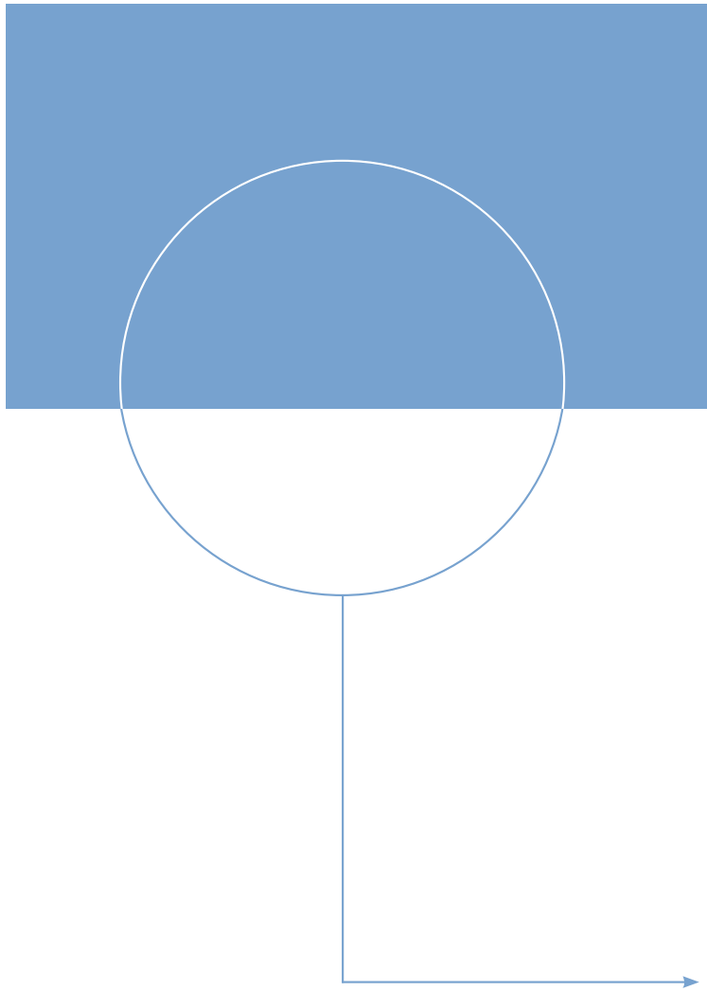

Master's thesis

2019

André Fosvold

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Computer Science

André Fosvold

CitySim: A modular Agent-Based simulation system for Modeling Cities as Complex Cystems

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

CitySim: A modular Agent-Based simulation system for Modeling Cities as Complex Cystems

Master in Informatics
Submission date: June 2019
Supervisor: Sobah A. Petersen

Norwegian University of Science and Technology
Department of Computer Science

Thanks to

My wife, Merete for taking the brunt of parenting while i worked in this project and being a sparring partner for ideas.

Fellow students and friends Andreas Risvaag and Kim Erling Rasmussen for giving advice, pointing out errors and helping with technical problems during development.

Fellow student and friend Andreas Risvaag for spending several days answering questions, and giving advice and critique on the report.

Supervisor, Sobah A. Petersen for giving good advice and pulling me out of my tunnel vision on the code to see the bigger picture throughout the project

Summary

With most of the world population living in cities, the understanding of these systems is increasingly important. In this project the author developed a modular agent-based system for modeling cities as complex systems. The model was validated by running tests and experiments that demonstrated its uses as an aid in understanding the emergent traffic patterns in a city. The experiments were simulation runs that had differing values for prices for buses and cars, variations on points of entry to the city, and varying amounts of parking spaces. The system will be further developed as another master's thesis where the focus will be on the electrical grid and its mutual interaction with other systems in the city. An argument is made for the benefits of using modular and reusable systems in this field.

Table of Contents

List of Tables

List of Figures

Abbreviations

DAG	=	Directed Acyclic Graph
DCG	=	Directed Cyclic Graph
MST	=	Minimum/Minimal Spanning Tree
GUI	=	Graphical User Interface
CAS	=	Complex Adaptive System(s)
ABM	=	Agent Based Model

Chapter 1

Introduction

With over 50% of the worlds population living in cities (Carlo Castagnari (2018)) and our growing understanding of climate change and its consequences, the understanding of our cities and how they operate grows in importance. Cities are complex systems with a plethora of interacting parts and it is difficult to isolate phenomena. To understand and eventually solve difficult problems such as lowering emissions, complex systems modeling is an important tool.

This is a master's thesis in computer science at NTNU worked on over the course of one year full time by the author. This project was supplied and supervised by Sobah A. Petersen with the project description:

> *Modelling Smart and Sustainable Cities as Complex Adaptive Systems ICT plays an important role as enabling technologies in the field of smart and sustainable cities. The Smart cities concept often takes on a limited view of a city and tends to focus on one or few aspects of a city. In this project, we would like to explore the possibilities of modelling a city by looking at it from a holistic way. We would use the ideas of conceptual modelling and enterprise architectures to understand a city. We would also like to explore the city as a Complex Adaptive System and use complex systems modelling approach to simulate how a city evolves. The tasks include:*
>
> - *Literature review of modelling smart and sustainable cities as emergent and complex systems.*
> - *Literature review of how to model a city.*
> - *Design and develop a model of a city as a complex adaptive system.*
> - *Evaluation of the model.*
>
> *The expected outcomes of this project would be a literature review and a simulation of a smart and sustainable city, as a complex adaptive system, and its evolving behaviour. The work could be extended to a Masters project where the model would be further enhanced and evaluated.*

The *smart* and *sustainable* parts were not focused on explicitly, but rather as part of the main focus of the project; the *holistic* view of a city. To achieve this holistic view, it became clear over for first quarter part of the project that modularity would be key in creating the model(simulation).

In addition to the authors work, a group of students produced a decision module for the model as part of an interdisciplinary project course(Experts in Team Work, TDT4857, Smart Citites as a Complex System) run by Sobah A. Petersen.

1.1 Background

To make the simulation modular enough to lay the ground work for a more holistic city simulation, agent-based modeling(ABM) is the technique to use. The framework used for this was Repast Simphony. As Michael J North (2013) et al. says in their introduction in their paper about modeling the framework:

> *Complex adaptive systems (CASs) are composed of interacting, autonomous agents (Holland 2006). CAS agents have properties and behaviors. They interact with and influence each other, learn from their experiences, and adapt their behaviors so they are better suited to their environment(s). By modeling these agents individually, the full effects of the diversity that exists among agents with respect to their attributes and behaviors can be observed as they give rise to the dynamic behavior of the system as a whole. Agent-based modeling provides a mechanism for modeling CASs (Bonabeau 2002; Macal and North 2010). Agent-based modeling has been used successfully to model complex adaptive systems in many disciplines, including archaeology, biology, ecology, supply chains, consumer market analysis, military planning, and economics (North and Macal 2007; North and Macal 2009)*

Looking at a city in a holistic way became a focus of the project and always took part in considerations. Here Yaneer Bar-Yam affirms the importance of the holistic view.

> *Don't assume that only a few parameters are important.The behavior of complex systems depends on many independent pieces of information. Developing an understanding of them requires us to build mental models. However, we can only have "in mind" 7 ± 2 independent things at once. Analytic approaches, such as scaling and renormalization, have been developed to identify the few relevant parameters when this is possible. Information-based approaches consider the collection of all parameters as the object of study. Computer simulations keep track of many parameters and may be used in the study of dynamical processes.*
>
> —Bar-Yam (1997)

Adding more parameters and aspects to a city model is important to capture the more of the effect and get better data. This is exemplified by the fact that personal electric consumption data is classified to protect privacy. One can measure human behaviour by analyzing the data from the electric grid. Although there is an obvious relation between

power usage and the behaviours of people, that relation is not trivial. You lose realism when modeling one without the other.

Adding more parameters, aspects and detail to a model does not come without a cost. That cost is computational power, and that is the ever present trade-off between realism and detail on the one hand, and the availability of computational power on the other.

From conversations with Kieth Downing and Gunnar Tufte it is common practice in a research project to create a specialized hard-coded simulation. They will use it once, discard it, and create a new one for the next project. This project attempts to tackle that trade-off with modularity. Allowing users and developers of the system to be better able to aim at a point in the landscape of detail and broadness. A point which to a higher degree satisfies the constraints in compute power and phenomena of a city that are to be studied.

With this as the starting point work began on simulating car traffic with the intent of adding more and more aspects of a city. These later additions were the buses and their operation, persons and their choices, parking spaces and the accompanying changes to the cars' operation, and finally the electric grid and associated systems. The electric grid was implemented and is functional, but some parts are missing to make it have a mutual effect on the city, such as adding electric cars and a pricing scheme. Therefore it has been dropped from the research considerations, and development of it will be continued by Sebastian Evans. He will continue this project as his masters thesis.

1.1.1 Research Goals and Questions

The research goal of the thesis is to

examine the uses of an agent based, modular micro simulator of a city as a complex system.

The research question is:

How can agent based, complex systems modeling help understand the emergent traffic patterns and how it mutually affects other aspects of the city.

This will be shown by referring to the literature, and in the model by running some simple experiments guided by the questions:

- How does the traffic picture evolve over 7 days based on prices for bus and car travel.
- How does the traffic picture evolve over 7 days based on loads on the points of entry in to the city
- How does the location and count of parking spaces affect the traffic picture.

Chapter 2

Literature Review

At the time of starting the project the author was not familiar with the subject of complex adaptive systems and having no experience in modeling. Therefore a good portion of the literature review time was spent getting familiar with the subjects in parallel with testing the concepts by programming simple models.

The texts that broke through the mists, so to say, and provided an intuitive understanding of what CAS are were Heylighen (2001) and Bar-Yam (1997). They gave an excellent introduction to the field and provided inspiration. Roger White (2015) provided the intuition for the practical aspects in the science of CAS. As they write in their book in chapter 1:

> *Classical science values simplicity in theory backed up by rigorous testing, but this is an untenable position in the disciplines that deal with complex systems. Indeed, disciplines that have opted for simplicity or mathematical rigor in theory, such as economics, have effectively abandoned empirical testing. The Only serious scientific alternative, as we will argue in chapter 2, is to immerse the theory and the models in empirical complexity. Though this seems messy and its results often inconclusive, the approach reflects the nature of the phenomena that we are trying to understand.*

Understanding Complex Urban Systems: Multidisciplinary Approaches to Modeling Schmidt (2014) is a collection of papers which is a collaborative effort to further the understanding of urban modeling. In its introduction paper Jens Martin Gurr and Christian Walloth write:

> *Cooperation between the primarily quantitative disciplines on the one hand and more qualitative approaches from the humanities and some branches of the social sciences is hardly taking place—although it is frequently called for and claimed to be necessary both in research (cf., e.g., Eckardt 2009; Portugali 2012) as well as political research agendas (not least on the EU level).*

It is the authors view that a modular simulation with a simple conceptual design such as this one would facilitate this cooperation they speak of. In the second paper of the collection Ernst Gebetsroither-Geringer writes:

> *It has been recognized that ABM offers a way of incorporating the influence of human decision making on land-use in a formal and spatially explicit way, taking into account social interaction, adaptation, and decision-making on different spatial and (or) hierarchical levels (Matthews et al. 2007, p. 1448).*

As cellular automata is a popular modeling approach in land-use models(Roger White (2015)) it is worth noting that this model has a grid layout of its environment, which is the most common layout for such a system. The above quote about ABM is one of many stated views that ABM of complex systems is an appropriate tool for understanding these systems. An argument which is also made by Michael Batty, which seems to be one of the foremost experts in the field of cities and complexity, in one of his books on the subject(Batty (2005)):

> *... Local movements must thus account for many varieties of behaviour. These will range from movements that are well defined and completely purposive to those that are more random and exploratory, based on walkers who know then environment completely to those who do not know the local environment at all. An agent-based approach is the only way to account for such diversity*

2.1 State-of-the art

In their paper on multilevel traffic models Igor Tchappi Haman (2017) presents the state of the art in the field while also defining the different abstraction levels. They write about both static and dynamic models in the literature. This is a dynamic model and that part is quoted from their paper and a connection to this project will follow.

