This is a very rough draft for documentation including referenced group information.

NOTE: Red text indicates some suggested groupings of the report from Dr Tratt. We will not exactly follow these, but they're a good start.

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 the 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.

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 of us unfamiliar with the concept.

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

References [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.

Integrated Development Environments:

Our team chose to use the IDE NetLogo, specifically RePast Simphony. Along with being freeware and cross-platform, NetLogo 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.

RePast Simphony is ideal for Agent Based Modelling because it allows programmers to focus on modelling the BDI and road networks and spend less time debugging code. It has libraries already created for coding the behaviours agents need to be aware of their environments and to move along road edges.

The biggest hazard of using an IDE like Simphony for this kind of project is that there are more opportunities to display a mastery of modelling concepts 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 (this was not communicated 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.

Other options we considered included …

"can be used to develop agent models, although the agent-specific functionality has to be written by the developer from scratch, as there are no dedicated libraries or modules that focus on agent-based modeling." [5e]

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] Further possible steps for this project could be to download a road map from a town in England where we know road data and compare our model to actual roads.

- We did not make the progress on the BDI that we hoped, so taking that further would have been our next priority.

- 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 (to model their effect on congestion and traffic flow).

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?

---

17 Jan (Waqar)

Functional Requirements:

- System should allow user to set parameters that include: time step, minimum speed, max speed, min acceleration, max acceleration, density and scenario, use of graphical user interface.
- The system should allow the user to add obstacles, lane closures, set traffic lights, and crossings by use for the GUI.
- System should visualize the simulation in the gui
- System should implement a lane-changing algorithm to enable vehicles to follow lane discipline system should implement behavioural algorithm to enable vehicles to adjust their speed and direction according to the situation
- System should display congestion level based on flow of traffic
- Differentiate between normal and large vehicles
- System should provide realistic data found in traffic studies

Non-functional requirements;

- the system should have the ability to pause and stop the simulation at any given time.
- The system should differentiate vehicles in congestion by colour so they can be seen apart
- Should be able to handle flow of hundreds of vehicles at the same time.

- vehicles stop at red signals behind the displayed signal

- there is always space between vehicles (vehicles do not collide with other vehicles)

- vehicles follow traffic signals

-

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.

Testing: because our code is agent based, agents are non-deterministic and there is no set predictability of where objects will be at a given time. As there is not a set result that can be expected for vehicle, so automated testing for agents is not generally possible and testing must be done visually. Automated testing will be done for

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

GitHub was used for creating the code, providing version control and easy sharing of code. Initial and final reports were also kept there.

For communication outside coding, the most used tool by far was WhatsApp; this was practical for team chat and coordinating meetings. Longer reports for the group were written on asana or sent by email. The team used these tools effectively to communicate. The group met about every other week in person to discuss where we were in coding and delegate tasks; often two or three members met as needed.

[Note: all members will be refered to as ‘he’, and different paragraphs may refer to one or more persons.]

Our team struggled to coordinate regarding GitHub, as several members focused on coding on their own machines rather than utilizing GitHub in the weeks prior to the initial report date. One person uploaded code 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 member did not inform the group until after the initial report presentation that he could not run Simphony on any of his computers and so did not have a way to code his part. This was partially

resolved so he could code from the last week of February, although still without full Eclipse functionality.

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.

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.

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 over coding the BDI, 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. In March James worked on documentation of the project with Andrea.

---

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.

The division of coding labour did not work out as originally intended. One member focused solely on documentation and coordination, and this was a result of 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 skilled in coding, and also very ambitious. In several hours we found they implemented work others were intended to do, so Adeela suggested she be moved to testing, and Andrea began focusing solely on documentation and communication.

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: