

Traffic Simulator

Revised Software Requirements Specification

**Khalid AlHokail
Luke Bay
James Grady
Michael Murphy**

**Version 2.0
03/07/2007**

Table of Contents

1. Map Model	2
1.1. Graph	2
1.2. Rules for Intersections	3
1.3. Number of Lanes	3
1.4. Status of Lanes and Intersections	4
1.5. 2D Map - Visualize Flow (over time)	4
2. Driving Model	5
2.1. Time of Day	5
2.2. Population Distributions	6
2.3. Traffic generator	6
3. Traffic Light Control	7
3.1. Rules for one concrete location	7
3.2. Traffic light state and timing	7
4. Decision Model	8
4.1. One concrete strategy for vehicle speeds and speed limits	8
4.2. One concrete strategy for vehicle status and capability	8
4.3. Driving speeds for individual cars	10
4.4. Route	11
5. Architectural Diagram	12

1. Map Model

1.1. Graph

1.1.1. All the nodes must be reachable.

- 1.1.1.1. The graph must be a connected graph that can be traversed from any node to any node.
- Mapped to the 2D map.

1.1.2. The properties of outgoing edges (that represents roads) and nodes (that represents intersections) must be readable, at each node, and must be assigned valid values, and cannot be assigned invalid values.

- 1.1.2.1. A value of a property can be produced by a system entity or a user.
- The system entity will be mapped to traffic simulator.
 - Users are represented by Users that interacts with the user interface.
- 1.1.2.2. A specific type of system entity called a constraint evaluator must determine the validity of each value.
- Mapped to the constraint evaluator.
- 1.1.2.3. Each property of each node or edge must be assigned only valid values.
- 1.1.2.4. Assignment must consult the constraint evaluator.
- Mapped to the Constraint Evaluator
- 1.1.2.5. If the constraint evaluator determines that the value is valid, assignment is performed.
- Assignment is mapped to Assignment.
- 1.1.2.6. If the constraint evaluator determines the value is invalid, an exception will be generated by the assignment and no assignment will be performed.
- Mapped to data storage.

1.1.3. The graph shall be a directed graph.

- 1.1.3.1. If a road is a two way street, then two different edges exist, one for each direction.
- Mapped to the 2D map

1.2. Rules for Intersections

1.2.1. The intersection must only allow legal turns.

- 1.2.1.1. Legal turns are stored in each node.
 - Mapped to Data storage
- 1.2.1.2. There shall be a directed edge for each direction a car may travel.
 - Mapped to the 2D map

1.2.2. The intersection must include a configuration of allowed turns (intersections, roundabouts, dead ends, one way streets, and highway onramps).

- 1.2.2.1. Allowed turns are determined by the strategy adopted by the Decision Model.
 - Mapped to the Decision Model
- 1.2.2.2. Each node will include all the allowed turns that can be made from that particular node.
 - Mapped to Data Storage.
- 1.2.2.3. Allowed turns are stored as an enumeration (e.g. Left, Right, Straight means that this node only allows Left, Right or going straight, and doesn't allow anything else).
 - Mapped to Data Storage.

1.3. Number of Lanes

1.3.1. The lane capacity shall be represented by an integer greater than or equal to 1.

- 1.3.1.1. Lane capacity represents the number of lanes on an edge.
 - Mapped to Data storage.

1.3.2. Each lane shall be divided into car-length segments.

- 1.3.2.1. Each segment will accommodate only one car at the same time.
 - Mapped to the 2D map.
- 1.3.2.2. A system entity called segment divider is responsible for dividing the lane into segments.
 - Mapped to segment divider.
- 1.3.2.3. Each lane segment shall correspond to a real world segment that is 8 feet long by 5 feet wide.
 - Mapped to the 2D map.

1.4. Status of Lanes and Intersections

1.4.1. The number of drivable lanes shall be represented by an integer greater than or equal to 0, and shall never exceed the maximum allowed capacity.

- 1.4.1.1. Maximum allowed capacity is same as the lane capacity defined in 1.3.1.
 - Mapped to Data storage.
- 1.4.1.2. A drivable lane is a lane that has not been closed because of a collision.
 - Mapped to the 2D map.