> *This section presents briefly the various road traffic modeling approaches that are presented in the literature. To this end, two main families of models are presented in the literature: static models and dynamic models.*
> *...*

2.1.1 Dynamic models

A dynamic model is based on the principle of the variability of transport demand in the study period. Therefore, dynamic models are used to describe the physical flow of road traffic. Several modeling approaches were proposed:

- *Microscopic models: It's the most accurate and closest to real behaviors of the entities of the system because it represents the individuals, interactions between the individuals, which create the dynamics of the*

system. There are several approaches. Firstly, the driver behavior based approach called nanoscopic model or behaviour model that emphasizes the real behaviors of the drivers and mathematics approach that describes behavior by equation. Another approach is cellular automaton that presents an interest because its speed and its dynamic behavior, but verification of cellular automaton reveals unconvincing results in urban network and highway at the macroscopic level. Finally, an activity-based approach named TRANSIMS (TRansportation ANalysis SIMulation System) that is an integrated system of travel forecasting based on four primary modules: population synthesizer, activity generator, route planner and microsimulator. Microscopic models lead to emerging phenomena such as congestion, but require a high computational cost.

- *Mesoscopic models: Mesoscopic traffic models can be considered as intermediate between macroscopic models and microscopic models. Two approaches to design are presented in the literature: one in which individual vehicles are not taken into account because the vehicles are grouped in packets or platoons that move along the links, and the one in which the dynamics of the flows is determined by the simplified dynamics of individual vehicles. Mesoscopic exhibits coarse behavior.*
- *Macroscopic models They represent traffic as a flux in analogy with the kinetics of gases. Macroscopic models have the advantage of being simple, easy to manipulate because they need few parameters, making calibration and model validation is easy. Moreover, the execution time of the macroscopic models is acceptable. They are therefore according to an appropriate point of view able to model large-scale systems such as traffic. However, macroscopic models are quite limited in urban areas and they are incapable to model a simple microscopic phenomenon like changing lane, and management of destination of each vehicle is quite difficult with macro models.*
- *Hybrid models: this approach integrates different levels of detail (micro, macro, meso) within the same model. These models are generally called multilevel models. It is an approach that generally combines the advantages of macroscopic models and microscopic models, but it is difficult to realize.*

From these definitions, this model is a microscopic model leaning towards nanoscopic. It can also, in fact, be defined as a TRANSIMS model. This is not the whole of the intent for this project, though it is the starting point. The intended future scope of this project is to capture multiple aspects of a city, and there are some other models that attempt this. Most of them are focused on only a select few aspects selected for a specific purpose. A notable model here is *Tangramob: an agent-based simulation framework for validating urban smart mobility solutions* (Carlo Castagnari (2018)). Their solution models the same aspects(and more) as this project in a similar fashion. Though our projects are similar, there are differences in modularity, scope and focus. Where Tangramob is a framework with a specific bounded purpose, this is built for expansion and adaptability to different projects.

Another related project is CASTLE Birdsey (2016). It is a similar project to this, but much more general. Its focus is CAS in general and not cities in specific. In hindsight it could have been beneficial to use their framework for this project as theirs is also built on top of Repast Simphony. Feng Zhu (2015) presents a *A high performance framework for modeling and simulation of large-scale complex systems* which may be of use for a potential future port to hetrogeneous computing and super computers.

Chapter 3

Basic Theory

3.1 Complex Adaptive Systems

This term seems to defy a concise definition, but the name states, CAS are systems that are complex and adaptive. A *system* is defined in the dictionary (2019) as *a regularly interacting or interdependent group of items forming a unified whole*, and *adapt* as *to make fit (as for a new use) often by modification*. For a definition of *complex* in this context I will borrow from the experts in the field.

3.1.1 Complexity

Bar-Yam (1997) loosely defines complexity in his book like this:

> *A dictionary definition of the word "complex" is: "consisting of interconnected or interwoven parts." Why is the nature of a complex system inherently related to its parts? Simple systems are also formed out of parts. To explain the difference between simple and complex systems, the terms "interconnected" or "interwoven" are somehow essential. Qualitatively, to understand the behavior of a complex system we must understand not only the behavior of the parts but how they act together to form the behavior of the whole.It is because we cannot describe the whole without describing each part, and because each part must be described in relation to other parts, that complex systems are difficult to understand. This is relevant to another definition of "complex": "not easy to understand or analyze."*
>
> —Bar-Yam (1997)

In this project the vehicles in the simulation are the parts that mainly act together and with the environment to form the whole, though also capable of acting on their own.

3.1.2 Emergence and abstraction

Yaneer Bar-Yam goes on to talk about emergence in complex systems:

> *There are two approaches to organizing the properties of complex systems that will serve as the foundation of our discussions. The first of these is the relationship between elements,parts and the whole. Since there is only one property of the complex system that we know for sure — that it is complex— the primary question we can ask about this relationship is how the complexity of the whole is related to the complexity of the parts. As we will see, this question is a compelling question for our understanding of complex systems. From the examples we have indicated above, it is apparent that parts of a complex system are often complex systems themselves. This is reasonable, because when the parts of a system are complex, it seems intuitive that a collection of them would also be complex. However, this is not the only possibility. Can we describe a system composed of simple parts where the collective behavior is complex? This is an important possibility, called emergent complexity. Any complex system formed out of atoms is an example. The idea of emergent complexity is that the behaviors of many simple parts interact in such a way that the behavior of the whole is complex. Elements are those parts of a complex system that may be considered simple when describing the behavior of the whole.*
>
> —Bar-Yam (1997)

From this we can see that there are many levels of complexity to a city. If we take the example of a person driving a car in a city, there are several levels to analyze. In a low abstraction level we can look a the human brain as a complex system, and when go up in the levels we can look at that brain in a body, which is driving a car, on a road with other cars, in a city, and so on. It is important to note that we ascend the levels of abstraction and limit our scope to a part in that level, we lose information. In the same manner as when aggregating a data set with an operation like the mean, the intricacies of the lower levels are lost while we gain an overview.

An emergent behaviour we can witness in this model the accordion movement of the traffic queues, which can be viewed in the videos(Fosvold (2019b)). By accordion movement the author refers to the phenomenon *which occurs when fluctuations in the motion of a travelling body causes disruptions in the flow of elements following it*(Wikipedia.org (2019)), a frequent cause of queues.

As to whether cities are complex systems, the author would argue that they are. This is because cities are characterized by large scale interactions of people and their systems, that is, they have many interactions between parts in a non-linear fashion, such as their social networks.

3.2 Agent-Based Modeling

Agent-based modeling has seen an increasing rise lately with the increasing availability of computing power.

> *In agent-based modeling (ABM), a system is modeled as a collection of autonomous decision-making entities called agents. Each agent individually assesses its situation and makes decisions on the basis of a set of rules. Agents may execute various behaviors appropriate for the system they represent — for example, producing, consuming, or selling. Repetitive competitive interactions between agents are a feature of agent-based modeling, which relies on the power of computers to explore dynamics out of the reach of pure mathematical methods. At the simplest level, an agent-based model consists of a system of agents and the relationships between them. Even a simple agent-based model can exhibit complex behavior patterns and provide valuable information about the dynamics of the real-world system that it emulates. In addition, agents may be capable of evolving, allowing unanticipated behaviors to emerge.*
>
> —Bonabeau (2002)

The agents in this model are the cars, buses and persons. They interact through the traffic system and make decisions based on past experiences, which are measured in the costs of choices.

> *The flexibility of ABM can be observed along multiple dimensions. For example, it is easy to add more agents to an agent-based model. ABM also provides a natural framework for tuning the complexity of the agents: behavior, degree of rationality, ability to learn and evolve, and rules of interactions. Another dimension of flexibility is the ability to change levels of description and aggregation: one can easily play with aggregate agents, subgroups of agents, and single agents, with different levels of description coexisting in a given model. One may want to use ABM when the appropriate level of description or complexity is not known ahead of time and finding it requires some tinkering.*
>
> —Bonabeau (2002)

This points back to the previous section talking about abstraction, as well as underscoring the potential modularity of ABM.

Chapter 4

Experiment

4.1 Tools

4.1.1 Repast Simphony

Looked at different frameworks, but decided to use Repast Simphony because of its flexibility, modularity, because it is focused on ABM, and the fact that it is based on Java, a programming language the author is familiar and comfortable with.

The first tool used for simulation was Unity, a game engine and game development environment chosen because of familiarity and aesthetics. Unity was discarded in favor of Repast Simphony after reading about the different available models. Two main reasons being that unity did not easily support scaling the time, and it would use too many computational resources on graphics and physics that were not needed. From the repast site:

> RepastRepast Simphony 2.6, released on 20 November 2018, is a tightly integrated, richly interactive, cross platform Java-based modeling system that runs under Microsoft Windows, Apple Mac OS X, and Linux. It supports the development of extremely flexible models of interacting agents for use on workstations and computing clusters.

Repast Simphony GUI

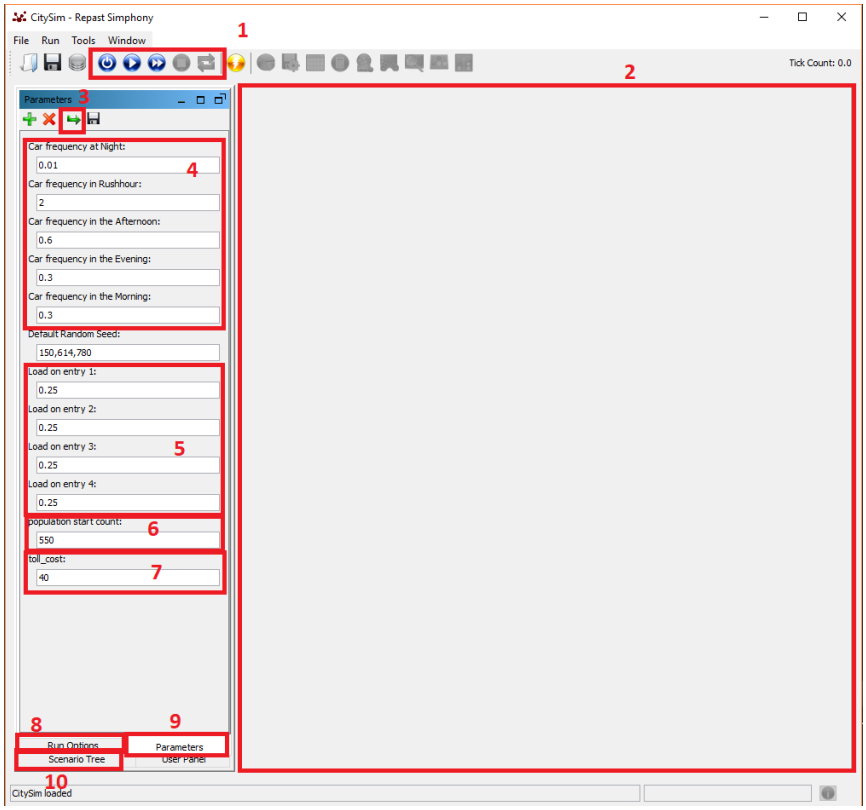

Figure 4.1: The Graphic User Interface of Repast Simphony.

1. From the left: Initialize, start, step one tick, stop, reset.
2. The simulation display where the city view and the graphs are shown
3. Apply Parameter change to this next run.
4. The frequencies at which shoppers spawn, based on time of day
5. Load distribution
6. The number of persons that will be in the run
7. The toll cost of taking a trip with a car. Constant per trip as there are no tolls as of now.
8. Run options: Sets the speed of the run and can set tick counts when the run will pause.
9. Parameters: Currently active tab. Sets values for parameters which are defined in code for the run.
10. Scenario Tree: Where the display, data gathering link and plots are set up.

4.2 Project

The project can be found on git(Fosvold (2019a)).

This is a micro/nanoscopic simulation of traffic, and a mesoscopic simulation of the electric grid for a part of Trondheim. The aspects of a city that were included in the model are traffic(cars and buses) and the electric grid, though other were considered. These are mentioned in chapter 6, future work. They were mostly discarded after deliberations found no implementation plan that would accommodate the time constraints in the project after implementing the traffic aspect.

The project structure is displayed in figure 4.2. The structure was created with modularity in mind. Making it easier to swap out, insert or remove modules. The *Agent* package contains the agents and their super classes. The *Environment* package contains the environmental entities and their helper classes. The *Structures* package contains data structures that are used in different parts on the project. The *Utils* package contains the utilities and tools needed by other classes, such as algorithms and look-up functions. The *CitySimBuilder* class is where all the initialization and setup for the simulation is done. Finally the *Reporter* class keeps track of the measures, calculates the averages when new data comes in, and reports it to the GUI for display.

