Informatik
Hauptcampus

HOCHSCHULE TRIER

Simulation

Simulation and Reinforcement Learning

Frederic Nicolas Schneider

Simulation and Reinforcement Learning

Betreuer: Prof. Dr. Christoph Lürig

Trier, 22.05.2023

Inhaltsverzeichnis

1

Identität

Im Rahmen der Vorlesung “Simulation and Reinforcement Learning” sollen drei Fahrstühle eines Bürogebäudes simuliert und anschließend ihre Funktion mittels Reinforcement Learning optimiert werden.

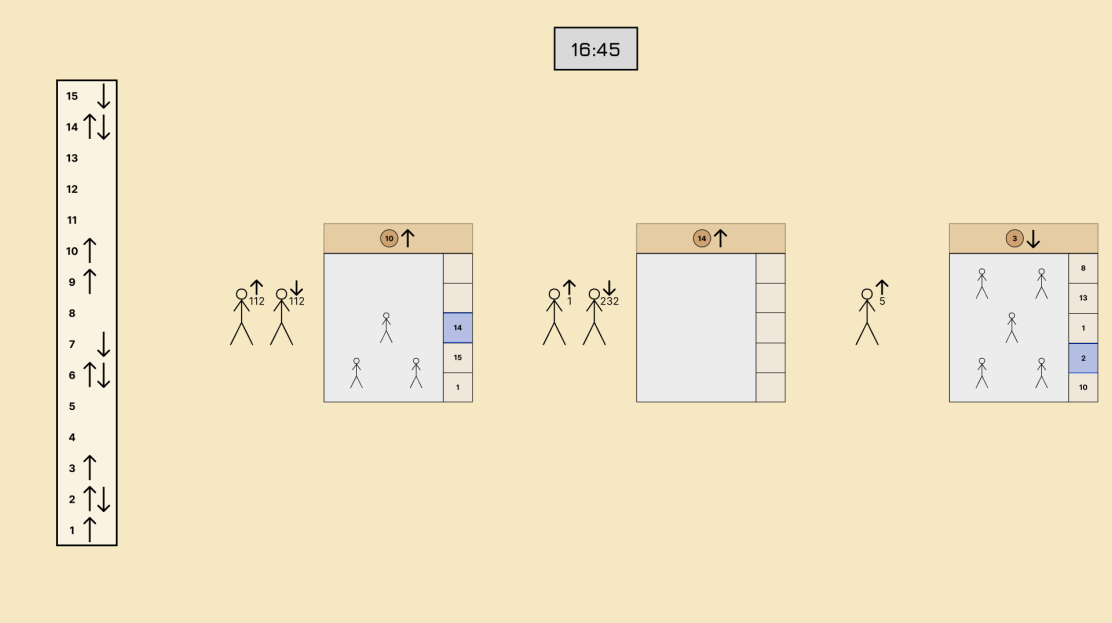

Abbildung 1.1: UI der späteren Fahrstuhl-Simulation

Hierfür wird die Simulation auf einen groben Detailgrad heruntergebrochen. Die Grenzen der Simulation werden in den nachfolgenden Kapiteln genauer beleuchtet.

2

Eigenschaften

2.1 Innere Struktur des Fahrstuhls

Der Fahrstuhl ist in 3 Elementen aufgeteilt.

- Stockwerksposition und Fahrtrichtung
- Personenanzahl für nach oben und unten des aktuellen Stockwerks
- Innenraum

Der Innenraum zeigt, wie viele Personen sich in ihm befinden und wo welche Knöpfe in welcher Reihenfolge gedrückt wurden. Des Weiteren wird sein Fahrziel hervorgehoben angezeigt.

Es gibt drei Zustände je Fahrstuhl:

- Warten
- Hoch
- Runter

Warten fasst zwei Zustände zusammen. Das Warten, bis ein Rufknopf gedrückt wurde und bis der Ein- und Aussteigevorgang abgeschlossen ist.

2.2 Äußere Parameter.

Die äußeren Parameter, die sich auf das Modell auswirken, sind die zu befördernden Personen. Die Fahrstühle kennen lediglich die Anzahl der Personen in ihrem Inneren und in welchem Stockwerk eine unbekannte Anzahl an Personen nach oben und / oder nach unten möchten. Dabei gilt ein exponentielles Wachstum.

2.3 Ein- und Ausgabeparameter

Um das Modell möglichst flexibel zu gestalten, wird eine Bandbreite von Ein- und Ausgabeparametern unterstützt. Diese können wiederum in Fahrstuhl, Personen und Haus unterteilt werden.

2.3.1 Eingabeparametern:

Zu den Eingabeparametern der Fahrstühle gehören:

- Kapazität der Fahrstühle
- Geschwindigkeit der Etagenwechsel und Dauer des Ein- und Aussteigevorgangs (in Takten)

Zu den Eingabeparametern des Hauses gehören:

- Anzahl an Etagen
- Liste von Etagen und Zeiten von Spitzenaufkommen (Bspw. Mittagspause).

Personen steuern im Gegensatz nur das maximale Tagesaufkommen zu den Eingaben bei.

