1. Introduction Describe the context for the work and the problem you are addressing. Briefly summarise what you achieved in the project.

This work was first and foremost an opportunity for working in a group. Employers want engineers who are not only skilled in their craft, but also persons who are able to collaborate, work well as part of a team, and invest in project aims and those working to achieve them.

The software engineering part of this project focused on creating a traffic simulation software where we can test various traffic management strategies. We wanted to create a simulation that would be visually interesting with user modifiable variables, and that would have potential for further expansion and use.

This term we were able to create a simulation that is interesting, flexible, and allows different layouts and road policies to be tested and visually represented. Any GIS map shapefile representing road networks can be loaded into our simulator and could easily be extended to take into account any other relevant attributes.

Our team further gained a more practical understanding of agent based modelling (ABM) through using Repast Simphony.

2. Review Describe related work.

[1a] gave us a picture of what an interesting simulation and its code might look like. [2b] offered a good description of what coding with cellular automata involves for those on the team unfamiliar with the concept.

[3c] show one way to conceptualize moving cars along a road from origin to destination.

[4d] Detailed example of an agent-based traffic simulation; referenced for approach and features often considered when creating an agent based model. Well written and fairly comprehensive.

[5e] argues that "only agent-based models can explicitly incorporate the complexity arising from individual behaviors and interactions that exist in the real-world."[5e 1458] Shows the wide uses of Repast in various fields [5e 1457 Table 1]. The article notes that agent modelling using desktop environments like Repast Simphony is a good way to explore the potential of ABM in a brief space of time and with minimal training investment [5e 1464] and this is a benefit our team hoped to gain through this project.

[6f] provided many ideas regarding behaviour rules that was useful, as was the brief but descriptive discussion of modelling.

Selected Bibliography [do not renumber, these are throughout the paper]:

[1a] https://optalk2011.wordpress.com/traffic-simulator1/

[2b] http://natureofcode.com/book/chapter-7-cellular-automata/

[3c] http://crimesim.blogspot.it/2008/05/using-Repast-to-move-agents-along-road.html

[4d] http://www.myhomezone.co.uk/project/Report.htm

[5e] http://www.informs-sim.org/wsc11papers/130.pdf

[6f] http://www.umc.edu.dz/vf/images/misc/session4A/34-4A-paper2-Benhamza-SERIDI%20Hamid.pdf

3. Requirements and design Describe the requirements you set for your project at the beginning and the design you have taken for your project. Focus on why you decided to tackle the problem in the way you did, and what effects that had on the design. You may also wish to mention the impact of team-working on your requirements and design.

Milestone 1:
- code compiles and runs simple simulation
- vehicles run on a map
- variable number of vehicles
- vehicles make decisions to reach a goal

Milestone 2:
- implement basic vehicle type: car
- traffic flows bi-directionally
- junctions: lights, give way, roundabouts
- multiple maps in GIS standard
- speed limits on roads
- cars have different behaviours: timid, aggressive, patient
- multiple lanes on some roads

Milestone 3:
- vehicles appear visually different by some criteria (type/behaviour, speed, congestion)
- implement vehicle types: lorry, emergency, motorcycles
- vehicles exhibit passing behaviours

Possible Traffic Policy Implementations:
- all light junctions
- all give way junctions
- mix lights/give way junctions
- variable speed limits

GIS Maps:

GIS maps were simplified by removing the road coordinates between the Junctions to make it more efficient. We have only one context which builds a road network and the GIS geography. The Road

Network is a directed graph representing the road topology, and the GIS Geography is used to display the simulation.

Any GIS map shapefile (.shp) representing road networks can be loaded into our simulator. The simulator uses some GIS attributes for modelling vehicle's behaviour (ex. road type, max speed, etc.) and could easily be extended to take into account any other relevant attribute.

Framework:

Our team chose to use Repast Simphony. Along with being freeware and cross-platform, it is designed for implementing ABM and uses Java (our language of choice). It gave us the potential to integrate advanced elements into our code, such as importing GIS maps, and allows us to make a more interesting simulation including an attractive display GUI with user defined options at runtime. This was very appealing to our group.

The team decided to use Repast Simphony for a variety of reasons. Repast provides many benefits that made it a good framework for our project. Besides being free to use and open source it has the following benefits that made it a good fit for our group project:

- It is fully object oriented, and easily provides an ABM environment with features such as behaviour activation, a discrete event scheduler and Watcher component, and space and grid management.

- It provides a practical skeleton of agents and their environment that can be programmed using java libraries and .jar utilities.