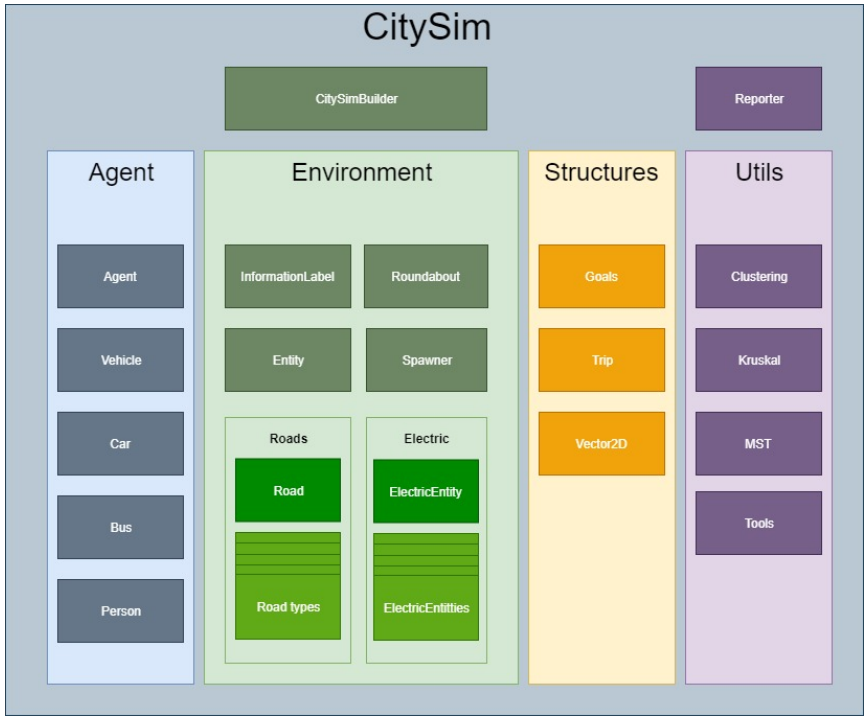

Figure 4.2: An overview of the project structure. Each ”outer” box is a package(folder) and the boxes with no boxes within are classes. So, *CitySim* is the root package which contains 2 classes; *CitySimBuilder* and *Reporter*, and 4 packages; *Agent*, *Envoronment*, *Structures*, and *Utils*.

There are two videos that have been uploaded to youtube showing two different runs of the model. One with just the city view (Fosvold (2019b)), and one also showing the data graphs which is linked to in the video description on YouTube.

4.2.1 Environment

The model environment consists of roads, buildings, side-walks, sub-stations, a road network, and an electric network. These are explained in the subsequent sections, and they are show in in figure 4.4. A relational view of the environment is shown in figure 4.3

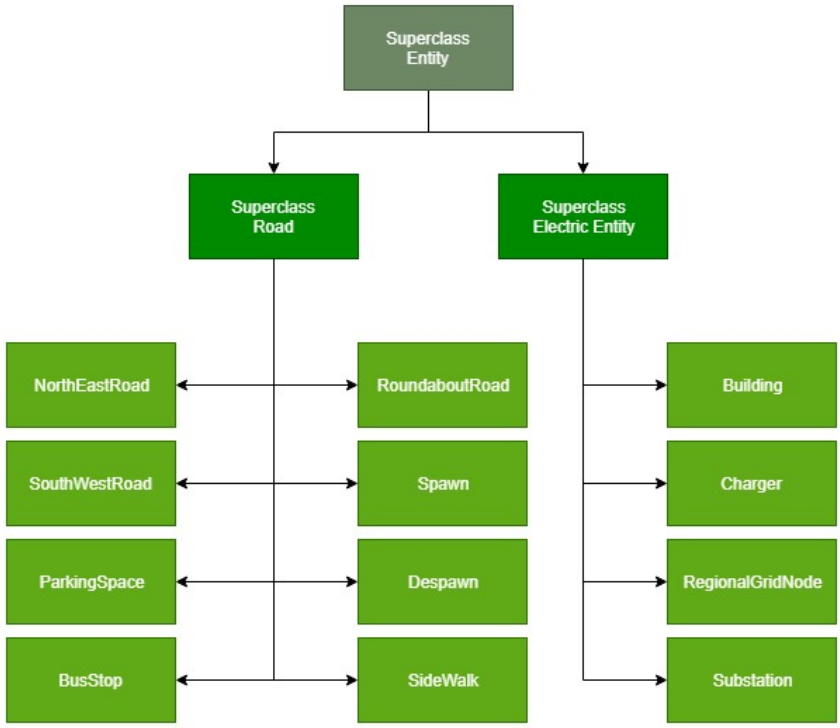

Figure 4.3: A simplified class diagram for the entities.

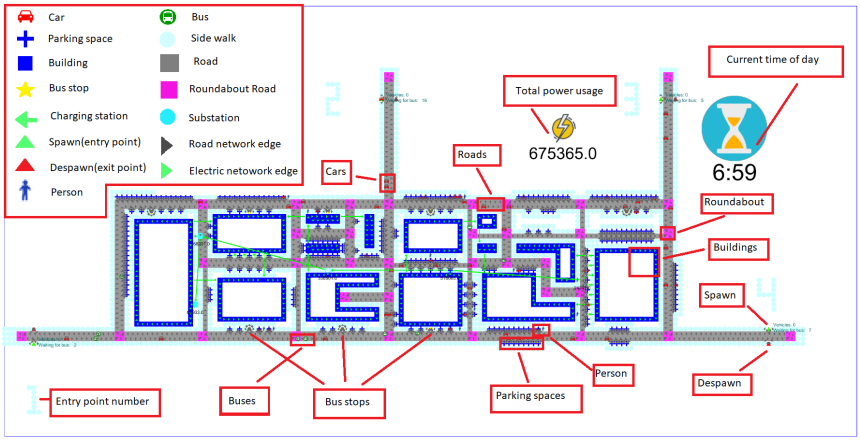

Figure 4.4: An image of the simulation during a run, where the different types of objects are marked with a red box.

Generating the Environment.

The environment is created by going through a PNG image pixel by pixel and reading the RGB values(see figure 4.6). Each entity class in the environment has its own color.This method of generating the environment was first meant as a first step towards having the system be able to taking in almost raw images or maps, it ended up being a convenient way of editing and creating new environments using simple tools like Microsoft Paint. which made sense as reflection revealed that making a dynamic import system would be a project in its own right. When a pixel is read, the build class creates the appropriate entity and adds it to the simulation. If the object in question is a subclass of Road, then a second image is also consulted to get the weight(how hard/slow is it to drive there) for the road. This is to make sure that the vehicles try to keep to the main roads and not use the small roads and alleys too much. Cost in this context refers to the navigational cost when the vehicles are doing their calculations to determine their route, not to be confused with the measure ”travel cost” discussed in the Data Generation section. After all of the entities are created, the networks are created.

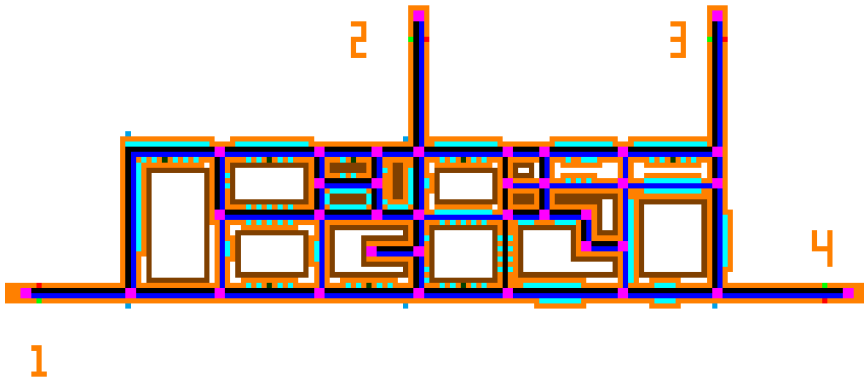

Figure 4.5: The .png image of the city that is used to generate all the non-electric objects in the simulation.

- Black and blue: Standard roads
- Pink: Roundabout road
- Turquoise: Parking space
- Brown: Building
- Orange: Side walk
- Green: Spawn
- Red: Despawn
- Dark Green: Bus Stop
- Blue-Grey: Parking Nexus point
- Numbers in Orange: They are just there to help keep track of which entry point has which number. ”Painted” with sidewalks.

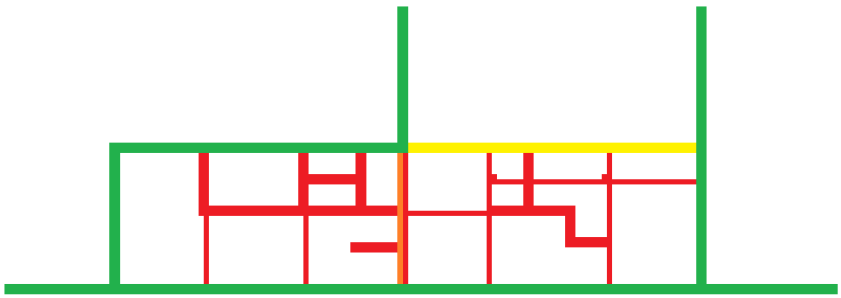

Figure 4.6: The .png ”overlay” image of the roads in the city that is used to determine the weight associated with the road.

- Green: Main road, with a cost of 0.5.
- Yellow: Standard road, with a cost of 2.
- Orange: Small road, with a cost of 4.
- Red: Alley road, with a cost of 8.

Roads

The Road class in the simulation (see class diagram figure) is a subclass of Entity. The roads are the spaces that vehicles can occupy. There are several types of roads, which were implemented to facilitate the driving behaviour and to aid in the setup of the road network.

The Road Network. The road network is what dictates the movement options of the vehicles. The network is a directed graph where the nodes are roads and the edges are allowed movement from one road cell to another. It is constructed by iterating through a list of all the road entities. For each road entity, it is decided whether to add an edge to an adjacent road entity. The decision is made with rules tied to the different sub-classes of roads, which are listed with their descriptions.

Standard Roads. These are the south-west and north-east road types. Although they are essentially the same type of object, it was split to manage the implementation of different directions. This also makes it so that vehicles that are approaching each other in different lanes can do a simple collision test based on road type instead of having to do linear algebra calculations. If the approaching vehicle is driving on the other type of road, then just ignore it. The rules for edge creation is that an edge is added if the direction from *this* to *adjacent* is [south or west]/[north or east], AND *adjacent* is of the same type as *this*.

Roundabout Roads. The roundabout roads are part of a roundabout object. This object determines its center, and keeps track of the member roads and connecting roads for the purposes of constructing the road network. The roundabout nature of the roundabout

is created using equation (4.1). Where A is the roundabout road cell that is doing the check, B is a non-diagonally adjacent roundabout road cell, and C is the center point of the roundabout. It computes the cross product of the two vectors $(C \rightarrow$A) and $(C \rightarrow B)$

$$det = (A_x - C_x) * (B_y - C_y) - (B_x - C_x) * (A_y - C_y) \tag{4.1}$$

If $det > 0$ then an edge is added between A and B in the road network. Earlier versions of the model had traffic lights and right of way rules for intersections, but these were abandoned in favor of roundabouts for simplicity's sake and due to time constraints. Having no traffic lights removes the considerations of $movements$(going right, left, traight), $phases$(grouping movements), designing $ringandbarrierdiagrams$(coordinating the phases), and $timing$(making sure the lanes are cleared druing green, etc). Not to mention coordination of multiple intersections. It is worth noting that having only roundabouts detracts from the realism of the traffic system as they make the traffic flow more fluent than the stop-and-go nature of traffic lights, and the roundabouts implemented in this model seem to be a bit too efficient compared to the real world with personal experience.

Parking Space. The parking spaces are roads that keep track of if a vehicle is occupying it, as well as functions for a car to occupy it on a first come first serve basis. Parking spaces connect to all adjacent south-west or north-east roads, and no other type in the road network.

Bus Stop. Bus stops are essentially parking spaces but only for buses. They have the same connection rules.