2.3.2 Ausgabeparametern:

Die Ausgabeparameter beschränken sich außerhalb des Logs lediglich auf die Aussagen, ob alle Personen bis zum Ende des Tages das Gebäude verlassen haben und welche durchschnittliche Wartezeit vorliegt.

3

Verhalten

3.1 Vorbedingungen

Als Vorbedingung der Simulation gelten die Eingangsparameter, die den Fahrstühlen ihre Eigenschaften beschreiben. Mit den Eigenschaften repräsentieren die Fahrstühle die Funktionen der Transportation der Personen auf verschiedenen Stockwerken. Dabei agieren sie nicht aktiv mit den Personen.

3.2 Interne Prozesse

Die Fahrstühle folgen während der Simulation folgendem Prozessablauf.

Zum Beginn wartet ein Fahrstuhl. Hier können zwei Events eintreten. Eine Person möchte auf dem aktuellen Stockwerk Einsteigen oder der Fahrstuhl wird von einem anderen Stockwerk aus gerufen. Im zweiten Fall fährt der Fahrstuhl zum neuen Stockwerk und wartet eine bestimmte Taktzahl (standardmäßig ein Takt) ab. In dieser Zeit wird der Einstiegsvorgang simuliert, der auch im ersten Fall vorliegt. Dabei erhöht sich die Belegung um eins und das gewünschte Stockwerk wird in die Aufgabenliste eingereiht. Nun entscheidet sich zunächst durch den Aufzug-Algorithmus (oder auch SCAN) entschieden, welches Stockwerk angefahren werden soll. Dabei gilt die Zielstockwerke der Insassen und der Personen, die auf die Stockwerke verteilt den Aufzug rufen. Jedoch kennt der Fahrstuhl nur die Richtung, in die die Personen außerhalb fahren möchten und nicht deren Zielstockwerk.

Das Verhalten einer Person beginnt, sobald diese vom Personen-Manager in das Gebäude geschickt wird und einen Aufzug für die gewünschte Zielrichtung ruft. Sobald ein Aufzug auf der Etage ankommt. Hat dieser noch Platz für die Person, steigt diese ein und drückt auf die Zieletage. Ansonsten wartet sie, bis der nächste Aufzug ankommt. Wurde die Zieletage erreicht, soll die Person aussteigen und ansonsten warten. Ist es Zeit für die Mittagspause oder der Tag zu Ende und befindet sich die Person nicht im Erdgeschoss, soll sie wieder den Aufzug in die gewünschte Richtung rufen. Am Arbeitstagende geht die Person nach Hause, wenn sie im Erdgeschoss ist.

