C or constraints in K, with the direct interaction $send_i(\sigma_{in}, \sigma_{ta}, v)$ between influencer in and target ta.

Using this definition, we specify the influence pattern

$$influence(\sigma_T, \sigma_N, setMode(N, mode = \{\textbf{safe}, \textbf{short}\})) \quad (7)$$

whereby the target entities are travellers in T who subsequently receive journey plans that follow the operational mode of the journey planner. The command to change the journey planner's *mode* is activated immediately when received by the Control Unit. The influence on the population determines delays, depending on when future journey requests are received and how they are carried out by individual travellers.

6.4.3 Pattern Adaptation Results. Figure 7 shows the application of the influence pattern (7) sent by the Control Unit to the transportation network entity and influencing the population. This causes the network to change the operational mode of the journey planner.

Adaptation starts at the beginning of the simulations. During the early days of an epidemic, travellers are able to mostly avoid infectious areas of the network; thus the *safe* mode succeeds in reducing the number of daily infections. However, as the epidemic continues, there are fewer risk-free locations in the network, making avoidance difficult or impossible. The *safe* and *short* modes therefore result in roughly the same number of infections past approximately 40 days into the epidemic. During our experiments, we also observe that while the *safe* mode moved travellers to safer parts of the network, it also contributes to the spread of the disease, since infected travellers would also follow the safe travel plans.

6.5 Pseudo-Emergence

The *pseudo-emergence* pattern creates and deploys a pseudo-entity to apply control. In our system, we consider the deployment of test station pseudo-entities to apply medical testing to travellers and activate quarantine in case of infection cases.

6.5.1 Pattern Syntax. The syntax of the pseudo-emergence pattern is of the form $pseudoEmergence : Entity \rightarrow Entity$ which takes as input a new pseudo entity $e \in Entity$ to create, and a target entity $ta \in Entity$ such that the pseudo-emergence pattern is the term $pseudoEmergence(ta, e)$ invoked by the Control Unit, which creates the new pseudo entity e in the system to send information to the target entity ta.

6.5.2 Pattern Semantics. The pseudo-emergence pattern creates a new entity which can directly send information to the target entity. The semantics of the pseudo-emergence pattern, is defined as $[\![pseudoEmergence(ta, e)]\!]$, i.e., a new entity σ_e is created, and there exists a direct interaction $send_i : Entity \times Entity \times P \rightarrow Entity$ such that the equation $\sigma'_{ta} = send_i(\sigma_e, \sigma_{ta}, v)$ means e sends the parameter value v to set in the target entity ta, i.e., $\sigma'_{ta}(v) = p_i$ is set using the corresponding set operation.

The term $pseudoEmergence(\sigma_T, \sigma_{ts})$ specifies the pseudo - emergence pattern such that it creates the test station pseudo-entity ts, defined with parameters $\sigma_{ts} = (l)$, where l is a location of G. Entity σ_{ts} is created in the PT system by setting the transportation network entity parameter $test_l := true$. The test station is specified by the predicate $test : T \rightarrow \mathbb{B}$ such that $test(t) = true$ if traveller $t \in T$ is infected with the disease, and $false$ otherwise. In symbols, $\sigma'_t = send_s(\sigma_{ts}, \sigma_t, s)$ sets $\sigma'_t(s) = Q$ if, and only if, $test(t) = true$. Otherwise, the traveller t's infection state s is left unchanged.

6.5.3 Pattern Adaptation Results. Figure 8 shows the application of the pseudo-emergence pattern 8 sent by the Control Unit to the transportation network entity and placing two test stations (e.g. due to budget limitations). The data series shows the result of triggering the pattern after 10, 5, and 1 infection cases on the network. Once the pattern is triggered, the Control Unit creates a test station pseudo-entity at a vertex with the highest exposure risk. The results of the graph show the importance of adaptation

Network #vertices	Population #travellers	MDP #states	MDP #trans	Sim. time (minutes)
7	100	10	23	0.21
	300			0.35
	500			0.37
16	100	19	53	0.79
	300			1.70
	500			1.83
36	100	39	124	4.22
	300			14.79
	500			18.66
64	100	67	230	22.70
	300			65.86
	500			88.83

Table 2: Scalability of the Feedback Loop Workflow

occurring early in the epidemic. When the only infection case is tested and quarantined, the epidemic stops. If adaptation is activated after 5 cases, the epidemic has spread sufficiently to maintain an epidemic throughout the simulation. A larger number of infections is shown when adapting after 10 cases.