- It includes charting and data collection functions, so we can focus on deciding on traffic policies and changing these parameters rather than spending time finding ways to display and record the results.

- It provides a graphical user interface (GUI) we can modify to suit our needs. The user can see the vehicle interactions clearly and attractively, and parameters can be varied by the user. The probe function is especially useful for our testing.

As an application programming interface (API) Simphony provides support for processing not directly related to traffic simulation, such as multi-threading and configuration files. It allows the option of writing one's own plugins, should that become desired at a later time.

We used Eclipse IDE as it is preconfigured for Simphony. Eclipse Simphony varies from the normal Eclipse IDE in that it adds three .jar files to the buildpath: JOGL (Java bindings for openGL), Java3D, and JAI (Java Advanced Imaging). These files are used to create graphical representation of objects.

The biggest downside of using Simphony for this kind of project is that there are more opportunities to display a mastery of modelling concepts and logic than a mastery of Java programming skill. However, upon choosing ABM as our design platform, behaviour and goals became more central than designing passive functions [4d], and we trust our ability to create an interesting system shows in our project and does credit to our ability.

An unforseen disadvantage of Simphony arose for us in that it cannot be installed or used on KCL lab computers. One member of our group did not have access to Simphony at home or on campus (the whole group did not know this until after the initial presentation), and he was not able to find a way to use Simphony until almost March, and then not effectively. This greatly reduced his ability to be useful to the team in coding and held up work on the BDI segment of our code.

Our work. While Simphony provided many tools to increase the speed at which we could code, it did not provide any logic for the agents. While it provided a very useful API, our team determined how each agent and behaviour should be modelled and the logic used to describe each object. Our team wanted to especially take advantage of Simphony's visualisation benefits, and our core requirements were based around the logic of the agents, not their appearance.

Overall our use of Repast Simphony was successful for our project. It provided a foundation API and Eclipse configuration that was not directly involved with agents or behaviours. It allowed us to create attractive visuals in the simulation and analysis graphing. We spent little time debugging, and were able to focus on simulating traffic.

Originally we anticipated using a belief-desire-intention (BDI) model to code the agents' behaviour, but as the member intending to code this was unable to use Simphony, behaviours were instead coded into the agents themselves.

Agents and Environment:

Our team used Traffic Simulation Theory. We used a discrete time model, where state is updated every tick of the clock.

The simulation models entities and features including (but not limited to): vehicles, roads, junctions, road network topology, and GIS geography.

Repast offers a ContextBuilder component containing agents and environments. Our agents included objects such as vehicles, junctions, and traffic signals. The environment is coded as a continuous space grid with an NxN matrix, in continuous space with (x,y) coordinates, on a directed graph network.

The matrix allows the agents to know their real position and determine the distance between objects. Having a continuous space coordinates allows the user interfaced to display the agents in detail during the simulation. The network represents roads as edges connected by nodes as the intersections.

"Agents also exhibit flexible behaviour… This means they are capable of reactive, proactive and social behaviour. Reactive means they can perceive and respond to changes in their environment, proactive means they can take the initiative to achieve their goals, and social means they can interact with other agents to satisfy their objectives" [4d]

Simulation engine

- Simulation Model class loads the environment and starts the UI.
  Road network can be loaded via csv file.
- ContextBuilder is a Repast component which contains:
  - Agents: Vehicles, Junctions, TrafficLights etc.
  - Environment: Grid, Continuous Space, and Road Network digraph

Environment

- Grid: N x N matrix, used by Agents to know their real position
- Continuous Space: (x, y) coordinates used by UI to display Agents during simulation

- Network: a directed graph, representing roads (edges) and intersections (nodes)

Agents

- move within the Environment
- know their position at any time
- can inspect Environment to interact with other Agents

Behaviour of Agents

- Simulation time is measured in discrete ticks
- methods can be scheduled when time of action is known at compile time.
  ex. every tick compute next location of Vehicle, or every 4 ticks Traffic Light becomes red/green
- 'watch' annotation checks other Agents' attributes at run time and react according to their behaviour.
  ex. if vehicle ahead is slowing down, then slow down

- Setting up global integration system (GIS) maps to be imported easily.

For an idea on how to build the chapters and what kind of content we want to put on the report refer to the following – which is excellent: [4d]

I would cover pretty much the same areas plus the ones we discussed today:

TRAFFIC ANALYSIS

- Introduce briefly traffic analysis and what can be understood by the model. Ex. how to build effective policies? What policies will the simulator implement for effective traffic management? T. lights, give way signals and so on..
- Driver's behaviour analysis
- Need for a simulator, why is it useful?