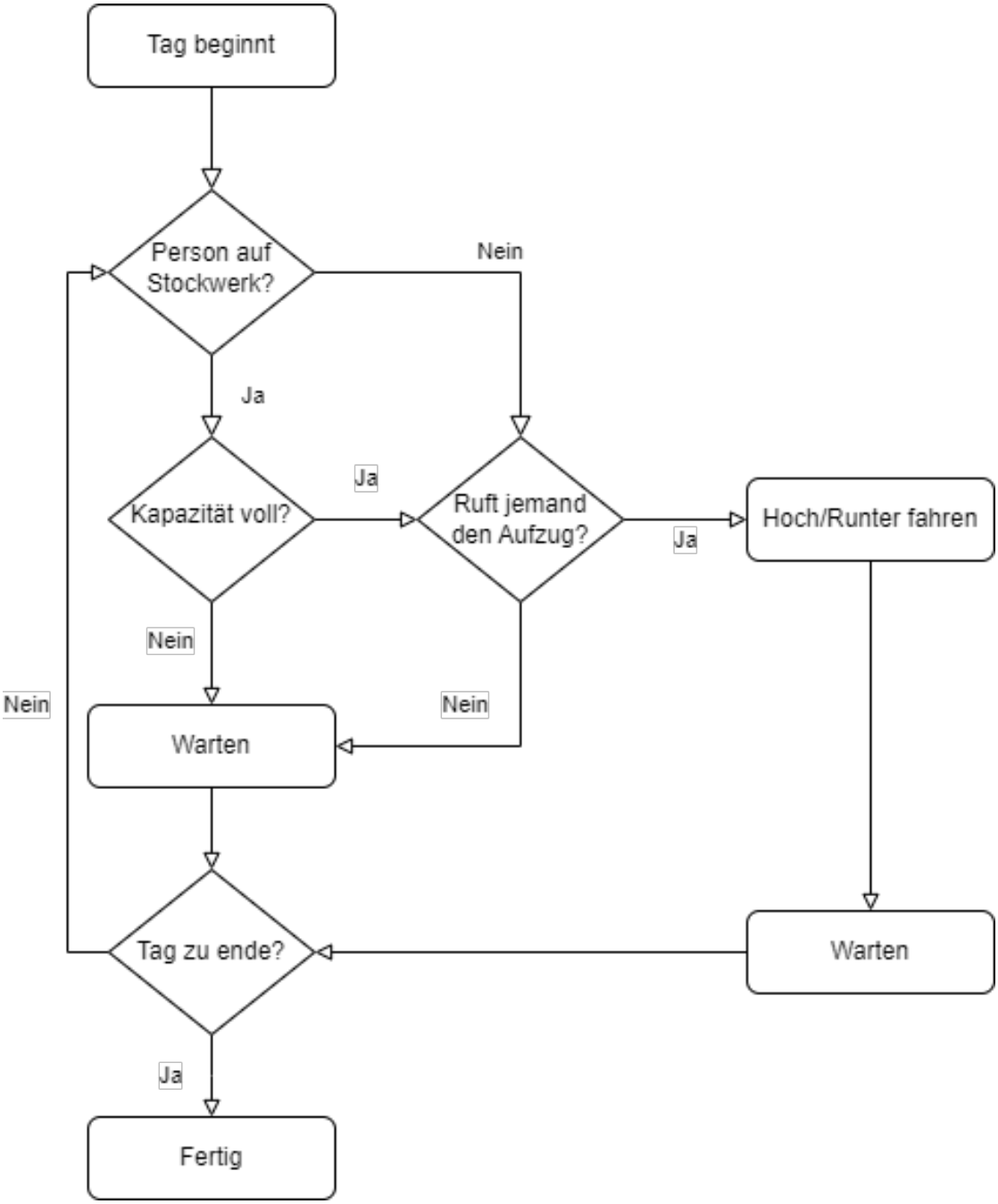

Abbildung 3.1: Prozessablauf eines Fahrstuhls

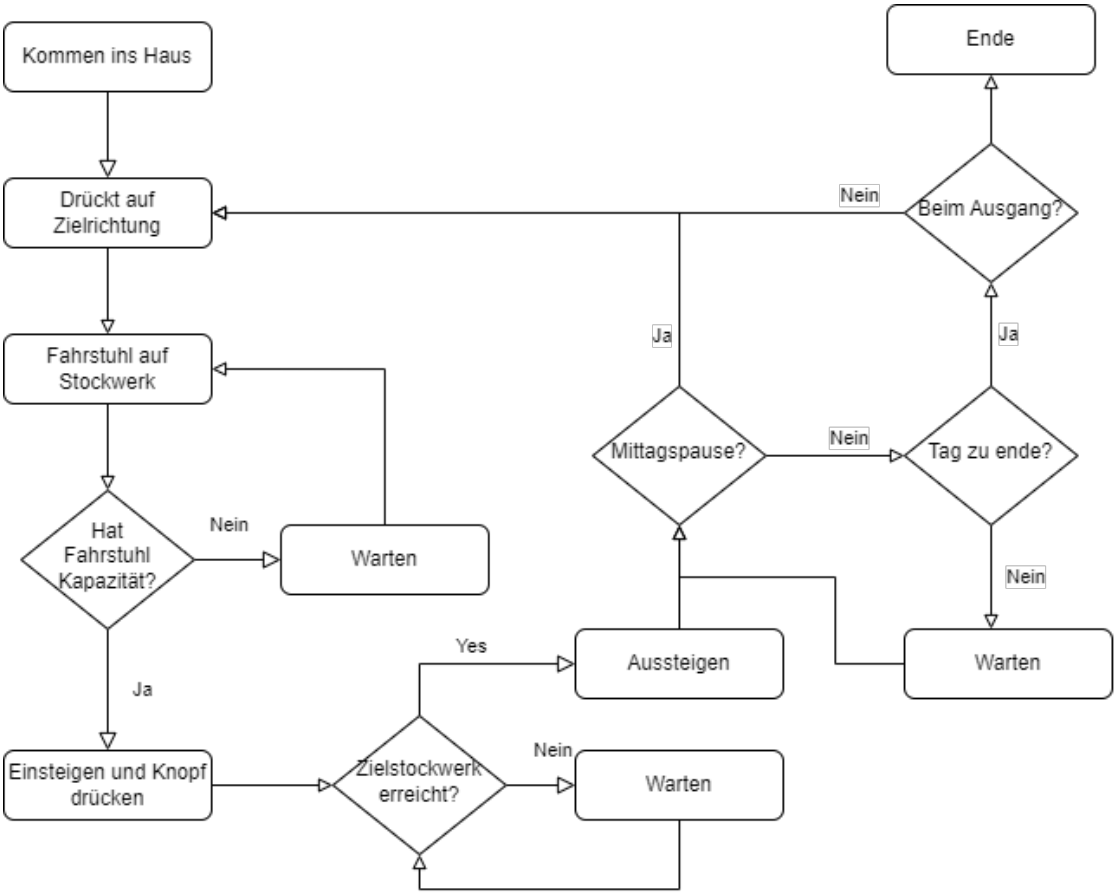

Abbildung 3.2: Prozessablauf einer Person

3.3 Fehlermöglichkeit

Die Situation, dass sich am Ende des Tages noch weitere Personen im Haus befinden, kann als Fehler der Simulation auftauchen. Dies geschieht, wenn die Fahrstühle zu wenige Personen bedienen konnten. Jedoch lässt sich dies auch auf ein zu hohes Personenaufkommen zurückzuführen. Diese Situation tritt vor allem dann auf, wenn selbst das optimale Verhalten der Fahrstühle nicht ausreicht, um sämtliche Personen zu transportieren.