Spawn. The spawns are where all the vehicles arrive from. They have two FIFO(First In, First Out) queues; one for vehicles that are to enter the world, and one for persons waiting for a bus. The spawn will check its surroundings each time tick to see if there is space to spawn a vehicle(defined as no other vehicle within 1.6 grid cell) and spawn one when available. When a bus is picked to spawn, it will pick up persons up to capacity, which is 50, before spawning. It has connections to standard roads, but not $from$

De-spawn. De-spawns remove all the vehicles that enter it and trigger their end-of-trip calculations and logging. It has the same road network connection rules as parking spaces. It has connections $from$ standard roads, but not to.

Side-walk. Side-walks are a subclass of Road, though vehicles cannot occupy them. Side-walks are where persons are placed when a car parks, or when a person gets off the bus. The persons will again be picked up when they have finished either shopping or working.

Helper objects

Parking Nexus. These are points in the image that indicate areas where cars should check if they do not find parking. During initialization of the model when these are read from the image, they are mapped to the nearest standard road. Then they are added to a list that cars can choose randomly from when trying to find parking.

Buildings

The buildings in the model are there to act as destinations for the agents, and to act as electric entities in the electric grid. As destinations, they only have their object identity, a location and a list of occupants. As electric entities they keep track of their electric consumption; base, per occupant, and the aggregate of their sub-tree in the network. These electric values are updated upon a change.

Electric Entities

The electric grid is generated automatically on start-up. An overlay PNG image of the city contains the locations and types of the electric entities that need to be networked. The overlay image for the the electric grid is almost identical to the one of the city, except that it has some new pixels for the electric car chargers. The entities are:

- Buildings
- Chargers
- Sub-stations
- Regional grid node

The grid is created in an five-step process:

Step one.	The entities are read from the overlay image and added to the model as their corresponding objects. The image only contains buildings and chargers.
Step two.	Algorithm 1 is run on the current entities added to the model with $k = 5$ and the initial centroids take on the locations of random entities:
Step three.	Substations are added to the model at the location of the centroids.
Step four.	• Algorithm 2 is run separately on the the set of substations and their clusters. • Add an edge from the substations to their closest entity, connecting the substation tree with their respective cluster trees. • Add an edge from the regional grid node to the substation tree. We now have a minimum spanning tree of all the electric entities, but not yet the directions on the edges.
Step five.	The tree is traversed in a depth-first manner, adding the edges to the electric network along the way.

The idea is to create an electric grid that has a distribution network with the substations and a minimal spanning tree of the leaf nodes in the clusters to reduce the length of wire. The current model has 5 substations in the city. During a presentation to Idar Petersen research scientist for SINTEF in the field of renewable energy and smart grids he provided the following feedback to the structure of the electric model.

- Instead of having the network of substations be a DAG as they are now, have them be a DCG for redundancy in the event of failure in the network. Though still do so in a MST fashion.
- Instead of having such large and sequential networks below the substations, it would be better to have many more substations in the network as they are connected to the $11KV$ grid which is better for those distances in city blocks.
- Instead of the focus on a MST, which often ends up being sequential, structure in the nets below the substations, a star pattern would be better. With more substations distributed about the city, that would not necessitate too much of a cable length increase.

Grid interaction. The grid is based on load(the consumption of electrical power). Each node carries the load of its child nodes, such that in the end the regional node(root) has the accumulated load of the entire network. The network is updated on an interrupt basis, where the nodes are updated by outside events and then notify upward in the network. Buildings are updated when an occupant arrives or leaves, and the chargers are updated when ever charging is initiated or ceased.

Data: points in n-dimensional space[X], and k randomly placed centroids[C]
Result: k points at the center of k clusters(centroids)
while *Any* $c \in C$ *has its location changed* **do**
  for $x \in X$ **do**
    Find the closest centroid $c \in C$
    x is added to c.
  end
  for $c \in C$ **do**
    the new location of c is the average of its points
  end
end

Algorithm 1: K-means clustering

Data: points in n-dimensional space
Result: Minimum-spanning-tree
Create a forest F (a set of trees), where each vertex in the graph is a separate tree
Create a set S containing all the edges in the graph
while S *is nonempty and* F *is not yet spanning* **do**
  Remove an edge with minimum weight from S
  If the removed edge connects two different trees then add it to the forest F, combining two trees into a single tree
end

Algorithm 2: Kruskal's Algorithm

4.2.2 Agents

The agents in the model are the vehicles and the persons(figure 4.7). The vehicles follow a list of goals that are destinations in the model, and persons choose travel methods based on previous, or estimated, trips. A lot of the work in the project went it to the agents as they are the heart of the system. The challenges were making them stupid and inefficient enough to be realistic, but also stable in the system. It was surprisingly hard to create sub-optimal behaviour with a reasonable amount of efficiency, and all too easy to create a perfect system with infallible coordination.

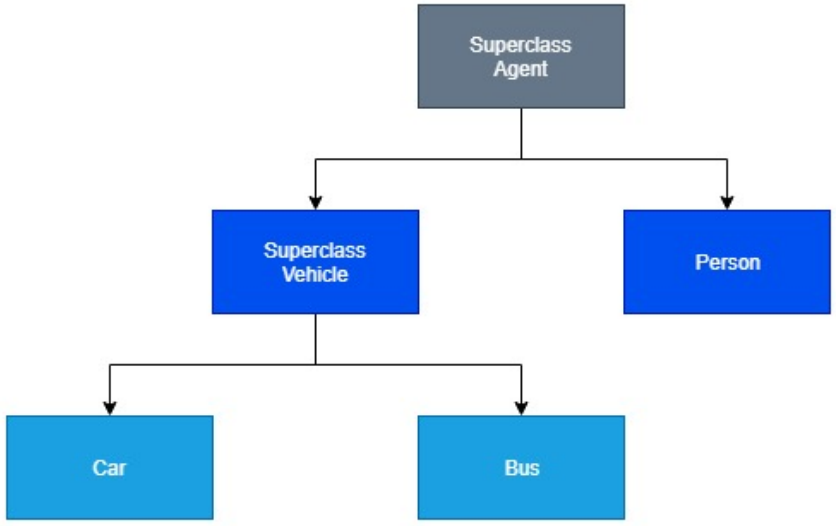

Figure 4.7: A simplified class diagram of the agents.

Vehicles

The vehicles use the road network and the environment to move. The road network provides legal movement options and planning, and the environment informs the rules and provides goals. The last item in the list of goals is always a de-spawn point where the vehicle exits the model. All vehicles have two areas of sensing with their own ranges, one sensing the environment and one for sensing other agents. The one environment sensor looks further afield with a ”radius” of 8 while the agent sensor is very short range with a ”radius” of 3. The ”radius” here has quotations marks since it is in a grid space and does not produce a circle, but a square.

Rules. Several solutions to the traffic interactions were attempted. The first dealt with angles between the movement of vehicles. In this implementation the, a vehicle sensing another agent would calculate its own and the other agent’s direction of movement to determine whether action was required. That approach worked well for straight roads, but caused problems in turns and intersections requiring special exceptions tuning of the angle thresholds. For instance, cars needed to ignore other vehicles waiting their turn in intersections while crossing them, which necessitated the addition of a new state in the vehicles broadcasting whether they should be ignored by others for one reason or another.

Thus, interaction based on angles was abandoned and a new approach based on the type of road was implemented. This implementation implemented functions in the vehicles that detected and broadcast what type of it was currently occupying. Interacting vehicles would exchange this information and consult a rule set to determine the required action. This approach proved unreliable due to inaccuracies in the Repast Simphony framework between the grid space and the continuous(Double precision) space. The interaction rules in intersections also became too complex and unreliable.

The current iteration used the vehicles paths as a basis for their interactions. Only takes action in regard to another vehicles if the other is either in its path, or their paths intersect. A vehicle is defined to in its path if it occupies a Road in the path and that Road is one of its next three steps. Two vehicles are defined to have intersecting paths if any of their next three steps in theirs paths are the same. In addition to these two rules there are some exceptions, like ignoring parked cars.

These rules will, on rare occasion, produce deadlocks. In the absence of human intelligence to negotiate such situations, some checks are performed and occasionally a free pass is given. The checks are triggered whenever a vehicle has to stop due do either a vehicle in front or another intersecting path. The blocking car is also checked to see if they are mutually blocking each other or there is a cycle of blocking cars, such as A blocking B, B blocking C, and C blocking A. Such situations are resolved by giving one of the vehicles a free pass, otherwise one of the vehicles gets a free pass after 100 ticks to catch any unknowns. A free pass means the vehicle can drive through the intersection and other vehicles(if need be) until it is clear of the intersection. Too many of these free passes occasionally produce pileups on the other side of the intersection that have no solution at the moment and the simulation will have to be restarted.

Path Finding. This is where most of the computational power disappears to in the model. The first iteration of the path finding used an algorithm to determine the shortest path from the start to the goal and the agent would just follow the path, but that was too efficient and unrealistic as the agents behaved optimally.

In the current iteration the path finding is done underway. While moving, the agent adds all the roads within the radius of the environment sensor to a list of potential Roads. It then calculates a new path every t ticks to the potential Road closest to its goal using the A-Star shortest path algorithm on the road network. That way the agent navigates optimally locally(An instrumental goal), but by direction globally(Terminal goal), which simulates the tendency for people to navigate by landmarks (Dudchenko (2010), chapter 4). The value of t is to set 1 for now because while higher values saves on calculations, there are some unresolved bugs when using higher values. Agents do not calculate their paths while in intersections to avoid a bug where vehicles would go around the roundabout indefinitely and create deadlocks.

Persons

The population in the model is static and is created at start-up. The number of persons can be set as a parameter in the GUI. 75% of the persons are workers, and the rest are shoppers. 2% of the workers do not go to work each day.

For each trip a person makes a choice between driving and taking the bus. How that choice is modeled is the contribution of the group of students from Experts in Teamwork.

The first way of modeling that choice was that a person would have one variable $p \in [0, 1]$. A p value of 1 would mean a 100% for taking the bus next trip, and a value of 0 would mean a 100% chance of taking the car. The initial value of p was random and a person would compare each trip with the last one of the other choice to update it with a constant Δp value. For instance: A person has an initial p value of 0.6(60% chance of bus) and then chooses to use the car which then costs 250. The next trip the person takes the

bus which costs 200. Now there are two opposing data points, so the person can update its preference, p, with something like a Δp of 0.1. So now the new p is set to $p + \Delta p$, giving it a higher probability of bus as that was the cheaper data point in the history.

The new way developed by the students is to use a binary decision tree, predictions and a factor to model the inveterate tendency of humans. As the model now only has two options and one choice, it is more a branch than a tree, though the system is ready for the structure.

Each time a person is to enter the model, it a probability is triggered based on a probability function:

$$P(X) = \frac{C(Y)}{C(Y) + C(X)} = 1 - \frac{C(X)}{C(X) + C(Y)} \tag{4.2}$$

where X and Y are the two choices in the decision tree, and $C(X)$ is a cost function with two potential definitions: If the persons has a previous trip with that choice, then $C(X)$ is a look-up of the history:

$$C(X) = C'(X)(1 - \alpha)^n \tag{4.3}$$

where $C'(X)$ is the most recent occurrence of choice X, α is the habit factor, and $n \in [1, 3]$ is the number of occurrences of that choice in the history. α is now set to 0.05. A high α and the higher the n, the lower the cost of choice X, meaning a higher probability of X.

