ESCUELA UNIVERSITARIA POLITÉCNICA DE TERUEL

Departamento de Informática e Ingeniería de Sistemas

Ingeniería Técnica en Informática de Gestión

"C4R: GENERACIÓN DE MODELOS DE MOVILIDAD PARA REDES DE VEHÍCULOS A PARTIR DE MAPAS REALES"

Trabajo Fin de Carrera

Autor: Alberto Pardo Calvo

Teruel, marzo 2011

ESCUELA UNIVERSITARIA POLITÉCNICA DE TERUEL

Departamento de Informática e Ingeniería de Sistemas
Ingeniería Técnica en Informática de Gestión

C4R: GENERACIÓN DE MODELOS DE MOVILIDAD PARA REDES DE VEHÍCULOS A PARTIR DE MAPAS REALES

Autor
Alberto Pardo Calvo

Directores
Francisco J. Martínez Domínguez
Piedad Garrido Picazo

TRIBUNAL

Presidente: _____

Secretario: Francisco J. Martínez Domínguez

Vocal: _____

CALIFICACIÓN:.....

FECHA: .....

Índice

Índice de Figuras

Índice de Tablas

Índice de Ecuaciones

Resumen

Hoy en día vivimos inmersos en una sociedad donde la movilidad, la comunicación y la información son imprescindibles. Las empresas y la sociedad demandan nuevos cambios en la conectividad a redes de datos. Como consecuencia, actualmente se están desarrollando nuevos sistemas de intercambio de información, como las Vehicular Adhoc Networks (VANETs), que permiten compartir información entre vehículos en movimiento. Para desarrollar este tipo de redes se suele recurrir a la simulación y construir modelos que representen el movimiento de vehículos de un modo fiel a la realidad, dado el alto coste que supondría realizar pruebas con vehículos reales.

Este documento aporta una visión innovadora en la generación de modelos de movilidad. Las soluciones adoptadas hasta ahora, consistían en introducir multitud de valores en cajas de texto. Sin embargo la solución que se propone en este trabajo consiste en permitir al usuario crear modelos de movilidad en pocos segundos y de un modo muy visual. De este modo, el usuario puede ver inmediatamente deficiencias en su modelo y corregirlas, con el consiguiente ahorro de tiempo. Además, otra característica importante de la solución, es que permite trabajar con mapas reales de cualquier parte del mundo, de modo que el realismo en cuanto a la topografía es máximo.

Palabras clave: Generación modelos de movilidad, VANETs, simulación, redes vehiculares, sistemas inteligentes de transporte.

---

1. Introducción

Las redes de vehículos son un campo apasionante y emergente del que se espera que sea capaz de mejorar la seguridad y confortabilidad de los pasajeros de los vehículos. Estas metas han hecho que grupos de investigación y empresas privadas empiecen a tener interés en este tipo de redes dedicando esfuerzos y dinero al desarrollo. A esto hay que sumarle que las tecnologías wireless han sufrido un gran avance y abaratamiento y ya son de uso común en ámbitos domésticos, lo que las hace interesantes para aplicaciones que han de ser usadas por un usuario final que no tiene por qué ser un experto en tecnologías de la información y la comunicación.

Las distintas aplicaciones que hacen uso de redes de vehículos se dividen en dos categorías: las relacionadas con la seguridad, y las relacionadas con la conectividad a Internet. Las relacionadas con la seguridad buscan aumentar la seguridad existente en los vehículos. Para ello se centran en cuestiones como avisos de peligro por accidente o avisos del estado de la calzada. Las relacionadas con la conectividad en cambio, buscan aumentar la confortabilidad de los pasajeros durante un viaje. Para conseguir sus objetivos estas aplicaciones se centran más en cuestiones como el acceso al correo electrónico o a contenidos Web. Existen otro tipo de aplicaciones que son difíciles de clasificar porque se encuentran a caballo entre las relacionadas con la seguridad y las relacionadas con la conectividad. Este tipo de aplicaciones ayudan en caso de avería, aliviando en la medida de lo posible el problema de una avería al conductor.