1.4.2. The number of drivable lanes shall be updated in real time when a lane is closed or opened.

- 1.4.2.1. A lane will close if a construction starts on that lane or if an accident occurs on that lane.
- 1.4.2.2. A lane will open if the construction finishes or the accident is cleared.
- 1.4.2.3. When a lane is closed, the number of drivable lanes is decreased by one, and when a lane is opened the number of drivable lanes is increased by one.
 - Mapped to Event Handler.

1.5. 2D Map - Visualize Flow (over time)

1.5.1. The 2D map must display roadways, intersections, and traffic distributions (the number of cars at a specific position and time) or individual cars.

- 1.5.1.1. This 2D map will be shown as a top-down view.
 - 1.5.1.2. The 2D map shall be displayed using a 4:3 (length:width) ratio.
 - 1.5.1.2.1. The view must display an area of $\frac{1}{4}$ of the total map.
 - 1.5.1.2.2. The view must show a mini-map, split into four sections, which overlays the upper right-hand corner of the screen and represents the complete map in miniature.
 - 1.5.1.2.2.1. The mini-map must display an indicator of which section the user is currently viewing.
 - 1.5.1.2.2.2. When a user selects a new section in the mini-map, the map section must change to this new section.
- 1.5.1 is mapped to the 2D map.

1.5.2. The map must provide real-time updates (the simulation display keeps up with actual time).

- 1.5.2.1. With each real time update, for every object with a changed attribute, if that attribute is visually represented by a sprite (i.e. position, color, creation, or elimination) the refreshed screen will display this change.
 - 1.5.2.1.1. The refreshed screen will change the color of the objects
 - Mapped to Colors.
 - 1.5.2.1.2. The refreshed screen will change the object either by removing it or changing its position.
 - Mapped to Sprites.
- 1.5.2.2. With each real time update, every object may change attributes when the screen is refreshed.
 - Mapped to Event Handler.
- 1.5.2.3. Visually represented changed attributes must be displayed on the 2D map.
 - Mapped to the 2D map.
- 1.5.2.4. When a sprite changes color, the new color shall be shown on the sprite.
 - Mapped to colors.
- 1.5.2.5. When a sprite is created, eliminated, or changes position, it must be created, eliminated, or repositioned on the refreshed screen.
 - Mapped to sprites.

2. Driving Model

2.1. Time of Day

2.1.1. The time of day shall be represented.

- 2.1.1.1. The time of day shall be stored in a class, similar to UNIX's ctime class, which provides a representation of the year, month, day, hour, minute, and second.
 - Mapped to Time Passing.

2.1.2. The update time shall be done according to simulation time.

- 2.1.2.1. One second of (wall) clock time = α seconds of simulation time, so $\alpha = 1$ would be real time. And $\alpha = 60$ would speed the model, so one minute of simulated time would pass for each second of clock time.
 - Mapped to Time Passing.

2.1.3. *The time of day shall be displayed to the user.*

- 2.1.3.1. The format of the time shall be mm/dd/yyyy hh:mm:ss.
 - Mapped to Data Storage.
- 2.1.3.2. The time shall be displayed as part of the legend.
 - Mapped to Data Display Plots.

2.2. Population Distributions

2.2.1. *Static population distributions shall be provided.*

- 2.2.1.1. A random number, > 10 , of start and end nodes shall be generated, and each of these nodes shall be outside the boundary of the displayed map.
 - Mapped to Data Storage.
- 2.2.1.2. Each of these nodes shall be associated with 2 double arrays:
 - a. Array1 = $[r_{\max}, w_{\max}]$ where r_{\max} is the maximum number of residents for the associated node, and w_{\max} is the maximum number of workers for that node.
 - b. Array2 = $[r_{\text{cur}}, w_{\text{cur}}]$ where r_{cur} is the current number of residents for the associated node, and w_{cur} is the current number of workers for that node.
 - c. For each node, r_{\max} and w_{\max} will not change, while $0 \leq r_{\text{cur}} \leq r_{\max}$, and $0 \leq w_{\text{cur}} \leq w_{\max}$.
 - Mapped to Data Storage.