6.6 Scalability of the Feedback Loop

To assess the scalability of our technique, we run 100-day simulations with different transportation network and population sizes. We use a standard MacBook Pro machine with a 2.3 GHz Quad-Core Intel Core i7 Processor and 16GB memory, and we collect the size of the MDP model (i.e., the number of states and transitions), and the simulation time, (i.e., the time, expressed in minutes), our approach takes to analyse the considered scenario. Table 2 summarises our results. The simulation time increases when considering larger network and population sizes. We do get analysis results in less than 2 minutes when 500 travellers move in a network of 16 vertices which seems to us a quite realistic scenario.

The canonical values in Equation (5) we set for the categories of risk impact the running time of the simulation. This is due to simulation of the Discrete-Time Markov Chain obtained from resolving the non-determinism of the travel network's MDP, to compute a strategy. If the probability transitions to reach a success state are low, then more time is needed to obtain the strategy. Our experimental results show that the values used for `RED`, `YELLOW` and `GREEN` in a synthesised Prism Module (2) a strategy can take two to three seconds per traveller to obtain in networks with 64 vertices. We cache the safest paths to be reused, but these must be reset after each new health recommendation.

Lastly, CTMC model checking involving the SEIR compartment model may be subject to the state explosion problem [21]. However, this is mostly mitigated by performing SEIR model checking per vertex, rather than across the entire population. We skip CTMC model checking on vertices with no infectious travellers.

7 THREATS TO VALIDITY

Besides inheriting all limitations of design patterns and probabilistic model checking research [18, 29], our approach exhibits the following threats to validity [27]. *External threats.* We are aware that the generalisation of results is not guaranteed since our formal modelling and analysis techniques are applied to five design patterns, and one case study has experimented. We intend to pursue the adoption of a larger number of patterns to further systems as part of our future research. *Internal threats.* Our work builds upon our interpretation of design patterns defined in [14] and there might be some biases. Moreover, the design of experiments might be critical, but we used numerical values to have a direct manipulation of the experimental results of interest. To smooth this last point, it is worth pointing out that models are publicly available [13] and software engineers can easily change the numerical values of input parameters and run further analyses that have been not considered. *Construct validity.* The statistical validity of the experimental results is smoothed by running a probabilistic model checking analysis with a well-assessed tool (PRISM [16]), thus monitoring the accuracy of presented numerical results.

8 RELATED WORK

Within the adaptive and self-managing research communities, we found the following contributions [6, 8, 28]. Fang et al. [8] predict the violation of system-level disruptions and aim to support proactive adaptation. We share the probabilistic model checking analysis, but our work shows the different goal of triggering different patterns of applied control inherited from the literature [14], thus preventing the spread of epidemics. Barkowsky et al. [6] present an approach to improve the computational effort when monitoring systems through incremental queries that minimize the information to extract from sensing the system's evolution. We plan to integrate this approach to improve the efficiency of our methodology when monitoring our transportation network. Besides, we are also interested in pursuing the idea of adaptive analysis which means balancing the effort of the analysis w.r.t. the actual impact of the solution, e.g., labeling travellers if this indeed prevents the spread of epidemics. Zeller et al. [28] consider a self-protecting layer to protect systems against attacks targeting business logic rules. We plan to make use of this approach to detect fake information circulating in our public transportation system, thus preventing an analysis that relies on manipulated (or outdated) data. When considering broader research communities, we select the following methodologies [20, 24, 25] as the most related ones. Silva et al. [25] make use of the SIR epidemic model and investigate the adaptive behaviour of the population that may not follow preventive instructions, given that actions are perceived as too restrictive. Similarly, our patterns imply more/less invasive adaptation actions (e.g., the constraint pattern limits the access to a path generating delays in commuting, whereas the isolation pattern makes a connection unaccessible) to decide how to prevent risky scenarios. Marchetti et al. [20] also evaluate the trade-off between health outcomes and the restrictiveness of mitigation strategies. Interestingly, the analysis in [20] confirms that it is indeed valuable to act at early signs of a surging wave and severe restrictions might be necessary. Further parameters can be considered to detail the infection process, e.g., vaccination coverage, and we plan to further play with the parameters of our model to include more refined conditions. Qian et al. [24] focus on the SEIR model and additionally consider the travel contagion distinguishing different types of transportation systems, e.g., private vehicles and metro stations. This represents an

interesting direction for our model where the contagion parameter can be modified based on such information. About adaptation, the methodology in [24] includes a procedure of entrance screening, i.e., identifying travellers with abnormal body temperature or presenting relevant symptoms. Our approach is instead agnostic of the health status of each traveller, the decision on adaptation actions relies on the status of the travel network only, i.e., the probability of having a certain number of infection cases. Summarising, to the best of our knowledge, there exist complementary research directions from which our paper may benefit, our work is novel in adopting patterns of applied control to counteract epidemic flows in the domain of public transportation systems.