Para el desarrollo de las redes de vehículos se han dividido las posibles dificultades en diferentes partes. De una de esas partes nace el concepto de modelo de movilidad que más adelante será explicado en profundidad. Un modelo de movilidad es el patrón que sigue un objeto o individuo para moverse. Este trabajo se va a centrar en la generación de los modelos que siguen los vehículos al desplazarse. Estos modelos son necesarios para el desarrollo de las redes de vehículos puesto que permiten simular el comportamiento de los vehículos en laboratorio evitando de este modo tener que hacer pruebas con vehículos reales y evitando, también, el coste que esto conllevaría.

Para entender el concepto de modelo de movilidad es necesario entender una serie de factores que a continuación se van a explicar y que van a ayudar a entender qué papel juega un modelo de movilidad en las redes de vehículos y por qué son necesarios. El presente documento está organizado de la siguiente forma:

- **Introducción:** Pone en situación al lector para comprender la problemática de los modelos de movilidad.

- **Marco teórico:** Esta parte recoge los fundamentos en los que se basan los modelos de movilidad. El capítulo está dividido en varios puntos en los que se describe el funcionamiento de las redes móviles, en qué consisten los modelos de movilidad, su clasificación y algunos ejemplos de modelos de movilidad.

- **Estado del arte:** Este capítulo da una visión general de las herramientas existentes que sirven para realizar simulaciones con modelos de movilidad. Al

final del capítulo hay un cuadro comparativo en el que se ven claramente las posibilidades que aporta cada aplicación.

- **Análisis y diseño:** Este apartado contiene las especificaciones técnicas de la aplicación. En él se podrán encontrar el documento de especificación de requisitos, la planificación de desarrollo de la aplicación y una estimación de coste.

- **Implementación:** Trata detalladamente los pormenores del diseño, la implementación y la justificación de las distintas decisiones tomadas durante el desarrollo de la solución.

- **Manual de usuario:** Que contiene las instrucciones para llevar a cabo cualquier operación con la aplicación. Es recomendable leerlo antes de empezar a operar con el programa.

- **Conclusiones:** Apartado dedicado a la reflexión sobre el proyecto y a las conclusiones más importantes.

2. Marco teórico

Para llevar a cabo el desarrollo de cualquier tecnología es necesario un conocimiento previo que siente unas bases a partir de las cuales se pueda desarrollar dicha tecnología. Este capítulo tiene como fin dar a conocer el marco teórico en el cual se fundamenta el desarrollo de VANETs.

2.1 Introducción a las redes móviles

Para entender cómo es posible que se puedan establecer comunicaciones entre vehículos, primero se describirá brevemente el funcionamiento de algunos tipos de redes. Estas redes poseen una serie de características que las hacen muy interesantes para establecer comunicación entre vehículos.

2.1.1 Redes ad-hoc y MANETs

Las redes ad-hoc son un tipo de redes que no hacen uso de infraestructuras fijas, por lo que son interesantes para ser usadas en circunstancias en las que es especialmente complicado desplegar estas infraestructuras, por ejemplo en la selva o en ambientes bélicos. Este tipo de redes también son usadas en lugares donde las infraestructuras existentes han fallado y se requiere restablecer rápidamente la situación, por ejemplo en caso de terremotos o desastres naturales en general.

Históricamente las aplicaciones de las redes ad-hoc tenían un ámbito militar. En concreto estas aplicaciones tenían el objetivo de facilitar la comunicación en el campo de batalla puesto que la naturaleza de la guerra hace imposible la implantación de infraestructuras fijas. A pesar de que las primeras aplicaciones de redes con capacidad de movilidad fueran militares, las aplicaciones civiles han crecido de tal forma que ahora son mayoritarias. Tanto es el crecimiento en los últimos años que muchas empresas están interesadas en estas tecnologías. Prueba de ello son los nuevos protocolos que han surgido y que permiten el uso de redes Ad-hoc como por ejemplo Bluetooth, IEEE 802.11 o Hiperlan.

A continuación vamos a revisar las características que hacen a las redes ad-hoc tan interesantes [MK04].

- **Movilidad:** Las redes ad-hoc tienen la capacidad de que sus nodos pueden estar en movimiento y aún así son capaces de seguir funcionando. Cuando se da esta situación podemos hablar de mobility ad-hoc network (MANET).

- **Multisalto:** Los paquetes que viajan por redes ad-hoc pueden atravesar varios nodos para establecer comunicación entre el nodo origen y el nodo destino.

- **Autoorganización:** Las redes ad-hoc deben ser capaces de organizarse autónomamente ya que este tipo de redes puede tener sus nodos en

movimiento. Estas redes deben tener la capacidad de autoconfigurarse continuamente y mantener los datos actualizados para realizar correctamente el enrutamiento de datos. Además los nodos al tener que actuar como enrutadores, en ocasiones tienen que realizar tareas de descubrimiento de nuevos nodos y de mantenimiento de la red. Esta capacidad implica ser capaz de organizar direcciones, rutas, posiciones, etc. En consecuencia los nodos deben decidir a qué nodos hay que enviar datos, ya que en función de la disponibilidad los datos se envían dinámicamente. Además hay que tener en cuenta que los nodos pueden formar parte de la red para luego desaparecer. Esto es debido, entre otras causas, a que la transmisión inalámbrica tiene inexorablemente colisiones y cuanto más larga es la distancia a la que se quiere transmitir, más fácil es que haya colisiones. La principal consecuencia de estas colisiones es la pérdida de paquetes.

- **Conservación de energía:** Las redes ad-hoc deben de tener la capacidad de usar eficientemente la energía de los dispositivos. Esto es debido a que los nodos al ser móviles, tienen la necesidad de ser autónomos y esto se consigue mediante baterías. De modo que si se quiere mantener a los nodos funcionando durante mucho tiempo, es crítico hacer un uso eficiente de la energía de las baterías.

- **Escalabilidad:** Las redes ad-hoc pueden crecer en el orden de miles de nodos, por lo tanto hay que controlar el crecimiento de la red de algún modo. Esta es una variable importante a tener en cuenta en el diseño de MANET.

- **Seguridad:** Las redes ad-hoc son más vulnerables que las redes con infraestructura. Uno de los problemas de seguridad que presentan estas redes es que es posible realizar un ataque con pocas probabilidades de ser descubierto. Debido a esto son muy importantes los cifrados para mantener la seguridad y confidencialidad durante una transmisión.

2.1.2 VANET (Vehicular Ad-hoc NETwork)

Las VANETs son un caso especial de MANET y consisten en un conjunto de vehículos viajando que son capaces de comunicarse entre sí, sin ningún tipo de infraestructura fija. Como se ha explicado anteriormente, el objetivo principal de este tipo de redes es mejorar la seguridad y confortabilidad de los pasajeros. Para conseguirlo los vehículos que forman parte de una VANET se envían mensajes entre ellos formando así una red. Estas redes se crean arbitrariamente con vehículos que están geográficamente cerca. Como se acaba de señalar, durante la comunicación en una VANET, se comparten datos. Estos datos son de tipo informativo como puntos de interés, estaciones de servicio, atascos o peligros en la calzada. La comunicación en estas redes se produce de vehículo a vehículo (V2V) y también de vehículo a infraestructura (V2I) de modo que los vehículos son nodos de la red. Hay que añadir que los flujos de comunicación se mueven en el mismo sentido en el que se desplazan los vehículos [LA05] porque se asume que los vehículos se desplazan en la misma dirección a una velocidad parecida. Según este razonamiento es posible crear bloques con grupos de vehículos tal como se ve en la *Figura 1*. Los vehículos de un mismo

grupo tienen en común que circulan cerca unos de otros y mantienen la comunicación en un tiempo mínimo. A estos grupos de vehículos vamos a denominarlos, según la terminología MANET, como clusters.