If there are no previous occurrences of the choice, then $C(X)$ is a estimation function:

$$C(X) = \sum_{f \in F} A_f C_f \tag{4.4}$$

where C is a set of costs that apply, and A is a set of accompanying alteration factors for those. Below is a view of the code at the top of the person class where these factors are defined:

```
//Factors that apply in calculations (A value of 1 has no
    effect)
//================================================================

//What is the cost of driving: Gas prices, wear and tear, ...
private static final Double DISTANCE_CONSTANT_CAR = 0.012d;

//Cost per distance of of walking from bus stop to destination
private static final Double DISTANCE_CONSTANT_BUS = 1d;

//Time value/dislike of driving (Set low because the tick count
    is pretty high for a usual trip)
private static final Double TIME_CONSTANT_CAR = 0.1d;
```

```
//Time Value/dislike of busses (Set low because the tick count
    is pretty high for a usual trip, but higher than car from
    personal preference)
private static final Double TIME_CONSTANT_BUS = 0.15d;

//To account for the feeling that crowded buses are less
    pleasant
private static final Double CROWD_CONSTATNT_BUS = 1.5d;

//To add some to the toll cost at will. For instance if people
    hate tolls. (Set to neutral at the moment)
private static final Double TOLL_CONSTANT = 1d;

//The degree to which people stick to their habits
private static final Double MEMORY_FACTOR = 0.05d;


//Estimates
//================================================================

//Estimate on how much time a trip will take
private static final int TIME_ESTIMATION = 100;

//Costs
//================================================================

//The cost of a bus ticket
private static final Double BUS_FARE_COST = 40d;

//The toll cost of a trip
private int TOLL_COST;//Set in the GUI
```

These variables are tweakable, and as such most of them are set by feel and by trail and error. Two exceptions are the distance constant for cars and the bus fare cost. The bus fare is set to 40 as that is circa the price in Trondheim, and the distance constant is set by:

What	Amount	Why
Meter per One grid cell in simulation	4	Found by counting pixels in the model and comparing it to the distance it is meant to represent in the real world on the map
Gas usage liter per meter	0,0007	Did not find a reference for this, so it was set by: Personal average gas usage is about 0.00055, and city driving tends to be a bit more.
Gas price per liter	16	From globalpetrolprices.com (2019)
Additional cost per liter gas	1	A rough estimate on wear and tear, insurance range, and accident risk
Price per distance unit in simulation	0,012	Calculated value from the above

Table 4.1: Caption

4.3 Research Methods

The research method consisted of studying the literature of complex systems related to cities, and developing an ABM that could aid in the understanding of these systems and their emergent phenomena.

The running of the model was done by initializing multiple instances of the simulation with differing variable assignments and running them in parallel.

4.3.1 Development

As this was new ground for the author, the process was iterative and proceeded in parallel with learning about the subjects. The iterations mostly took the form of; first adding a new module to the simulation or fixing a problem, then testing and debugging the problems that arose from that, and finally a ”clean up” phase where the system would be made more modular again by separating the modules and their functions.

4.3.2 Verification

With all the myriad ways that vehicles have to interact in a traffic situation, debugging the system was a significant portion of the development time. Problems would arise, such as a time when some few cars would suddenly stop in the middle of the road and stand there for the remainder of the run. The solution to that problem was found by first implementing a new customized debugging tool that would visualize the goals, path, and other interactions for cars. Then observe the runs and wait for the anomaly to appear to analyze it. Once the problem was identified, the solution would often require a redesign of one or more parts.

Testing was mostly done as unit tests and integration tests. The unit tests usually took the form of either printing the status of objects to a console, or attaching a label to the objects that could be read in the GUI during a run. As well as custom tools such as the one described in the previous paragraph.

The integration tests would take the form or doing long runs with the simulation with a few different variable assignments. These would often run at nights and over the weekend.

4.3.3 Validation

In addition to the validation from the experiments, the validation of the model was done by some simple tests suggested by Kieth Downing, and by informal expert opinion. He argued that simple tests were often the best one could do when validating complex systems, as more involved tests could presume too much.

Tests

A very simple test of the persons choice model and it's effect on the simulation was to drastically alter the prices on buses and cars. The first test set the toll price for cars to be 10 000, which is two orders of magnitude higher than an average trip cost. The second test returned the toll price to normal(40) and set the bus fare to 10 000.

First this revealed a bug where some of the shoppers would always use their car, but after that was resolved the results were predictable. Though some would go against the high probability and choose the expensive option, that option would always have virtually no takers. This test also demonstrated how the traffic picture suffers when none used public transport.

The second simple test was on time and distance. The progression of time in the model was first set haphazardly, then in an effort to normalize it was set by gut feeling, and lastly an effort using actual measures of time and distance. The first one had one tick in the simulation represent 30 seconds of real time and the gut feeling one used one tick representing 6 seconds. Kieth Downing suggested measuring the time the vehicles used for a given distance and see of that was reasonable. Thus, a stretch of road in the model that represented one in reality was measured using google maps. Then the time used by vehicles in the simulation was measured and compared to how long a vehicle would use in reality. After doing this test it was found that one tick in the simulation would have to correspond to about 1 second, or change the base speed of the vehicles. It was decided to change the time as that would affect more aspects and help balance out the traffic. The change worked as intended but the simulation runs would take too long to complete using 3 days of real time to run 8 separate runs 2 days in simulation time. Being short on time it was then decided to revert back to the old time and accept the inaccuracies.

A third test was running with differing population counts and see if the results seemed to be reasonable. These tests were run with a population count of 400, 500, 800, and 1000. As expected, the higher the population count(with the same environment), the more traffic is slowed down and trip times increase. The plots from this test can be found as figures 6.1, 6.2, 6.3, and 6.4 respectively, in the appendix .

Experts

The notes from meetings with Gunnar Tufte, A researcher in the field of complex systems(among others) and Kith Downing(AI researcher) are mentioned at the relevant places in the text.

A more formal meeting with Idar Petersen, a Researcher in the field of smart grids and electrical engineering covered the electrical network and its relation to the city. The feedback to the electric system is detailed in chapter 4, but he also talked about the model in general. He was pleased with the construction and potential uses of the model, stating that there is a massive need for models like this in the field of energy, and that energy companies and the municipalities would be interested. A specific case he mentioned to be useful in the model was the daily cycling which is if paramount importance for the energy grids, though it should've had details about days of the week and such. The importance of the more holistic view of this model was that it modeled the movement of people to a higher degree, because the power consuption follows the people.

4.3.4 Data Gathering

As the model grew in complexity the runs took longer and longer. The first solution tried was to solve this by optimizing the code, which worked for a while, but the issue was that the tasks were not running in parallel. The simulation would use only some of the machines resources, running at 800MB RAM and using about one logical CPU core. This was solved by the simple expedient of running multiple instances of the program with differing ranges of the variables. Thus to gather data and debug the system, the author would start as many simulations as there were cores available and let it run over night.

The data was generated through the Repast Simphony framework using their GUI. Through the GUI one can set up data to be gathered at intervals and displayed as plots in real-time as the run progresses. Once the run is completed/stopped one can save these plots as images for later study. The measures used are detailed in tables 4.2 and 4.3 below. The data from table 4.3 is handled by a class called reporter that the persons have access to, and reports to at the end of a trip. The end of a trip being when they reach their end-goal, which is the despawn when they exit the simulation. The averages are calculated by keeping the previous average and adding to it using equation 4.6.

The baseline load distribution used for the other experiments is the uniform one because that is what was used in development and debugging of the system. The differing distributions is a new addition provided for the purposes of the load distribution test. An effect of that is that it makes it a potential source of error.

Measure	Description
Average per bus	The average number of persons on the buses currently active in the simulation.
Counts	The number of cars(red) and buses(blue) currently active in the simulation
Transport use	This plots the number of persons that are currently occupying cars(red) and buses(blue)

Table 4.2: The data gathered by the framework while running. These are some of the tabs at the bottom in the GUI. The other tabs are representations of the data detailed in table 4.3

Measure	Description
Average Car Travel Cost	The accumulated cost of one trip for a car averaged over all the cars for the entire run.
Average Car Travel Time	The time use of one trip for a car averaged over all the cars for the entire run. Minus the time spent being parked.
Average Car Travel Distance	The distance traveled in one trip for a car averaged over all the cars for the entire run.
Average Bus Travel Cost	The accumulated cost of one trip for a person in a bus averaged over all the cars for the entire run.
Average Bus Travel Time	The time use of one trip for a bus passenger averaged over all the bus trips for the entire run.
Average Bus Travel Distance	The average distance from the destination building for the person and the nearest bus stop times 2(there and back again).
Average General Travel Cost	The accumulated cost of one trip for a vehicle averaged over all the vehicles for the entire run.
Average General Travel Time	The time use of one trip for a vehicle averaged over all the vehicles for the entire run. Minus the time spent being parked.
Average General Travel Distance	The time use of one trip for a car averaged over all the cars for the entire run. Minus the time spent being parked. Buses are not included in this one.

Table 4.3: The data gathered at the end of a run when a person returns with a vehicle(car or bus).

$$s = \frac{a_1 + ... + a_n}{n} \tag{4.5}$$

$$s' = \frac{a_1 + ... + a_n + a_{n+1}}{n+1} = \frac{ns + a_{n+1}}{n+1} = \frac{(n+1)s + a_{n+1}}{n+1} - \frac{s}{n+1} = s + \frac{a_{n+1} - s}{n+1} \tag{4.6}$$

4.4 Findings

This section presents the data generated from the model. From several stability tests it was found that a population of around 400 seems to be stable, but a population of around 550 is stable *enough*. Meaning deadlocks are rare enough to not hinder testing.

4.4.1 Load Distribution Test

This is a test that was suggested independently by both Kieth Downing and Gunnar Tufte. The distributions that are tested are detailed in table 4.4 below.

Name	1	2	3	4
Uniform	25	25	25	25
Full 1	100	0	0	0
Full 2	0	100	0	0
Full 3	0	0	100	0
Full 4	0	0	0	100
1-2	50	50	0	0
1-3	50	0	50	0
1-4	50	0	0	50
2-3	0	50	50	0
2-4	0	50	0	50
3-4	0	0	50	50

Table 4.4: The percentages of the different distributions

The desired measure for this test is to see to what degree the different distributions inhibit the traffic flow. The measure will then be $Traveltime/Traveldistance$ to control for different average distances from the different entry points. Only the values of the cars will be used as the distance measure for the buses are the walking distance for its passengers. These values will be read from the last measurement in the plots from the runs, see figures 4.8 and 4.9. To get an overview of the different entrances and the city, see figure(4.10). The table below (4.5) shows the results and the $Traveltime/Traveldistance$ fraction results ordered by $Traveltime/Traveldistance$. Table 4.6 shows the average $Traveltime/Traveldistance$ for each entry point.

Name	Time	Distance	Time / distance
Uniform	640	380	1.68
1-2	445	260	1.71
1-4	660	375	1.76
1-3	690	390	1.77
Full 1	520	290	1.79
2-4	640	355	1.80
2-3	660	360	1.83
Full 2	455	245	1.85
Full 4	825	440	1.88
3-4	900	460	1.96
Full 3	910	440	2.07

Table 4.5: The time and distance results for the different distributions, ordered by time/distance

Entry Nr.	Time / distance
1	1.76
2	1.80
3	1.91
4	1.85

Table 4.6: The average time/distance for each entrance

Name	Time	Distance
Uniform	37.5(5.9%)	-26.25(6.9%)
1-2	42.5(9.6%)	7.5(2.9%)
1-4	12.5(1.9%)	-10(2.7%)
1-3	25 (3.6%)	-25(6.4%)
2-4	0 (0%)	-12.5(3.5%)
2-3	22,5(3.4%)	-17.5(4.9%)
3-4	-32.5(3.6%)	-20(4.5%)

Table 4.7: This table shows the calculated difference between *Average(Full x, Full y)* and *x-y*. Ex.: **1-2** in this table is the difference between *Average(Full 1, Full 2)* and *1-2* from table 4.5

1-4 and 2-4 deadlocked, meaning a traffic jam occurred that did not resolve it self, so they were restarted to run again with the same assignments. Then 2-4 deadlocked 2 more times and needed a restart. To see how a plot looks when it deadlocks, see figure 6.5 in the appendix.

time.png

Figure 4.8: The average travel time accumulated over the course of 7 days simulation time. Yellow is average Car travel time, Blue is average bus travel time and Red is the total average travel time.