Daher wird sich für die spätere Optimierung mittels Reinforcement Learning auf den Personenandrang zurückgegriffen, die bereits der Aufzug-Algorithmus bewältigen konnte.

3.4 Nachbedingung

Nach dem Abschluss der Simulation muss als Nachbedingung ein leeres Gebäude folgen. Keine Person sollte sich noch darin aufhalten.

4

Verifikation und Validierung

Um die Simulation abschließend bewerten zur können, benötigt es Dokumente und Metriken zur Verifikation und Validierung des Programmes.

4.1 Verifikation

Auf die Frage, ob die Simulation richtig implementiert wurde, dient dieses Conceptual Model als Anforderungen an die Simulation.

4.2 Validierung

Als Methoden der Validierung werden Behavior Validation und Replicative Validation angewandt.

Als Behavior Validation wird auf veränderte Parameter ein plausibles Verhalten erwartet. Im Falle der Fahrstühle wird bei einem erhöhten Personenaufkommen auch eine höhere durchschnittliche Wartezeit erwartet und vice versa.

Bei der Replicative Validation werden mittels Experimente die Ergebnisse verglichen. So soll bei gleichen Parametern trotz Zufallselementen, wie die Zielstockwerke der Personen, das gleiche oder ein ähnliches Ergebnis erzielt werden.

5

Simulationsübersicht

Die Simulation wurde in Python realisiert und anhand des Conceptual Model gestaltet. In den nächsten Kapiteln wird der implementierte Ablauf der Logik der Simulation erläutert.

Die Klasse Simulation selbst initialisiert Personen- und Aufzugsmanagement und führt in einer „run“-Methode diese taktweise aus. Dabei wird am Ende eines Taktes die generierten Logs zusammengeführt und in einer CSV-Datei gespeichert.

Am Ende des Tages wird eine Statistik erstellt, welche in eine CSV-Datei gespeichert wird, in der jede Zeile ein simulierter Tag entspricht.

Durch das intensive Logging soll im späteren Teil des Reinforcement Learnings das Vergleichen der verschiedenen Generationen erleichtern.

5.1 Validierung der Simulation

Die Validierung der Simulation erfolgt anhand der UI, die die Zustände und Daten der Simulation visualisieren. Dabei kann diese beschleunigt und pausiert werden. Diese ist jedoch nur dann aktiviert, wenn für das Programm ein Flag gesetzt wurde. Dadurch kann der Reinforcement Learner später einen simulierten Tag verarbeiten, ohne verlangsamt zu werden.

Als zweite Methode helfen die Log-Dateien, in denen jede Zeile einen Takt der Simulation repräsentieren.

Des Weiteren werden einige Daten geplottet, unter anderem die durchschnittliche Wartezeit. Wenn diese steigt, sobald viele Menschen unterwegs sind und sinkt, wenn dies nicht der Fall ist, so verhält die Simulation wie erwartet und bringt ein Indiz, dass diese Valide ist (5.2).