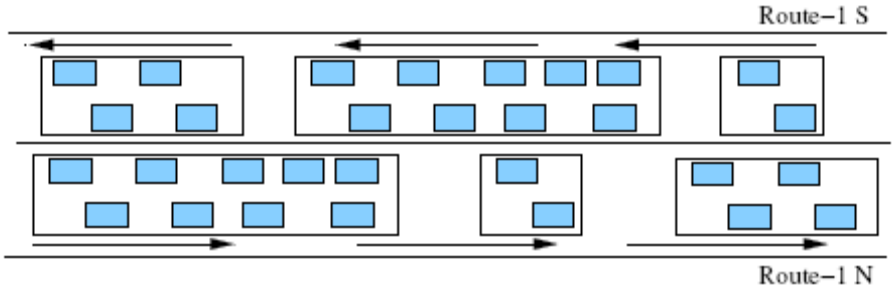

Figura 1 *Bloques de vehículos (clusters)* [LA05].

Una variable a tener en cuenta en el uso de las VANETs es la densidad de tráfico. Tanto un exceso como un defecto en la densidad pueden provocar un mal funcionamiento. Cuando hay poca densidad existen espacios vacíos que dificultan la repetición de mensajes. Un exceso en la densidad de vehículos provoca la ralentización en la entrega de mensajes. Otros problemas que hay que añadir a este tipo de redes son la alta velocidad que alcanzan los vehículos y la variabilidad del número de nodos de la red. Estos inconvenientes hacen que crear redes ad-hoc de forma espontánea en entornos vehiculares sea todo un reto.

No podemos olvidar tampoco la importancia de la seguridad en las VANETs. Es necesario prevenir algunos de los problemas que pueden surgir de la falta de seguridad, como la recepción de datos por destinatarios no autorizados, inyección de datos o ataques de denegación de servicio (DoS).

Existen algunas soluciones para evitar estos problemas de seguridad y tratar de que los sistemas funcionen correctamente. Por una parte el protocolo IEEE 802.11 ha incrementado la seguridad en la capa de conexión de datos introduciendo en el estándar el cifrado Wired Equivalent Privacy (WEP). Lamentablemente este tipo de cifrado hoy en día no es en absoluto seguro y ya se han añadido al protocolo dos extensiones la IEEE 802.11i y la 802.11x, para mejorar esta deficiencia. Otra forma de proteger las comunicaciones en una VANET es usar tres capas de seguridad con cifrado RSA, que todavía se puede considerar seguro.

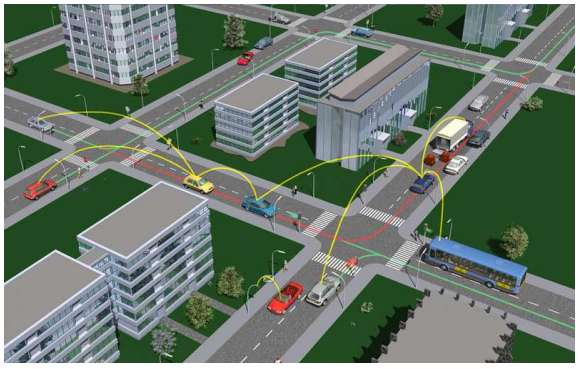

Figura 2 *Recreación de una VANET.*

Una vez vistas las dificultades que plantean las VANETs es fácil entender que establecer una comunicación en este tipo de redes es bastante complejo. Los nuevos protocolos, como el IEEE 802.11p [JD08, SE07], se han desarrollado para poder realizar la comunicación en VANETs adecuadamente.

2.2 Simulación de redes VANET

Probar el funcionamiento de redes VANET en un entorno real es extremadamente caro, puesto que requiere equipar vehículos con el hardware necesario y probarlo en entornos reales, por lo que es necesario buscar alternativas para desarrollar VANETs de una forma más barata e igualmente eficaz. La solución por la que se ha optado es realizar simulaciones mediante computador. Con este método se pueden realizar todas las pruebas que se consideren necesarias con un coste más reducido. La simulación necesita de una inversión inicial en investigación necesaria para desarrollar todas las herramientas encargadas de la simulación de VANETs. A esto hay que sumarle el coste que supone crear juegos de pruebas que permitan calibrar las herramientas para asegurarse que se reproduce la realidad lo más fielmente posible.