- Top left: Full 1
- Top right: Full 2
- Bottom left: Full 3
- Bottom right: Full 4

distrance.png

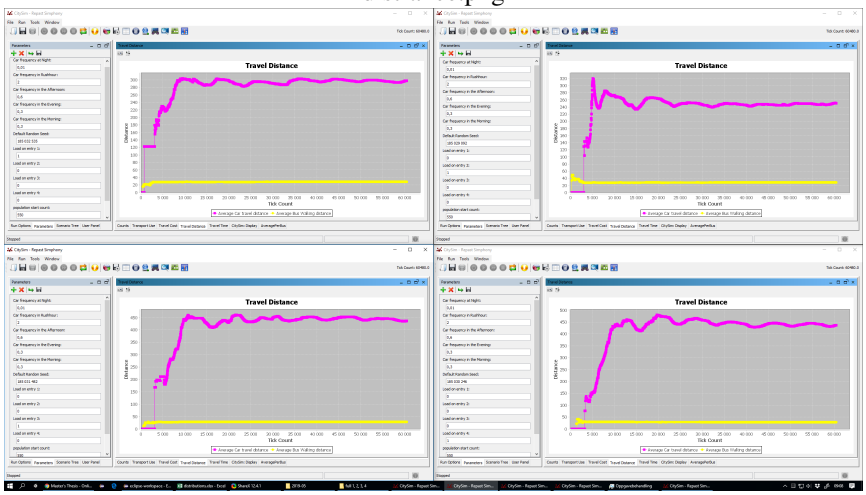

Figure 4.9: The average travel distance accumulated over the course of 7 days simulation time. Yellow is average Car distance time, Blue is average bus travel distance and Red is the total average travel distance.

- Top left: Full 1
- Top right: Full 2
- Bottom left: Full 3
- Bottom right: Full 4

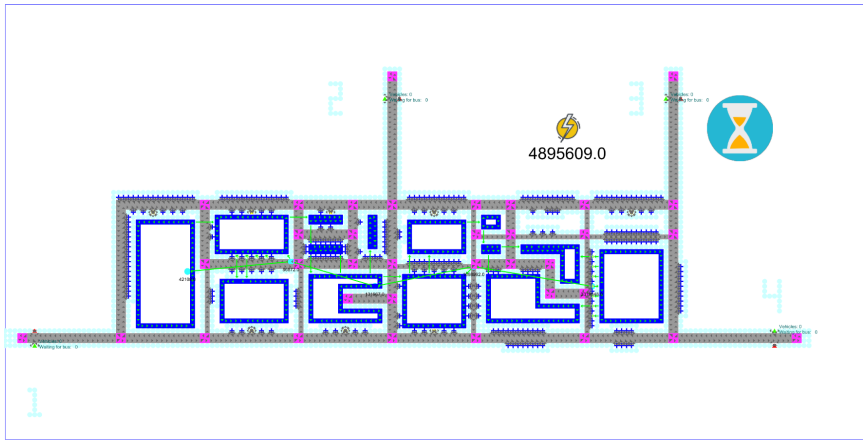

Figure 4.10: Image of the city view without clutter

4.4.2 Pricing test

The pricing test was done by only varying the toll price for the cars. This is because the persons always value their options or relation to each other, so increasing one would be the same as decreasing the other when determining between bus and car. Another reason for the toll price is because it is constant for each trip and does not vary with time or distance, making for a simpler data set.

The data points that will be selected from this will be the peak of the last day within the plots for transport use, which counts the number of persons in cars and buses. This is because the persons will try different options, based on probability(equation 4.2), and will to the degree of the memory factor, α (equation 4.3), stick to that choice when repeated. Picking the last day also gives time to somewhat rectify the situation/balance if the estimation is wrong(equation 4.4).

The toll prices used were of the form 20×2^x and were: 20, 40, 80, 160, 320, 640, 1280. The table below lists the results of the test as well as the fraction Car/Bus, the plots are displayed in figures 4.11 and 4.12.

Price	Car usage	Bus usage	Car / Bus
20	280	100	2.80
40	260	100	2.60
80	260	120	2.17
160	230	120	1.92
320	190	118	1.61
640	120	165	0.73
1280	80	180	0.44

Table 4.8: The transport usage results of the pricing test

4.4.3 Parking Test

The parking test was done by using 3 different maps(.png images) with a large, medium and small about of parking spaces, where the large one is the one used for all the other tests. The medium and small maps were created by removing parking spaces by hand in the image. Medium is about 2/3 the amount of parking spaces as large, and small is about 1/2 the amount of parking spaces as medium. The amounts of parking spaces here is less rigorous as the intent is to see general patterns relations, not concrete numbers. Figures 4.13(large), 4.14(medium) and 4.15(small), show the city maps. Looking at figures 4.16, 4.17 and 4.18 we can see that travel time, travel cost, and travel distance for cars goes down with a decreasing amount of parking spaces. Simultaneously a decreasing amount of parking spaces bring a decreased usage of cars and a different time distribution of the car usage.

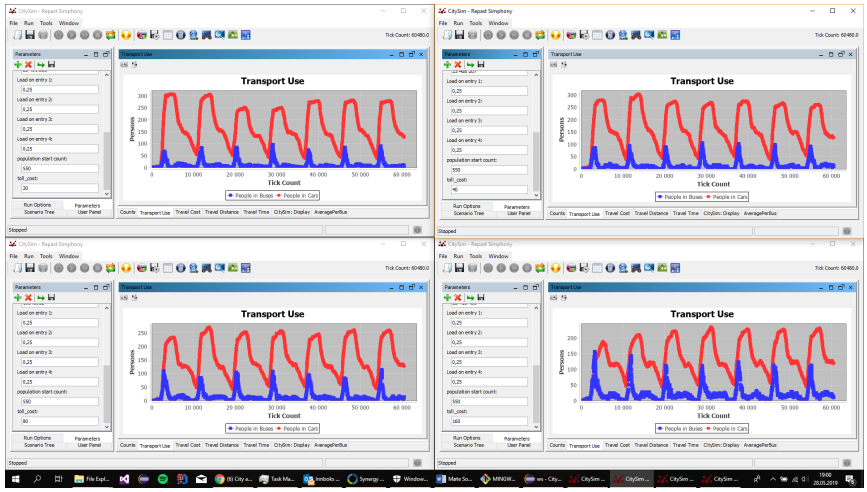

Figure 4.11: The transport usage with the prices set to 20, 40, 80 and 160 in the plots top-left, top-right, bottom-left and bottom-right respectively.

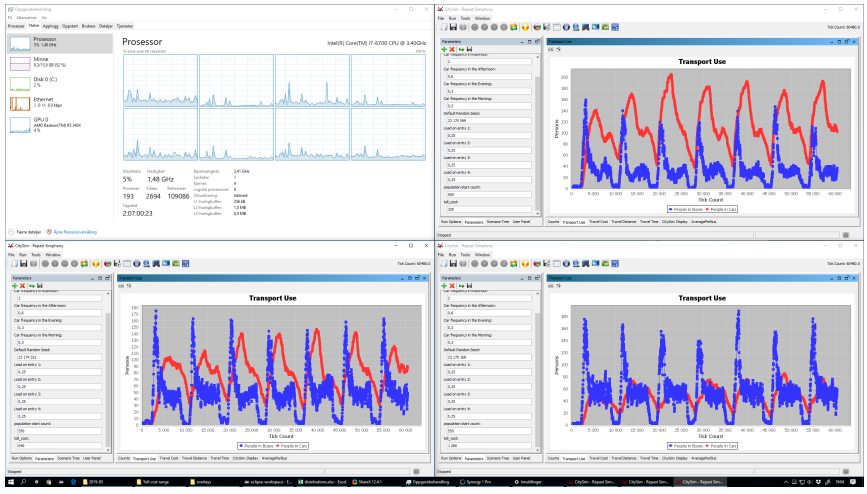

Figure 4.12: The transport usage with the prices set to 320, 540 and 1280 in the plots top-right, bottom-left and bottom-right respectively. Top-left is just the task manager.

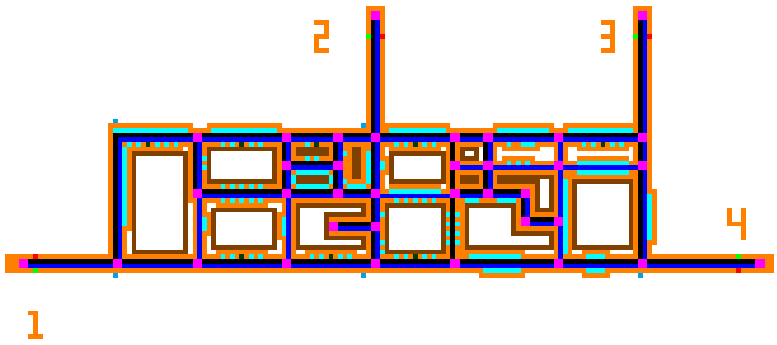

Figure 4.13: The city map with the standard amount of parking spaces(Truquise pixels)

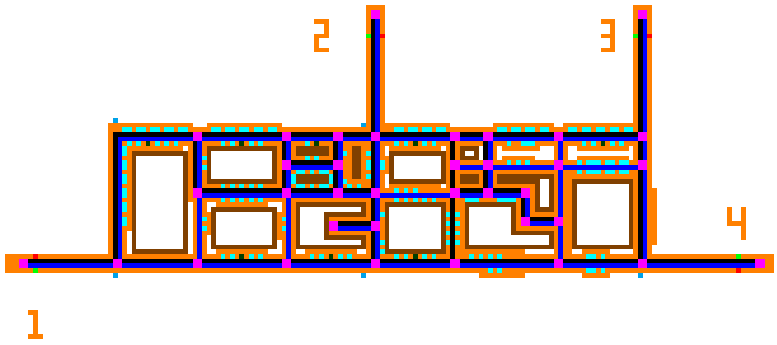

Figure 4.14: The city map with the medium amount of parking spaces(Truquise pixels)

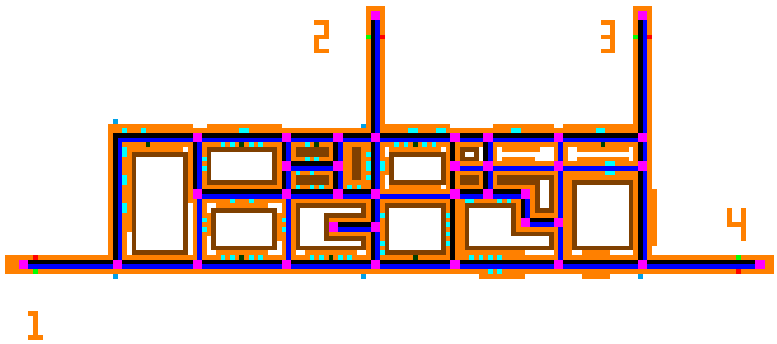

Figure 4.15: The city map with the small amount of parking spaces(Truquise pixels)

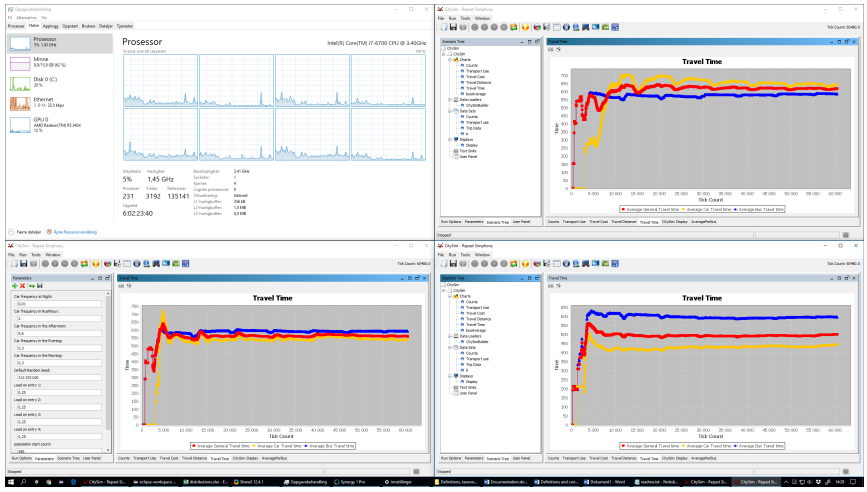

Figure 4.16: Showing the travel time of the different parking maps. Top-right is large, bottom-left is medium, and bottom-right is small.