9 CONCLUSION

This paper investigates the adoption of patterns of applied control defined in the self-adaptive community, and presents a novel modelling and formal verification approach. Our results support public authorities in making more informed healthcare decisions during outbreaks and epidemics. We quantitatively compare different adaptation strategies for a public transportation system, and we provide formal guarantees on the number of infections.

There are several directions in which to take this work. First, we plan to consider the travellers' satisfaction as an additional goal for our system and solve a multi-objective problem that balances infection prevention and people's willingness. Next, we plan to develop a CAT pattern-based language to express and simulate more complex adaptation scenarios. Lastly, we aim to develop a user-friendly visualisation tool to enable healthcare domain experts to simulate a wider range of scenarios.

10 ACKNOWLEDGEMENTS

This work has been partially funded by the MUR-PRIN project DREAM (20228*FT*78*M*), the MUR-PRO3 project on Software Quality, and the MUR-PNRR project VITALITY (*ECS*00000041).

REFERENCES

[1] Daniel Miravet Aaron Gutiérrez and Antoni Domènech. 2021. COVID-19 and urban public transport services: emerging challenges and research agenda. *Cities & Health* 5, sup1 (2021), S177–S180. https://doi.org/10.1080/23748834.2020.1804291

[2] Joan L. Aron and Ira B. Schwartz. 1984. Seasonality and period-doubling bifurcations in an epidemic model. *Journal of Theoretical Biology* 110, 4 (1984), 665–679. https://doi.org/10.1016/S0022-5193(84)80150-2

[3] Imen Ayouni, Jihen Maatoug, Wafa Dhouib, Nawel Zammit, Sihem Ben Fredj, Rim Ghammam, and Hassen Ghannem. 2021. Effective public health measures to mitigate the spread of COVID-19: a systematic review. *BMC Public Health* 21, 1 (29 May 2021), 1015. https://doi.org/10.1186/s12889-021-11111-1

[4] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. 2000. Model-Checking Continuous-Time Markov Chains. *ACM Trans. Comput. Logic* 1, 1 (jul 2000), 162–170. https://doi.org/10.1145/343369.343402

[5] Carlos Balsa, Isabel Lopes, Teresa Guarda, and José Rufino. 2021. Computational simulation of the COVID-19 epidemic with the SEIR stochastic model. *Computational and Mathematical Organization Theory* (30 Mar 2021). https://doi.org/10.1007/s10588-021-09327-y

[6] Matthias Barkowsky, Thomas Brand, and Holger Giese. 2021. Improving Adaptive Monitoring with Incremental Runtime Model Queries. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 71–77.

[7] Lama Bou-Karroum and et al. 2021. Public health effects of travel-related policies on the COVID-19 pandemic: A mixed-methods systematic review. *Journal of Infection* 83, 4 (2021), 413–423. https://doi.org/10.1016/j.jinf.2021.07.017

[8] Xinwei Fang, Radu Calinescu, Colin Paterson, and Julie Wilson. 2022. PRESTO: Predicting System-level Disruptions through Parametric Model Checking. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 91–97.

[9] H Hansson and B Jonsson. 1994. A logic for reasoning about time and reliability. *Formal Aspects of Computing* 6, 5 (01 Sep 1994), 512–535. https://doi.org/10.1007/BF01211866

[10] Hans Heesterbeek, Roy M Anderson, Viggo Andreasen, Shweta Bansal, Daniela De Angelis, Chris Dye, Ken TD Eames, W John Edmunds, Simon DW Frost, Sebastian Funk, et al. 2015. Modeling infectious disease dynamics in the complex landscape of global health. *Science* 347, 6227 (2015), aaa4339.