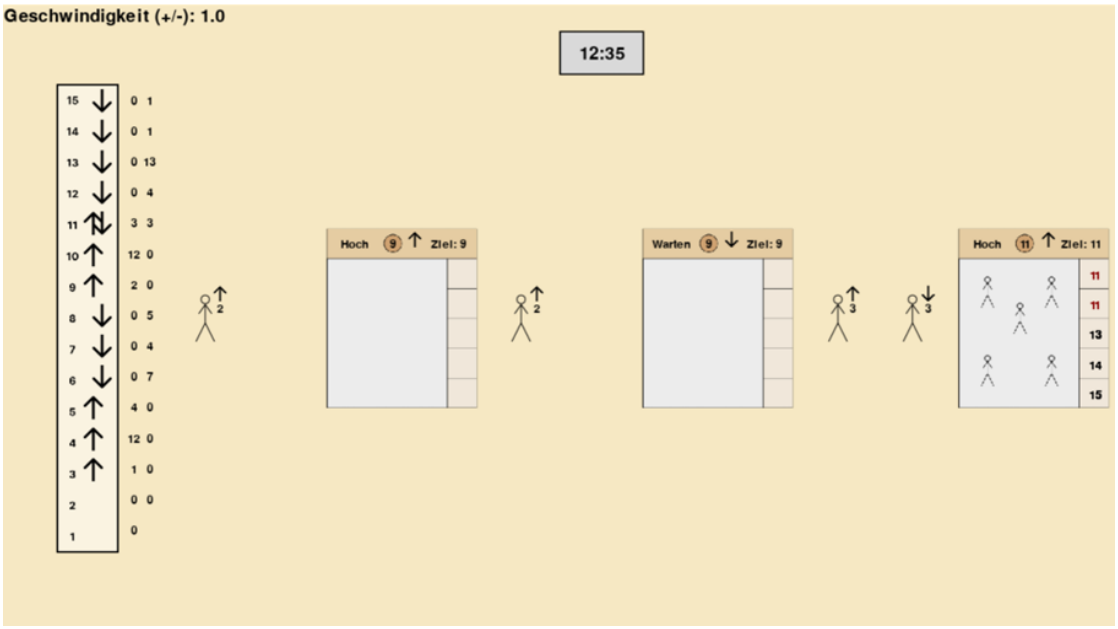

Abbildung 5.1: UI der Simulation zur Validierung

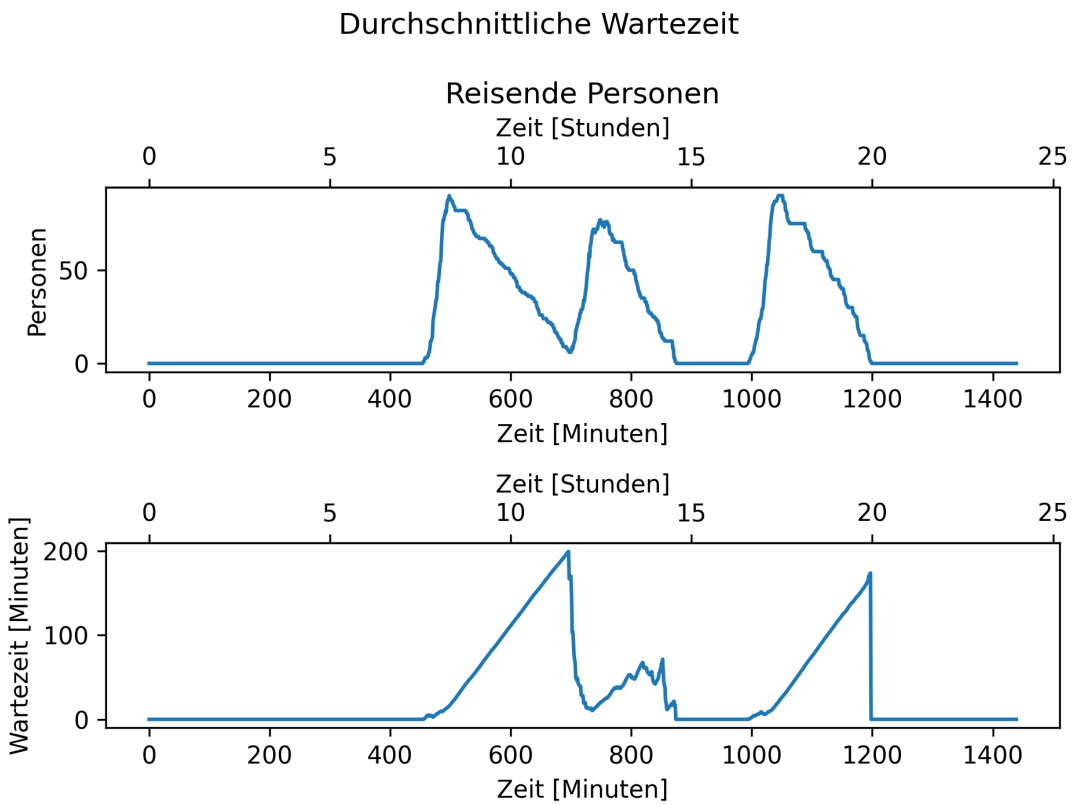

Abbildung 5.2: Anzahl reisende Personen vs. durchschnittliche Wartezeit zum Zeitpunkt t

6

Personenmanagement Logik

Die Personen, die im Gebäude arbeiten werden als einzelne Klasse modelliert und speichern deren aktuellen Zustand, wie Position, Heimat-Stockwerk und Schedule, wann sie wohin möchten.

Die Klasse „PersonManager“ generiert bei der Initialisierung sämtliche Büroarbeiter und verteilt deren Heimat-Stockwerke gleichverteilt über die Etage außer dem Erdgeschoss (6.1).