En las VANETs existen dos partes bien diferenciadas. Por un lado está el comportamiento de los vehículos y por otro el funcionamiento de la red que forman los vehículos. Para que funcione esta tecnología correctamente hay que enlazar estas dos partes y hacerlas cooperar. Para ello es importante describir el modo en el que se mueven los vehículos. A este efecto se han desarrollado numerosos modelos de movilidad, los cuales, serán descritos con profundidad más adelante. Una vez se conoce la forma en la que se mueven los nodos de la red, es necesario conocer las condiciones en las que llevará a cabo la transmisión de datos. Hay que tener en cuenta la topología por la que han de moverse los vehículos, ya que condiciona su movimiento y también limita la comunicación. Por lo tanto antes de construir herramientas que ayuden a simular cada uno de los elementos presentes en una VANET en un escenario real, tendremos que tener en cuenta todos estos factores.

El funcionamiento de un simulador VANET es esquemáticamente sencillo. Primero se crea un mapa por el cual han de circular los vehículos. Una vez se tiene el mapa, un generador de tráfico se encargará de la simulación del tráfico de los vehículos, que estará basado en uno o varios modelos de movilidad. Dicho generador de tráfico generará una traza de salida que contendrá los datos que indican cómo se moverán los distintos vehículos dentro del mapa. Finalmente, esa traza será utilizada, junto con la especificación de los datos que se transmitirán en las VANETs, por un simulador de red, que permitirá simular las comunicaciones inalámbricas que se producen entre los vehículos, la propagación de los mensajes que éstos envían, etc.

Ahora que se entiende el funcionamiento de los simuladores para VANETs, es posible explicar los distintos tipos de simuladores que se pueden encontrar en el mercado. Los simuladores pueden clasificarse en dos tipos distintos atendiendo al nivel de detalle con el que tratan la movilidad [HFB09]: macroscópicos y microscópicos. En los simuladores macroscópicos la simulación depende de factores como la densidad de vehículos o la velocidad, mientras que en los simuladores microscópicos se considera cada vehículo como una entidad independiente. Como ventaja los simuladores

macroscópicos simulan la realidad de un modo simple, y tienen un coste reducido en términos computacionales. Como desventaja, este tipo de simuladores no consigue simular la realidad con la precisión que sería deseable. Los simuladores microscópicos sin embargo, al tratar cada vehículo como una entidad independiente, consiguen un grado de realismo mayor que los simuladores macroscópicos, pero su coste computacional es elevado.