2.3. Traffic generator

2.3.1. *The traffic generator shall populate the map with vehicles.*

- 2.3.1.1. Throughout the day, there will be random traffic migrations chosen from random start nodes to random end nodes.
 - Mapped to Traffic Simulator.
- 2.3.1.2. For each day there will be a traffic migration from home nodes to work nodes in the morning, and, in reverse, from work nodes to home nodes in the evening.
 - Mapped to the Traffic Simulator
- 2.3.1.3. Every migration will have the following constraints:
 - a. The start node may send from 0 to a maximum of m cars to the end node, where

$$m = \min(\text{start node } w_{\text{cur}}, (\text{end node } w_{\text{max}} - \text{end node } w_{\text{cur}})).$$
 - b. A start node has its $[r_{\text{cur}}, w_{\text{cur}}]$ decremented by one when a car leaves, and an end node has its $[r_{\text{cur}}, w_{\text{cur}}]$ incremented by one when that car arrives.
 - Mapped to the Traffic Simulator.

- 2.3.1.4. Every path shall start and end with one of these nodes.
 - Mapped to the Traffic Simulator.

3. Traffic Light Control

3.1. Rules for one concrete location

3.1.1. The rules of the location shall dictate the order in which the lights will cycle.

- Mapped to Data Storage.

3.2. Traffic light state and timing

3.2.1. The lights shall be represented by enumerations and guide the drivers at intersections.

- 3.2.1.1. The enumeration shall contain the colors Red, Yellow, and Green.
 - Mapped to Data Storage.

3.2.2. Each traffic light shall have a separate timing.

- 3.2.2.1. Each light color shall have a separate timing associated with it.
 - Mapped to Time passing.
- 3.2.2.2. The timing shall be expressed as cycle time in milliseconds.
 - Mapped to Time passing.
- 3.2.2.3. When the cycle time has passed, the stored color of the light will change.
 - Mapped to Data Storage.
- 3.2.2.4. When the cycle time has passed for a specific light color, the displayed light color shall change on the screen.
 - Mapped to Colors.
- 3.2.2.5. Light colors shall initially be cycled at random lengths of time.
 - Mapped to time passing.
- 3.2.2.6. The user must be able to change the cycle time of each light, individually.
 - Mapped to Data Storage.

3.2.3. The current status of the light shall be displayed.

- 3.2.3.1. The light's current color shall be displayed on the intersection as a sprite.
 - Mapped to Colors.

- 3.2.3.2. When the traffic light color changes, the displayed sprite must also change.
 - Mapped to sprites.

4. Decision Model

4.1. One concrete strategy for vehicle speeds and speed limits

4.1.1. The drivers shall follow edges, road signs and speed limits.

- 4.1.1.1. The driver shall be able to read the properties of edges and nodes.
 - Mapped to the graph
- 4.1.1.2. The driver can only drive on roads, and roads are represented as edges.
 - Mapped to Trigger object specific behavior.
- 4.1.1.3. The driver can only travel from one node to another if these nodes are connected by an edge.
 - Mapped to Trigger object specific behavior.
- 4.1.1.4. Each edge will have a value proportional to its length. Edges are instantiated as contiguous road segments, and each road segment has its own speed limit.
 - Mapped to Data storage.

4.2. One concrete strategy for vehicle status and capability

4.2.1. A driver will try to drive around a blocked road segment.

- 4.2.1.1. When a vehicle is in motion and tries to move onto a blocked segment, it will stop and move to an adjacent open lane if one exists.
 - Mapped to Trigger object specific behavior.

4.2.2. The vehicle must never break down or run out of fuel.

- 4.2.2.1. All vehicles have an unlimited supply of fuel.
 - Mapped to Traffic Simulator.
- 4.2.2.2. Vehicles will not stop unless they are a part of a traffic incident or are blocked by traffic congestion.
 - Mapped to Trigger object specific behavior.

4.2.3. The status of the vehicle shall be displayed (start node, current node, next node, end node, edge position, speed).

- 4.2.3.1. By selecting a vehicle, an information box will appear that displays the real time status of the vehicle's start node, current node, next node, end node, edge position, and speed.
- Mapped to Data Display plots.

4.2.4. The number of traffic incidents shall be determined by a probability for each vehicle.

- 4.2.4.1. The probability that a vehicle will cause an accident is a random number p , such that $0 \leq p \leq 0.001$ for a roadway segment that is not an intersection.
- Mapped to the Traffic Simulator.
- 4.2.4.2. The probability that a vehicle will cause an accident is a random number p , such that $0 \leq p \leq 0.005$ for an intersection. An accident at an intersection will only occur if the portion of the intersection that is perpendicular to the vehicle's direction is spanned by stalled traffic.
- Mapped to the Traffic Simulator.