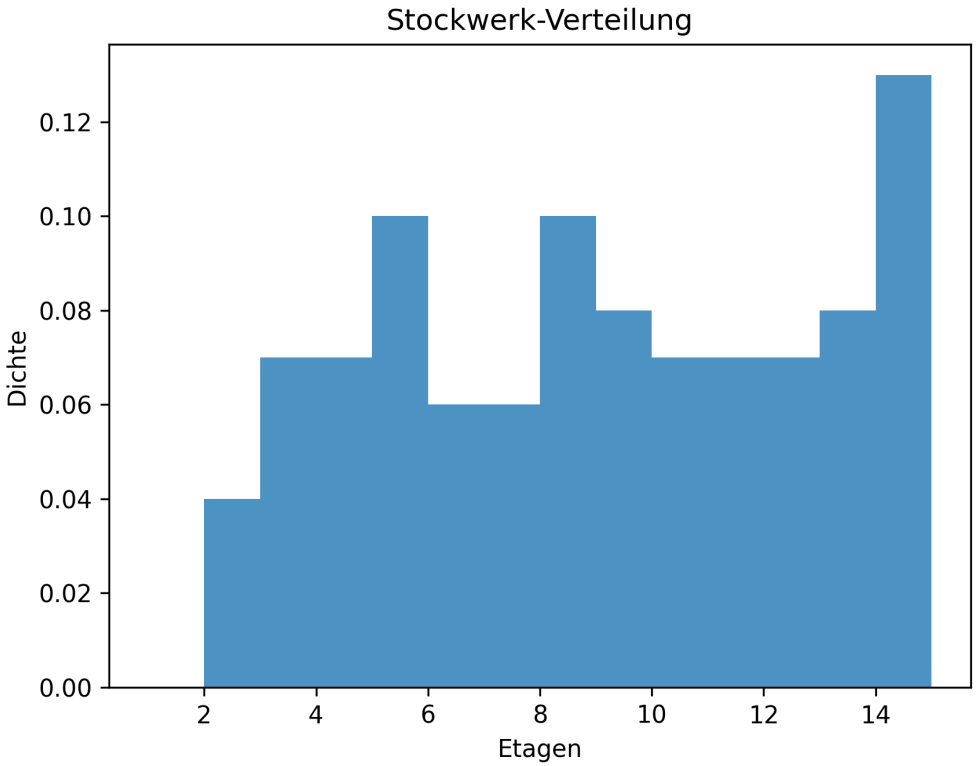

Abbildung 6.1: Etagen Verteilung

Anschließend wird für jede Person einen Schedule aus Tupeln von wann und wohin sie müssen generiert. Das Wann wird hierbei gammaverteilt festgelegt, um einen natürlichen Andrang anzunähern (6.2).

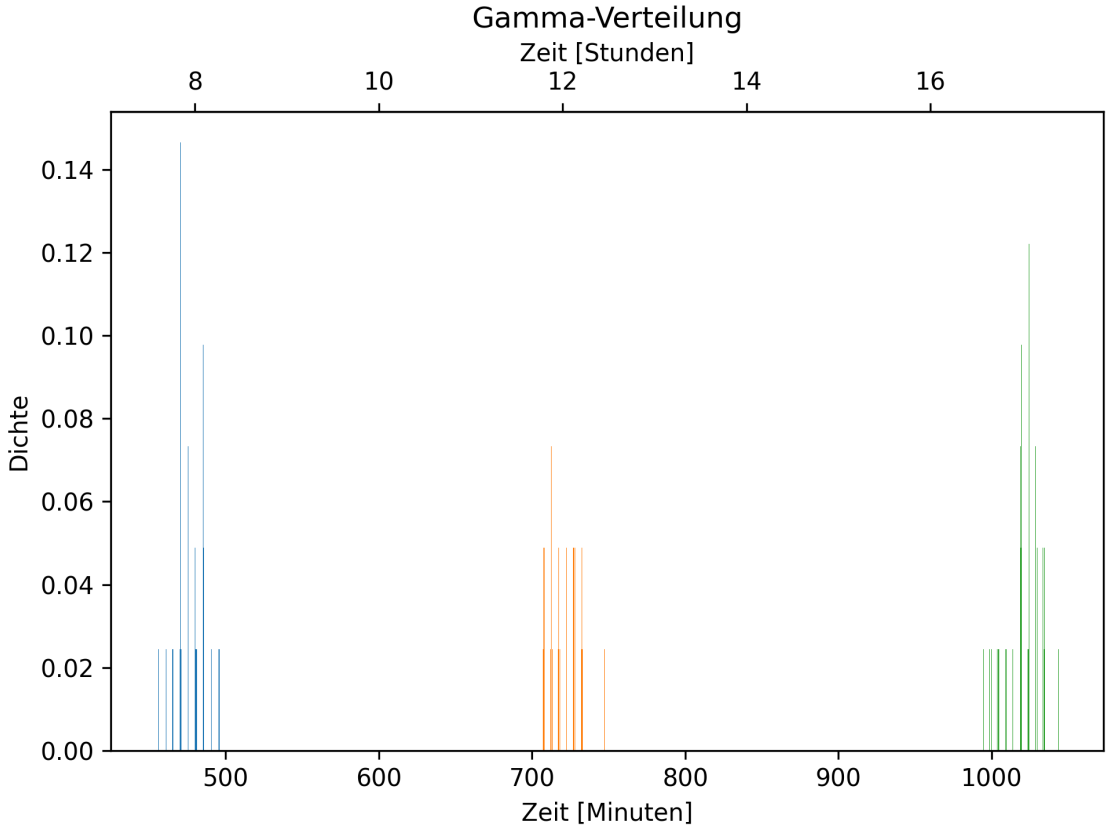

Abbildung 6.2: Verteilung wann Person die Etage wechseln möchte

Nach der Generierung wird in jedem Takt die „manage“-Methode aufgerufen. Diese iteriert über alle Personen, die sich noch im Gebäude befinden. Ist die aktuelle Person bereits unterwegs, geht es zur nächsten Iteration. Ist die Person nicht unterwegs aber deren Scheduler leer, dann wird diese Person aus der Liste der Personen entfernt, die noch im Gebäude sind. Denn der letzte Punkt eines Schedules ist immer der Weg nach Hause. Ist dieser abgeschlossen und aus dem Schedule entfernt, bedeutet es, dass diese Person nicht mehr im Gebäude befindet.