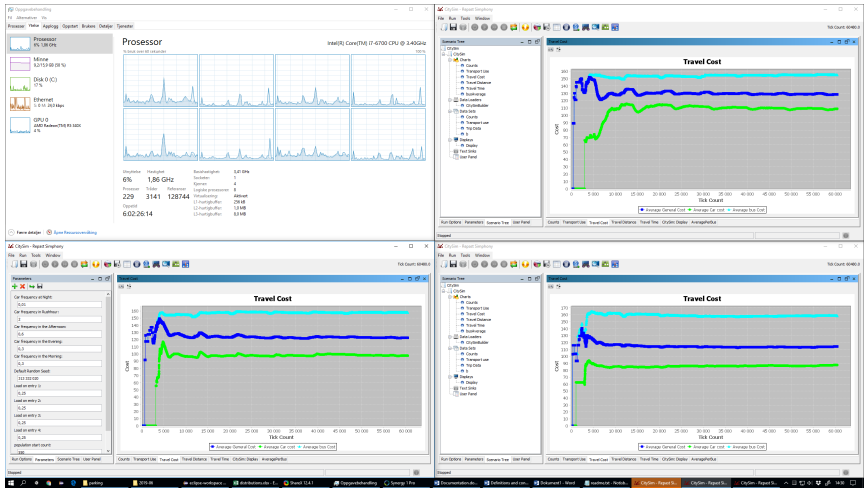

Figure 4.17: Showing the travel cost of the different parking maps. Top-right is large, bottom-left is medium, and bottom-right is small.

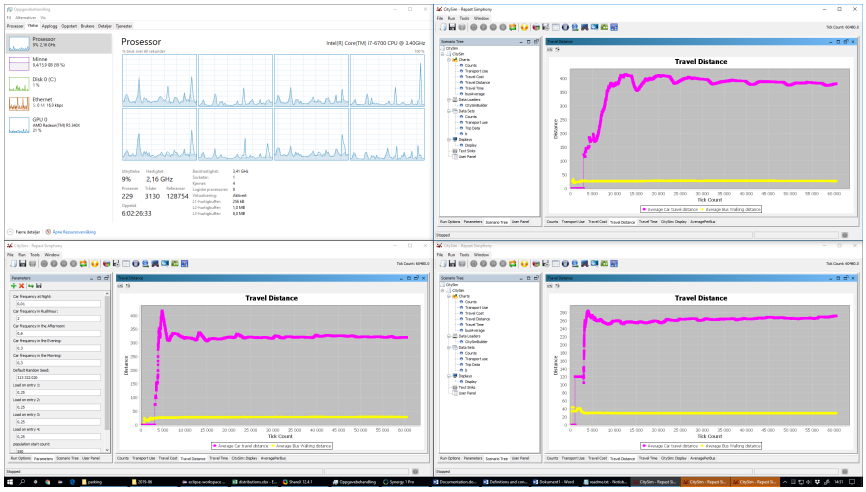

Figure 4.18: Showing the travel distance of the different parking maps. Top-right is large, bottom-left is medium, and bottom-right is small.

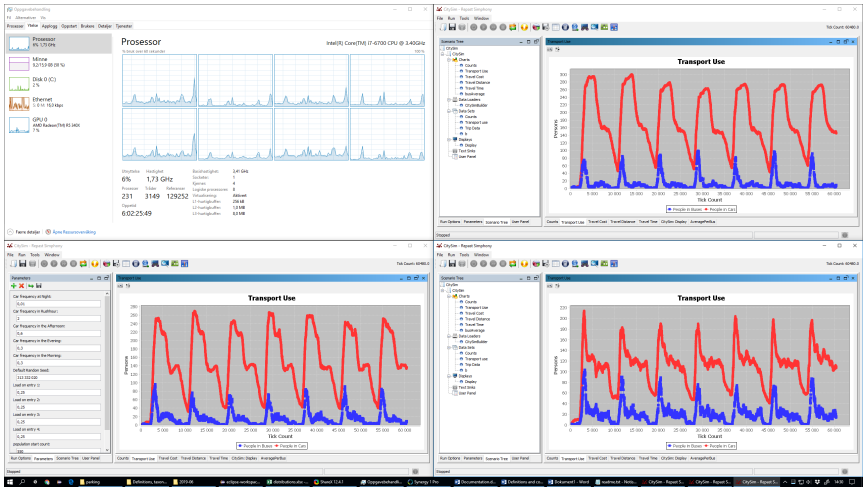

Figure 4.19: Showing the transport usage of the different parking maps. Top-right is large, bottom-left is medium, and bottom-right is small.

4.4.4 Argument for modularity

Modularity

Rick Sturdivant, Senior Member of the IEEE, defines modularity like this one his website:

> *Modularity is a direct consequence of the subdivision of a system into its building blocks. In this way, modularity is a measure of the degree of mutual*

> *independence of the individual components in the system. An important goal of systems engineering is to achieve a high degree of modularity in a way that makes the interfaces and interactions as simple as possible.*
>
> —Sturdivant (2016)

Argument

As mentioned in the introduction, this project attempts to tackle the trade-off between detail and the required computing power by making the simulation in a modular fashion. Figure 4.20 visualizes the authors mental model of abstraction and the interconnected nature of cities. Lets say level 0 is a city at the level of human behaviour, and the different dots are aspects of human behaviour that we measure. We could say that the grey dots in ellipse(grouping) 1 are the positions of people measured over time and space. Then from grouping 1 we could extract and plot the average directions of all the citizens over time to get something like a coarse flow pattern or a vector field overlay on a map. Then, lets say that ellipse 2 is the electric poser usage for those same people. Using that data we could build almost the same map overlay of people's movement from grouping 1, as electrical engineers say; the charge moves with the peoples movement. This example is an argument by analogy for the interconnectedness of cities. In addition to the fact that the factors in a city are interconnected, this connection can itself be *complex*, and as such may not be easily modeled mathematically.

We now have these factors:

- Cities are complex systems and should be modeled as such
- Cities have many aspects to them which are interconnected and may often be so in *complex* ways. Alternatively, it is very difficult to decide which factors actually contribute to a phenomenon.
- Making an ABM for a complex systems is time consuming, and making a new one for each new problem is time consuming.
- Simulating everything is unfeasible due to the required processing power.

From this it is then argued that creating a modular simulation is a possible solution. With a modular solution it is to a much greater degree affordable to make mistakes in choosing ones factors and their abstraction levels to describe the desired phenomenon, something which is mentioned to be such a challenge in complex systems. When a mistake is made, it will take much less time to make a change or add another factor. In addition, with a simple modular design one also has a higher degree of freedom in regards to the abstraction levels. Changing the abstraction level of an agent would require the replacement of that module, and creating a new one that can operate in the environment. Here, changing the environment could also be done to accommodate the new agent.

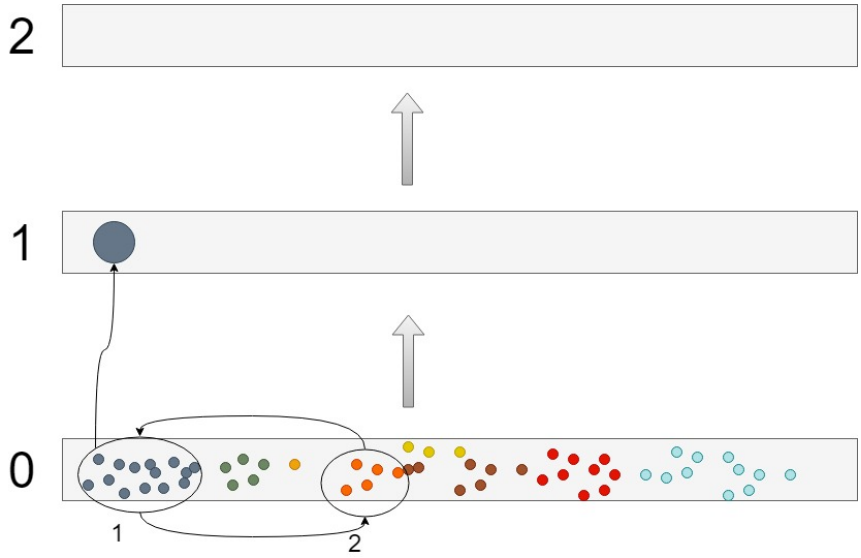

Figure 4.20: A visualization of abstraction levels. Each rectangle is an abstraction level, and the dots within are parts of the world that we observe. The ellipses are selections/groupings of observations. The arrows between objects in a level are interactions, arrows between objects in different levels are abstractions of a group, and the large upward arrows indicate that one cannot go downward.

Chapter 5

Analysis

5.1 Tests

5.1.1 Load Distribution Test

This test shows no large difference between the entry points. The differences between making all the traffic come from two points and having all the traffic come one of those two at a time, then taking the average of that is small. This would mean that the entry points do not by themselves put a high enough load on their single road to slow down traffic to a significant degree. This is likely to be because the distance from the entry point to the first intersection is not sufficient to get a proper ”accordion effect” going. That being said, having the traffic come from multiple entry points performs better in general, with the uniform distribution having the best(lowest time per distance) performance. These results seem reasonable for a section of town this small. The results arguably validate the model in the sense that it will be of use for city planners when considering decisions about roads and their layout.

5.1.2 Pricing test

The pricing test shows the expected result of shifting the population towards using the bus with a higher cost of car travel. An interesting observation here is that travel time goes down with increasing toll prices, which could very likely be because bus travel can support a higher population density on the roads with up to 50 occupants per vehicle which is an order of magnitude higher than a car.

5.1.3 Parking Test

It is surprising that time usage and distance go down with a smaller amount of parking spaces. The expected outcome was that time and distance would increase as more driving would be required to try to find parking. The plots of transport usage(figure 4.19 seem to

show that the cars arrive, look around for a parking then then leave fairy quickly. While the usage of cars go down with decreased parking space, the usage of buses also goes up and is more spread out. It is unclear what is happening here. Having a plot showing when people gave up after not finding parking would be helpful to find out what is happening here, there may be a lot of those cases.

5.2 Sources of errors

This section lists the potential and known sources of error that were thought of.

What	Note	Proposed change
The uniform distribution was the standard during development	This means that all the bug fixing and testing was done with the uniform distribution, and none of the others. Some results may be affected by bugs rather than environment.	Do testing on the other distributions.
The origins and how far away they are from the city are not taken into consideration	As the EiT group found, distance from the city matters for travel choice, especially when the distances get such that the public transport gets sparse.	An idea I had was to try to find or create a map of population density around Trondheim. Then use that map to create a histogram of density and distance. This Histogram could then either be read from or fit a function to. Which would serve as a probability distribution for the population.
The simulation is only of a part of Trondheim City centre	Although the systems supports larger areas, it does increase the run time and I suspect the run time increases with something like r^2 where r is the radius. Adding the suburban areas would probably be a boon to the model.	Better optimized code, and using the compute cluster version of Repast could be enough to be able to add relevant areas while maintaining a reasonable run time.
The economic situation of people is not taken into consideration	From personal experience and observation, those who have a higher disposable income prefer taking the car rather than the bus.	Could at least have a simple income model for the persons, maybe even based on the area they live in.
The total average as a measurement may get too smooth	The averages that are calculated during the runs are always the total and do not reset after a day. This causes a loss of the daily data.	Could have two separate averages, with one that resets at midnight for example.

Table 5.1: The sources of error

By using the average for the entire run and not resetting it each day, the plot gets smoother and more resistant to change. That is why we see their erratic behaviour at the

start and then a convergence.

Chapter 6

Conclusion

Understanding cities and their parts in interaction is difficult as they are complex systems and it can be hard to isolate phenomena within them. Researchers tend to create specialized one-use models for their projects with a minimal amount of reuse.

As a contribution to a solution to this state of affairs, the author has created and presented a city modeling system using Repast Simphony. An argument was made for the benefits and the uses of such a system and how it can contribute to the understanding of cities as complex systems. A group of students created a module for the system which could be integrated in an afternoon, demonstrating the modularity of the system. Lastly simulations were run to validate and demonstrate these uses in the model, showing results that were within expectations and could be used to explore and further the understanding of emergent traffic patterns.

The system was created with a modular and simple design to enable easier further development and changes. It currently models car and bus traffic at a micro/nano level, where the vehicles move and interact on a road network with intersections and interaction rules for these. The model constructs the environment using pixel graphics as a map where objects have designated colors for ease of construction and change. These maps can have overlays to add additional attributes the the objects. People and their choices between buses and cars are modeled. These people have a randomly assigned role as either worker or shopper, and they then have a random goal for a trip. Workers go to work in the morning, work for 8 hours and then go home. Shoppers are random in their trips based on defined frequencies.

6.1 Future Work

This section proposes future work for the system.

- Time/speed change: As mentioned in chapter 4, a discrepancy was detected with relation between time and distance for the vehicles. Fixing this would make the model closer to reality.