FROM TRAFFIC SIMULATION THEORY… (a lot of references needed here!)

- Continuous vs discrete simulation. We use discrete time model: every tick, the state is updated and agents perform some action based on their behaviour.
- Microscopic vs macroscopic modelling: we use microscopic
- Simulation: what are the features and entities our simulator will model in practice? Different speed limits per type of road, multi-lane highways, traffic lights and/or give way signs, different vehicles type (ambulance, cars, trucks etc), driver's behaviour (recklessness), vehicle's attributes (max speed, acceleration level). (…BDI should take driver's behaviour and vehicle's attrib. and come up with actions to be taken at any time…)

… TO A MODEL (more details soon!)

- Vehicle
- Roads
- Junctions
- Road Network Topology
- GIS Geography

AGENT BASED SYSTEM: REPAST SIMPHONY

- Why ABM and Repast symphony?

…

…

LOGIC (check the actual code for this, you will find a lot of stuff in the comments, I will put placeholders so that you can search it in the code)

- From GIS map shapes to Agents
- Vehicle following
- Acceleration: kinematics equations
- Collision avoidance
- Traffic light management
- Roundabouts (TBD once and iff multi lanes done)
- Give way management (TBD)
- Lane changing (TBD)
- Calibration/Accuracy of simulation → constant values used to make simulation more realistic

IMPLEMENTATION & EVALUATION

- We have a very similar project structure, just check what he's done
- Updated class diagram
- Sequence diagram showing interaction of vehicles and other classes during every step(), which is the central part of the simulation. Ie. what messages are exchanged between the classes at every iteration of the simulation.
- Evaluation and testing is being developed by Adeela, she will give you more material

EMPIRICAL EVALUATION

- Data collection: what data is relevant? How can we extract it? (Waqar could do it?) Repast can provide both aggregate and non-aggregate data. Check the zombies tutorial.
- Show some data we collected in tables and pictures and what can we deduct from it

CONCLUSION

- What can we get from the simulation/data collected? What are the best policies?

FUTURE WORK

- How could the simulation be extended? Ex take into account pedestrians, vehicle crashes, etc.

What are the disadvantages of Repast Simphony?

---

4. Implementation Describe the most significant implementation details, focussing on those where unusual or detailed solutions were required. Quote code fragments where necessary, but remember that the full source code will be included as an appendix. Explain how you tested your software (e.g. unit testing) and the extent to which you tested it. If relevant to your project, explain performance issues and how you tackled them.

The architecture for our simulation is a hierarchy of Java classes, where each class is an agent or a context. Behaviour can be embedded through the agents reacting to the environment themselves, by using the watcher component which listens for events and triggers actions, or by globally scheduling events to happen. No main method is required; the program is built around a context builder.

After the build, the “Parameters” and “Run Options” tabs have user defined options. Initialize; GIS display will appear, run.

Code Overview:

Under src > signalGreen:

- singalGreenBuilder.java	This is the contextBuilder file, replacing the usual “Run” method in java. It initializes the traffic simulator, including loading map, assigning and mapping junction types, and creating lanes.
- vehicle.java	Contains all vehicle behaviour and decision making. Uses kinematics equations. Assigns vehicle properties such as velocity and displacement.
- utils.java	multiple lanes, returns junctions for behaviour;
- constants.java	Very boring.
- coordinateAgent.java	Very boring.
- gisAgent.java	GIS coordinates, projections
- junction.java	[Detail] Controls vehicle junction behaviour. A queue of vehicles is formed at each junction
- laneAgent.java	na
- light.java	light object for traffic light junctions; returns color of current signal
- priorityBlockingDeque.java	**** This object was written by someone else
- Road.java	extends gisAgent. Roads, speed limits, lengths
- trafficLight.java	subclass of junction.java; changes the light color

Testing:

ABM presents some challenges for testing. Agents are non-deterministic and there is no set predictability of where objects will be at a given time. There is not a set result that can be expected for each vehicle, so automated testing for agents is not generally possible and testing must be done visually. Automated testing will be done for….

[Adeela’s report content summarized and report document attached]

5. Team work Describe how you worked together, including the tools and processes you used to facilitate group work.

Our approach to the project was to set down ideas, goals, and tools in the first two weeks, then jump into the coding and get as far along as we could until the initial report came due. This worked well in many ways.

Tools:

- GitHub. GitHub was used for creating the code, providing version control and easy sharing of code. Initial and final reports were also kept there. Andrea held the main code repository, and the team branched off that code.