[11] T. D. Hollingsworth, N. M. Ferguson, and R. M. Anderson. 2007. Frequent travelers and rate of spread of epidemics. *Emerging Infectious Diseases* 13, 9 (Sep 2007), 1288–1294. https://doi.org/10.3201/eid1309.070081

[12] Leonhard Horstmeyer, Christian Kuehn, and Stefan Thurner. 2022. Balancing Quarantine and Self-Distancing Measures in Adaptive Epidemic Networks. *Bulletin of Mathematical Biology* 84, 8 (2022), 79. https://doi.org/10.1007/s11538-022-01033-3

[13] Kenneth Johnson, Samaneh Madanian, and Catia Trubiani. 2024. Patterns of Applied Control for Public Health Measures on Transportation Services under Epidemic – Replication Data. https://zenodo.org/doi/10.5281/zenodo.10609044.

[14] Christian Kroher, Lea Gerling, and Klaus Schmid. 2023. Control Action Types - Patterns of Applied Control for Self-adaptive Systems. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*.

[15] Christian Kröher, Lea Gerling, and Klaus Schmid. 2023. Control Action Types-Patterns of Applied Control for Self-adaptive Systems. In *IEEE/ACM Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 32–43.

[16] Marta Kwiatkowska, Gethin Norman, and David Parker. 2005. Probabilistic model checking in practice: Case studies with PRISM. *ACM SIGMETRICS Performance Evaluation Review* 32, 4 (2005), 16–21.

[17] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11) (LNCS, Vol. 6806)*, G. Gopalakrishnan and S. Qadeer (Eds.). Springer, 585–591.

[18] Marta Kwiatkowska, Gethin Norman, and David Parker. 2018. Probabilistic model checking: Advances and applications. *Formal System Verification: State-of the-Art and Future Trends* (2018), 73–121.

[19] Nita Madhav, Ben Oppenheim, Mark Gallivan, Prime Mulembakani, Edward Rubin, and Nathan Wolfe. 2018. Pandemics: risks, impacts, and mitigation. (2018).

[20] Sabina Marchetti, Alessandro Borin, Francesco Paolo Conteduca, Giuseppe Ilardi, Giorgio Guzzetta, Piero Poletti, Patrizio Pezzotti, Antonino Bella, Paola Stefanelli, Flavia Riccardo, Stefano Merler, Andrea Brandolini, and Silvio Brusaferro. 2022. An epidemic model for SARS-CoV-2 with self-adaptive containment measures. *PLOS ONE* 17, 7 (07 2022), 1–18. https://doi.org/10.1371/journal.pone.0272009

[21] Paolo Milazzo. 2021. Analysis of COVID-19 Data with PRISM: Parameter Estimation and SIR Modelling. In *From Data to Models and Back*. Springer International Publishing, Cham, 123–133.

[22] Seyed M. Moghadas, Nick J. Pizzi, Jianhong Wu, and Ping Yan. 2009. Managing public health crises: the role of models in pandemic preparedness. *Influenza and Other Respiratory Viruses* 3, 2 (2009), 75–79. https://doi.org/10.1111/j.1750-2659.2009.00081.x

[23] Mohammad Nazayer, Samaneh Madanian, Hamidreza Rasouli Panah, and Dave Parry. 2023. Public Health Emergencies: Health Data and Public Health Surveillance. In *2023 IEEE International Conference on Digital Health (ICDH)*. 295–297. https://doi.org/10.1109/ICDH60066.2023.00050

[24] Xinwu Qian and Satish V. Ukkusuri. 2021. Connecting urban transportation systems with the spread of infectious diseases: A Trans-SEIR modeling approach. *Transportation Research Part B: Methodological* 145 (2021), 185–211. https://doi.org/10.1016/j.trb.2021.01.008

[25] Diogo H. Silva, Celia Anteneodo, and Silvio C. Ferreira. 2023. Epidemic outbreaks with adaptive prevention on complex networks. *Communications in Nonlinear Science and Numerical Simulation* 116 (2023), 106877. https://doi.org/10.1016/j.cnsns.2022.106877

[26] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M Göschka. 2013. On patterns for decentralized control in self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, Revised Selected and Invited Papers*. Springer, 76–107.

[27] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, and Björn Regnell. 2012. *Experimentation in Software Engineering*. Springer.

[28] Silvan Zeller, Narges Khakpour, Danny Weyns, and Daniel Deogun. 2020. Self-protection against business logic vulnerabilities. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 174–180.

[29] Cheng Zhang and David Budgen. 2012. What Do We Know about the Effectiveness of Software Design Patterns? *IEEE Transactions on Software Engineering* 38, 5 (2012), 1213–1231.