- Create an object oriented structure for time that that it can be used as a variable, and also makes it more robust. The time system was tweaked enough to merit a proper structure for it.
- Bus time tables: Add proper time tables for the buses that are taken from reality, as now they just go every 20 minutes.
- Add people just passing through town. Currently all the traffic in the city is contained to those with an errand or are going to work. Adding traffic that is going through would add realism.
- Aspects of a city that were discussed and planned are:
  - Medical care: adding hospitals and the like and enable simlation of injuries and its economical effects.
  - Internet shopping and related service vehicles: Explore the effects of shopping holidays on traffic.
- Additional experiments that can be run with the model as it is or with minior changes:
  - The planning of parking space placement and allocation
  - The placement of charing stations and the effects on the power grid
  - Experiment with adding and removing roads.
  - Add an overlay for tolls and experiment with their placement.

Bibliography

Bar-Yam, Y., 1997. The Dynamics of Complex Systems. Addison-Wesley.
URL https://www.idi.ntnu.no/emner/tdt22/papers/Bar-Yamchapter0.pdf

Batty, M., 2005. Cities and complexity, Understanding cities with cellular automata, Agent-Based Models, and Fractals. MIT Press.

Birdsey, L., 2016. A framework for large scale complex adaptive systems modeling, simulation, and analysis. The University of Adelaide.
URL http://www.ifaamas.org/Proceedings/aamas2017/pdfs/p1824.pdf

Bonabeau, E., 2002. Agent-based modeling: Methods and techniques for simulating human systems. PNAS.
URL https://www.pnas.org/content/99/suppl_3/7280.full

Carlo Castagnari, Flavio Corradini, F. D. A. J. d. B. G. F. A. P., 2018. Tangramob: an agent-based simulation framework for validating urban smart mobility solutions. University of Camerino, via Madonna delle Carceri 9, Camerino MC, Italy.
URL https://export.arxiv.org/pdf/1805.10906

dictionary, 2019. Merriam-webster. springfield, ma, usa.
URL https://www.merriam-webster.com/dictionary/system

Dudchenko, P., 2010. Why People Get Lost: The Psychology and Neuroscience of Spatial Cognition. Oxford Scholarship Online.

Feng Zhu, Yiping Yao, W. T. D. C., 2015. A high performance framework for modeling and simulation of large-scale complex systems. Future Generation Computer Systems.
URL https://www.sciencedirect.com/science/article/pii/S0167739X14002520#!

Fosvold, A., 2019a. Github repository.
URL https://github.com/andrfo/CAS-City

Fosvold, A., 2019b. Simulation run video.
URL https://youtu.be/Azb7dZai46E

globalpetrolprices.com, 2019. Norge bensinpriser, liter.
URL https://no.globalpetrolprices.com/Norway/gasoline_prices/

Heylighen, F., 2001. The science of selforganization and adaptivity. Center "Leo Apostel", Free University of Brussels, Belgium.
URL https://www.idi.ntnu.no/emner/tdt22/papers/heylighenSelfOrg.pdf

Igor Tchappi Haman, Vivient Corneille KAMLA, S. G. J. C. K., 2017. Towards an multilevel agent-based model for traffic simulation. Procedia Computer Science.
URL https://www.sciencedirect.com/science/article/pii/S187705091731092X

Michael J North, e. A., 2013. Complex adaptive systems modeling with repast simphony. Springer Open Journal.

Roger White, Guy Engelen, I. U., 2015. Modeling Cities and Regions as complex systems, From theory to planning applications. MIT Press.

Schmidt, C. W. M. G. A., 2014. Understanding Complex Urban Systems: Multidisciplinary Approaches to Modeling. Springer Complexity.
URL https://link.springer.com/book/10.1007/978-3-319-02996-2

Sturdivant, R., 2016. Modularity in systems engineering.
URL http://www.ricksturdivant.com/2016/06/15/modularityse/

Wikipedia.org, 2019. Wikipedia accordion effect.
URL https://en.wikipedia.org/wiki/Accordion_effect

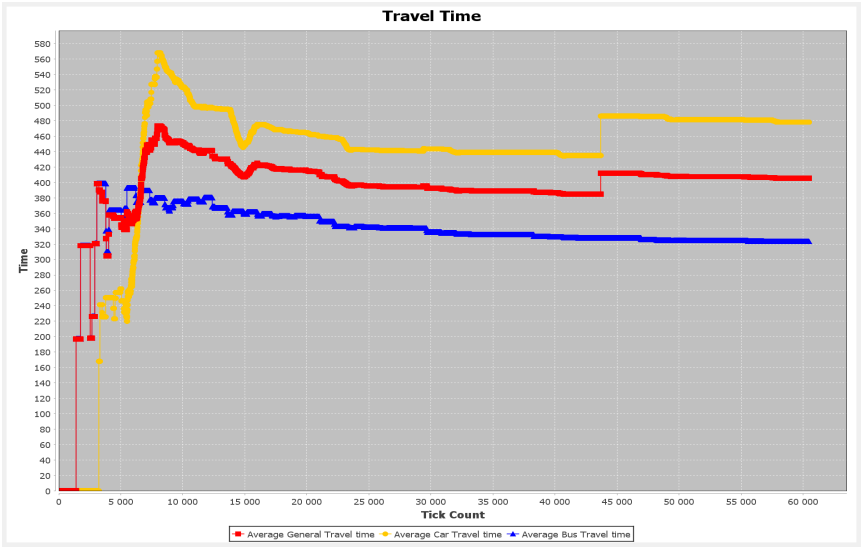

Figure 6.1: The average travel time accumulated over the course of 7 days simulation time. Yellow is average Car travel time, Blue is average bus travel time and Red is the total average travel time. The population is set to 400.

Appendix

6.1.1 Plots

Population test

Load Distribution test

Pricing test

Div

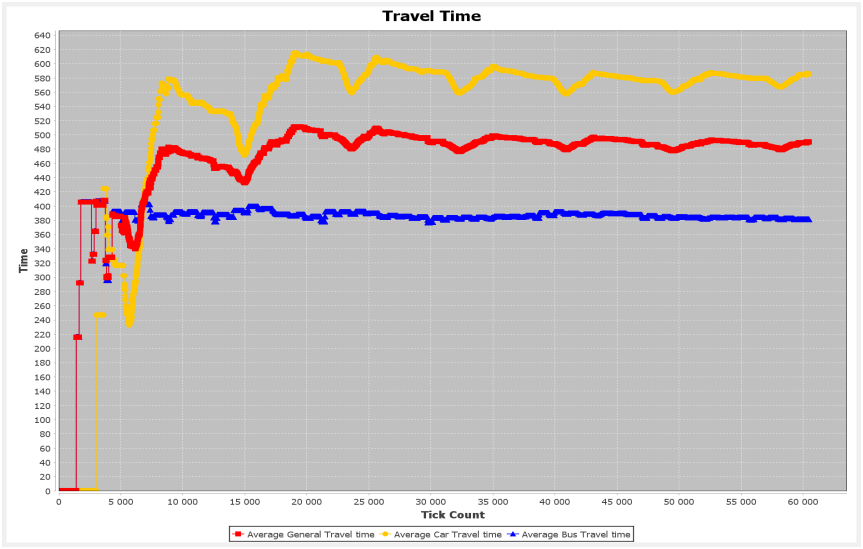

Figure 6.2: The average travel time accumulated over the course of 7 days simulation time. Yellow is average Car travel time, Blue is average bus travel time and Red is the total average travel time. The population is set to 500.

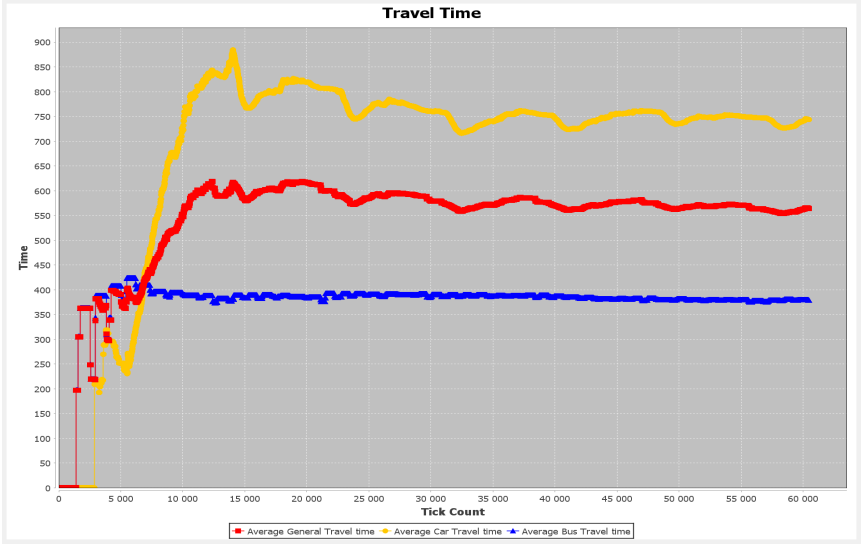

Figure 6.3: The average travel time accumulated over the course of 7 days simulation time. Yellow is average Car travel time, Blue is average bus travel time and Red is the total average travel time. The population is set to 800.

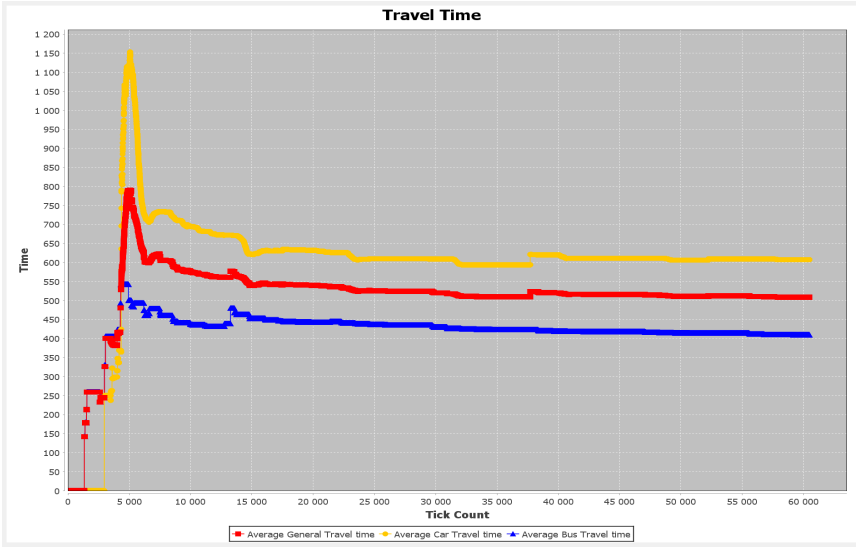

Figure 6.4: The average travel time accumulated over the course of 7 days simulation time. Yellow is average Car travel time, Blue is average bus travel time and Red is the total average travel time. The population is set to 1000.

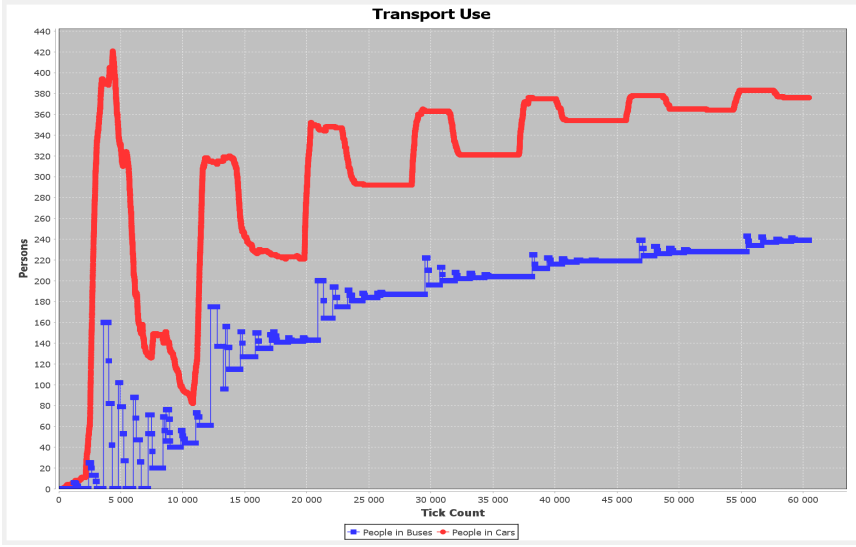

Figure 6.5: The travel choice plots of a run when it deadlocks.