- Communication. For communication outside coding, the most used tool by far was WhatsApp; this was practical for team chat, answering basic questions, and coordinating meetings. Longer reports for the group were written on asana or sent by email. Asana was used consistently by team members and was helpful for the documentation of the final report.

- Meetings. When one or more group members believed a whole group meeting was needed to further the project, this was communicated on WhatsApp and a time and place of meeting was arranged using input from all members. These meetings were in person. Other times two or three members met together using Skype or in person as needed to work on parts of the project pertaining only to them.

Team Challenges:

Our team faced several challenges in working together and with our chosen tools. [Note: all members are here referred to as ‘he’, and different paragraphs may refer to one or more persons.]

- Our team struggled to coordinate regarding GitHub the first part of the project, as several members focused on coding on their own machines rather than branching from GitHub in the weeks prior to the initial report date. Initial code was uploaded to GitHub the day before the initial report was due. The members leading in coding later created a full GitHub project for everyone to access.

- One team member had strong feeling regarding GitHub and architectural forms, but he missed two central meetings (he said he would attend) where decisions were being made, and so this person’s choices were not implemented. This was difficult for the person to accept and caused some friction.

- One member did not inform the group until after the initial report presentation that he did not have the ability to run Simphony and so did not have a way to code his part of the project. He found a way to partially resolve this so he could code from the last week of February, although still without full Eclipse functionality. (His coding was not implemented in the end.)

- After difficulty getting team members to follow through with attending agreed meetings, we voted to modify our practice so that absence to full group meetings would result in a point for that person being deducted.

Team Roles:

It quickly became apparent that Yoann and Waqar were very skilled with writing code. They created the general framework for the project and got the initial simulation working by the first week of February. James took responsibility for coding the BDI from Yoann, and Adeela worked with helping Waqar. It soon became clear that Waqar was made progress more quickly than Adeela, so we agreed she should shift to creating test cases for the project. Andrea focused specifically on the documentation and on coordination for the team. James transitioned to assisting Andrea with documentation.

6. Evaluation Critically evaluate your project: what worked well, and what didn't? How did you do relative to your plan? What changes were the result of improved thinking and what changes were forced upon you? Note that you need to show that you understand the weaknesses in your work as well as its strengths. You may wish to identify relevant future work that could be done on your project.

Our team was able to create a simulation that involved each member in the process. In spite of differing abilities and communication styles, each person was able to contribute in unique ways.

Roles and Task Division:

The division of coding labour did not work out as originally intended. One member focused solely on documentation and coordination, and this was a decision based on listening to others who have worked on projects before. We intended for the coding to be written for easier division of labour, so that from February we would be all be able to code separate parts of the system. However, Yoann and Waqar were skilled in coding and also very ambitious. We found they occasionally rapidly implemented work that others had been assigned, so Adeela suggested she be moved to testing, and Andrea began focusing solely on documentation and communication.

Development and Reporting:

Originally we intended to utilize an Agile method of development with SCRUM reports once a week, either in person (if we had a meeting) or written if we did not gather. This part of the project did not pan out as anticipated; early meetings had absences, which made SCRUM impractical as we waited until other members arrived (which they occasionally did not).

Our actual method of development looked more like a Waterfall style of development. One member had industry experience with Waterfall, and with the short time frame of the module and meeting reality of this project Waterfall would probably have been an equally appropriate choice. In the end, our development looked closer to this model.

As of mid-February, the group coordinator began asking for WhatsApp/asana updates from the team approximately every week so everyone was aware of what the others were doing and to help remind the team to continually progress. Team members requested help as needed from each other and there was always a visible "next task" waiting when one's current task had been completed, so while SCRUM was not overtly performed the goals behind the method did remain.

Further work:

"Validation data is usually macroscopic statistics such as flow rate, speeds and queue time, which can easily be compared with data from real traffic experiments." [4d] If we had more time we would have liked to download a road map from a town in England where we know road data and compare our model to actual roads. This would be a big step in validating our logic and making the code useful to others wishing to model traffic.

- We did not make progress on the BDI as planned, so this was left out of the code. The code could be restructured to more neatly implement this.

- Further development can be done in displaying road closures (for modelling construction).

- Implement source/sinks for allowing traffic to come on/off the map.

- Adding other vehicle types such as motorbikes or road features such as pedestrian crossings to model their effect on congestion and traffic flow.

7. Peer assessment In a simple table, allocate the 100 ‘points’ you are given to each team member. Valid values range from 0 to 100 inclusive.

Waqar:	
James:	
Adeela:	
Andrea:	
Yoann:	