Gibt es noch Elemente im Schedule, wird geprüft, ob es Zeit ist, sich zur Zieletage des nächsten Elements im Schedule zu reisen. Ist dies nicht der Fall, dann wird weiter iteriert. Ansonsten stellt sich die Frage, ob die Person nach oben oder unten möchte. Abhängig davon, wird sie in die entsprechende Liste eingetragen, die als eine Liste je Stockwerk für die jeweilige Richtung realisiert wurde. Möchte die Person auf das gleiche Stockwerk, in der sie sich bereits befindet, so wird dieses Element des Schedules entfernt und dadurch als erfüllt angesehen.

Am Ende der Iteration wird der aktuelle Stand des Personenmanagements geloggt.

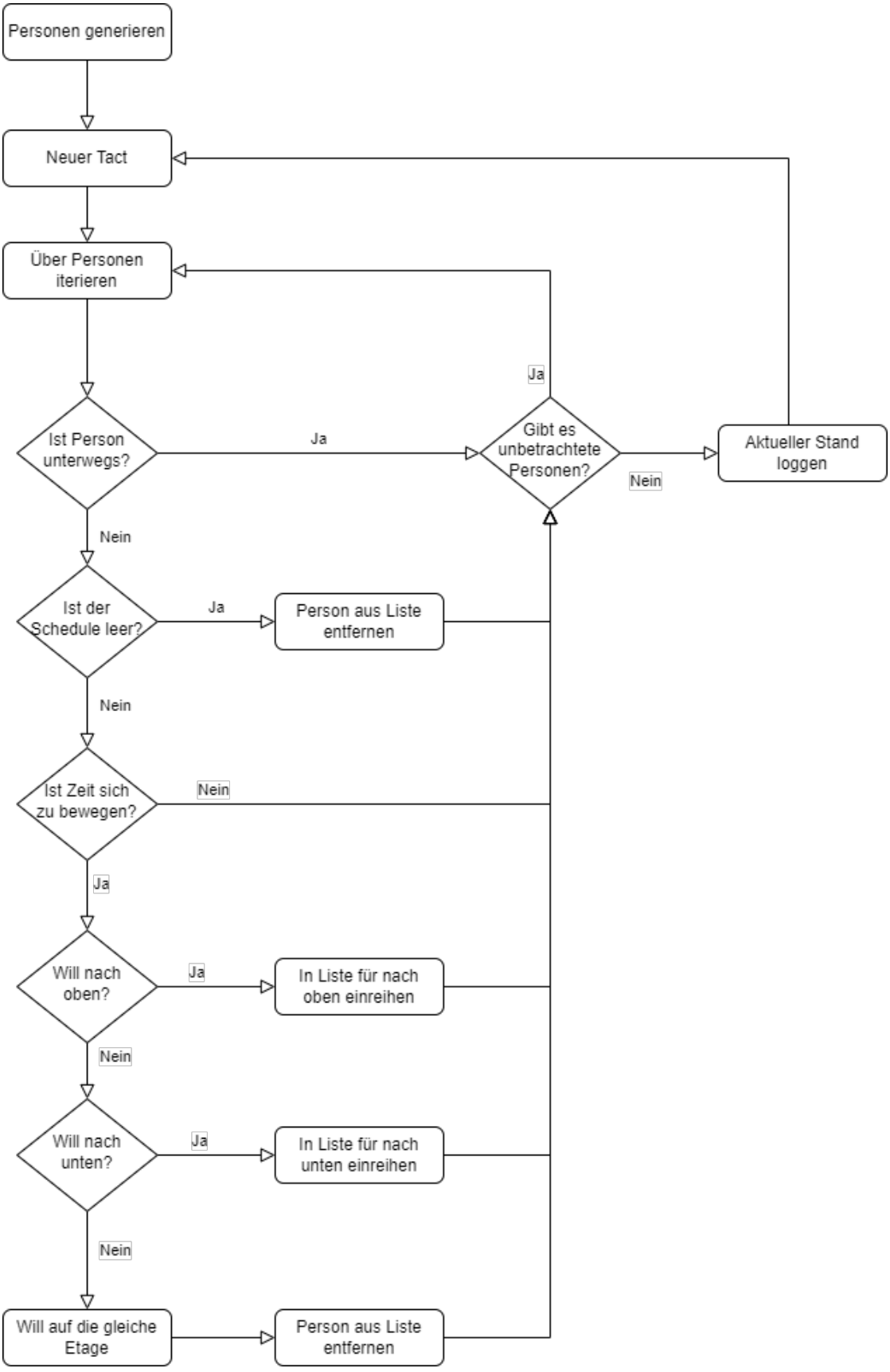

Abbildung 6.3: Prozessablauf einer Person

7

Aufzugsmanagement Logik

Die einzelnen Aufzüge entscheiden, wann sie wohin möchten, und kommunizieren nicht untereinander. Bei jedem neuen Takt wird geprüft, ob das aktuelle Ziel-Stockwerk erreicht wurde. Ist dies der Fall, soll der Bewegungszustand des Aufzugs „Warten“ sein. Ansonsten wird dieser darauf gesetzt, der aktuelle Stand geloggt und für diesen Takt beendet. Wartet der Aufzug bereits, lässt er Passagiere, die im aktuellen Stockwerk aussteigen wollen, aussteigen und die Personen einsteigen, die in die gleiche Richtung, wie der Aufzug möchten. Als Nächstes wird die neue Ziel-Etage aus den Zielen der Passagiere ermittelt.