4.2.5. When a traffic incident occurs, the map model is updated with the proper road segment or intersection closures.

- 4.2.5.1. Accidents located on a roadway will block the roadway segment that the accident occurred on. The blockage will not allow vehicles to travel on the blocked segment.
- Mapped to the map model.
- 4.2.5.2. Road segment closures shall be displayed as a black road segments.
- Mapped to Colors.
- 4.2.5.3. Accidents located on an intersection will restrict the allowable turns by blocking the roadway segment directly behind and the segment to the right of where the collision occurred in the intersection. So, if there is an accident in an intersection, then oncoming traffic cannot make a left turn at that intersection.
- Mapped to the map model.
- 4.2.5.4. The blocked segments of an intersection shall be displayed as black segments.
- Mapped to Colors.

4.2.6. Traffic incidents shall be represented on the 2D map.

- 4.2.6.1. If two or more moving objects (cars) occupy the same space, then a collision occurs.
 - Mapped to Detect Object Collisions.
- 4.2.6.2. If a car on segment i collides with a car on segment $i + 1$, both cars will occupy segment $i + 1$.
 - Mapped to the Traffic Simulator.
- 4.2.6.3. Each traffic incident shall be displayed as a flashing red \emptyset .
 - Mapped to Data Display Plots.

4.3. Driving speeds for individual cars

4.3.1. Car speeds must be updated according to the road conditions.

- 4.3.1.1. On segments that have traffic incidents; drivers will travel at 0% of the speed limit.
 - Mapped to the Traffic Simulator.
- 4.3.1.2. For a car at segment i , let

$$n = \sum_{j=i-20}^{i+20} \text{segment}_j, \quad \text{where } \text{segment}_j = 1 \text{ iff it contains a car}$$

be $1 +$ the number of cars in the 20 contiguous segments that precede and succeed segment i . Then, the car at segment i will travel at $p = (1.07e^{-0.07n})\%$ of the speed limit. If segment i has fewer than 20 contiguous preceding or succeeding segments, then the value used by the summation for each non-existing segment j is 0. The author generated the equation for p by entering estimations of empirical observations into an Excel spreadsheet, using Excel to produce an exponential trend line, and altering that trend line until R^2 , the coefficient of determination, = 1.

- Mapped to the Traffic Simulator.

1 + Number of Surrounding Cars	Percentage of Speed Limit
1	100%
5	75%
10	53%
20	26%
40	7%

- 4.3.1.3. If a car travels from road segment i that has speed limit j to road segment $i + 1$ that has speed limit k , where $j \neq k$, then the car's change in speed from one segment to the next shall be instantaneous.
- Mapped to the Traffic Simulator.

4.4.Route

4.4.1. For each driver a list of waypoint nodes shall be generated. This list will detail a path that the driver will follow from his source to his destination.

- 4.4.1.1. Each driver shall follow a path in the directed graph, G , where a path is a sequence of vertices (nodes) $\langle v_0, v_1, v_2, \dots, v_n \rangle$ such that $\langle v_i, v_{i+1} \rangle$ for each i from 0 to $n-1$ is an edge in G .
- Mapped to the Traffic Simulator.
- 4.4.1.2. The waypoints shall be generated using Dijkstra's algorithm.
- Mapped to the Traffic Simulator.
- 4.4.1.3. The waypoints shall be stored in an array such that element 0 is the start node, element n is the end node, and elements 1 to $n-1$ are the nodes which, when followed in order, connect the edges of the graph. These edges, which are a mathematical construct, correspond to the path the driver will follow. This path is represented by (a subset of the) connected road segments displayed on the screen.
- Mapped to Data Storage.
- 4.4.1.4. At each node, the vehicle will read the allowable turns.
- Mapped to the map model.
- 4.4.1.5. In the case that the vehicle is unable to take a needed turn, it will travel to a node that is non-blocked and adjacent.
- Mapped to the Route generator.
- 4.4.1.6. Once the vehicle has selected the new node to travel to, it will generate a new route, using Dijkstra's algorithm, with the node it is traveling to as a new start node and with the original end node.
- Mapped to the Route generator.
- 4.4.1.7. If a vehicle cannot make a needed turn and there are no other reachable nodes, the vehicle stops and waits for the turn to become available.
- Mapped to the Route generator.

5. Architectural Diagram