Ist der Aufzug leer und ohne Ziel, zu dem er gerufen wurde, dann scannt dieser nach einer Aufforderung. Dabei wird zunächst geschaut, ob in der aktuellen Richtung bis zum Rand des Gebäudes (Erdgeschoss oder oberste Etage) gerufen wurde, und übernimmt die Etage, die am nächsten zur aktuellen Position liegt. Ansonsten wird in der Richtung nach der weit entfernten Forderung in die Gegengesetze Richtung gesucht. Sind in der Richtung des Aufzugs für beide Rufrichtungen keine Aufforderung von Personen vorhanden, wechselt der Aufzug seine Richtung und prüft dies noch einmal vollständig. Ist am Ende des Scans kein Stockwerk gefunden, in dem ein Aufzug gerufen wurde, dann wartet der Aufzug einen weiteren Takt.

Der vorangegangene Absatz kann wie im folgenden kommentierten Codeausschnitt beschrieben, zusammengefasst werden:

- if on the way up: check if anyone from position to 14 wants to go up.
- if on the way up: check if anyone from 14 to position wants to go down.

  switch direction: up -> down

- if on the way up: check if anyone from position to 0 wants to go down.
- if on the way up: check if anyone from 0 to position wants to go up.
- if on the way down: check if anyone from position to 0 wants to go down.
- if on the way down: check if anyone from 0 to position wants to go up.

  switch direction: down -> up

- if on the way down: check if anyone from position to 14 wants to go up.
- if on the way down: check if anyone from 14 to position wants to go down.

Sind Fälle vom erreichten Ziel, beziehungsweise der angeforderten Etage und des leeren wartenden Aufzugs nicht erfüllt, so wird geprüft, ob sich der Aufzug gerade am Bewegen ist. Ist dies der Fall, dann wird sich die Position in die jeweilige Richtung bewegt.

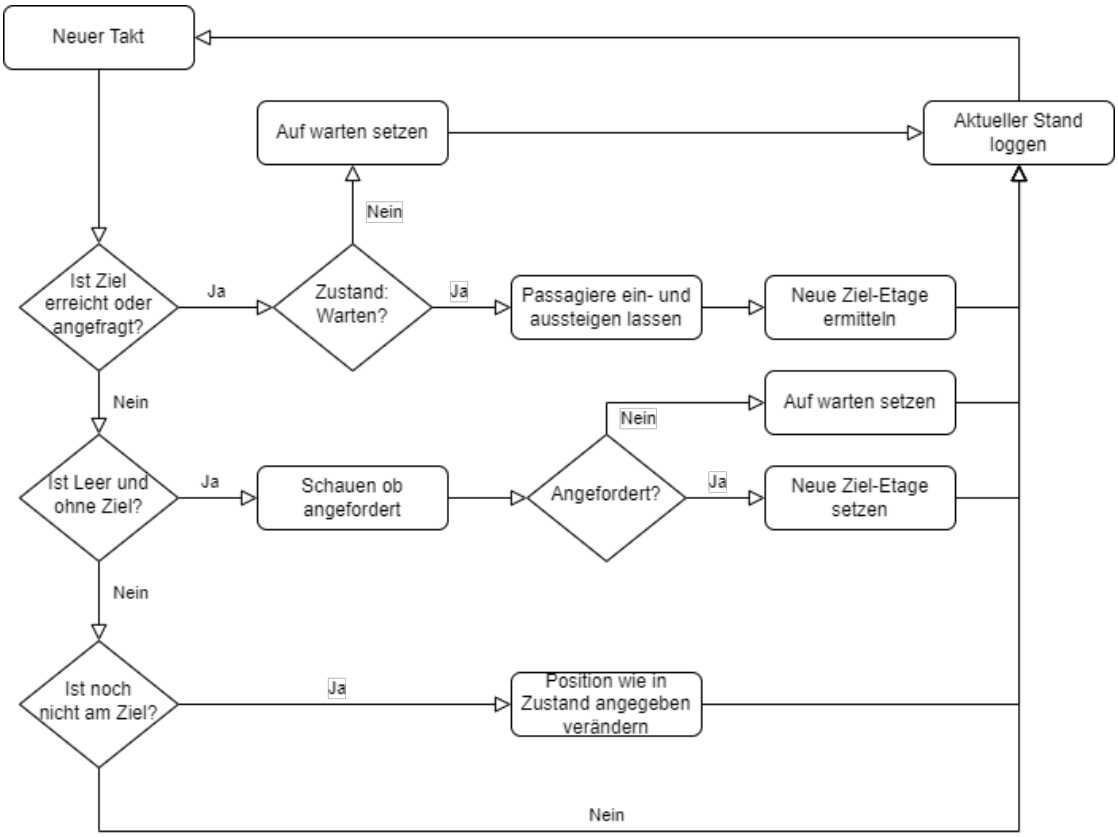

Abbildung 7.1: Prozessablauf eines Aufzugs