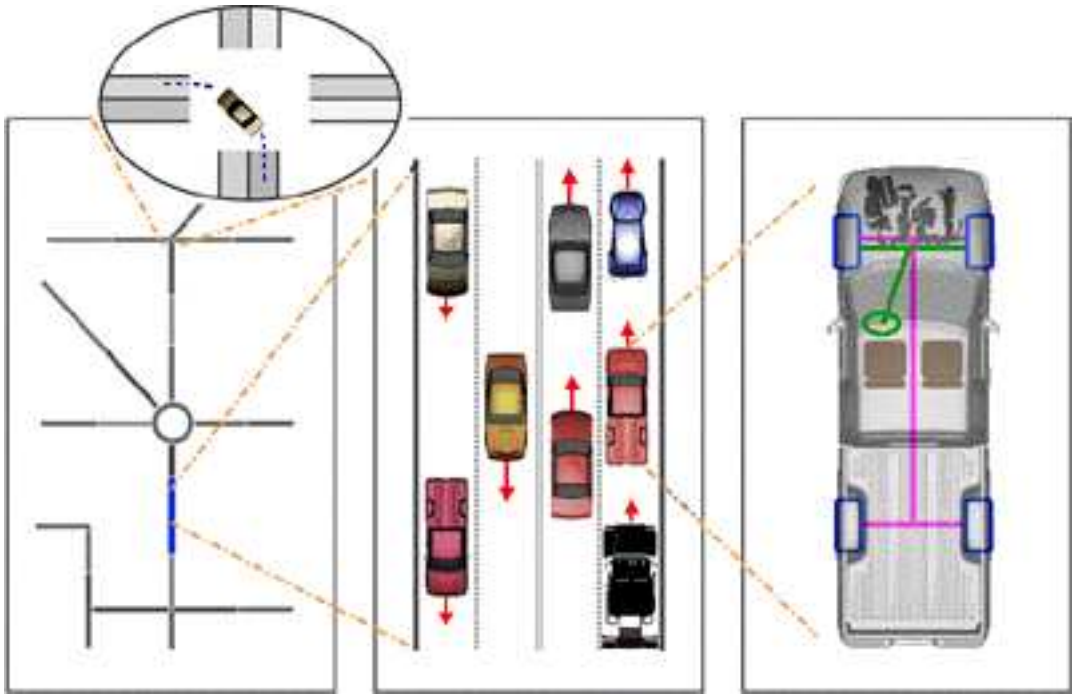

Figura 3 *Las diferentes granularidades en la simulación: de izquierda a derecha: macroscópica, microscópica, sub-microscópica (en el círculo: mesoscópica) [SUMO].*

2.3 Modelos de movilidad

Un modelo de movilidad es un conjunto de reglas que unidas entre sí describen el comportamiento espacial de cuerpos durante un periodo de tiempo. En este trabajo los modelos de movilidad se refieren exclusivamente a vehículos. Como se puede comprobar en la *Figura 4*, un modelo de movilidad es una pieza más en un sistema que pretende interconectar varios vehículos. En concreto, es la parte que se encarga de describir cómo tiene que desplazarse un vehículo y qué decisiones tiene que tomar.

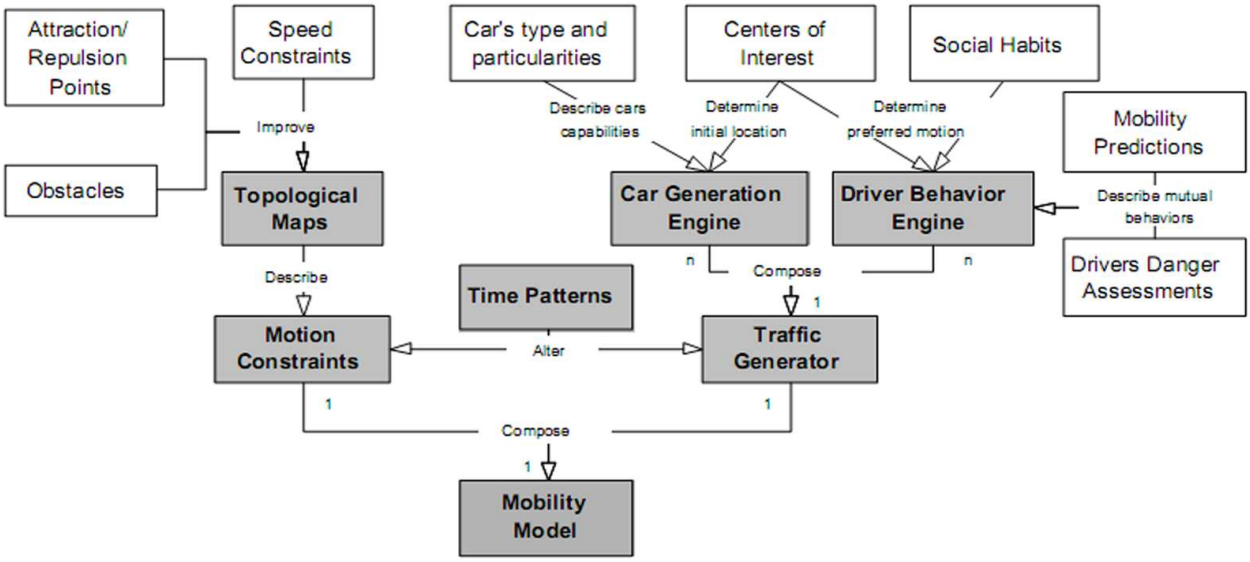

Figura 4 *Mapa conceptual de la generación de modelos de movilidad entre vehículos.* [EURECOM].

Los distintos modelos muchas veces eligen entre complejidad y detalle, es decir, eligen entre aproximarse a la realidad de una forma relativamente simple, analizando los principales factores, o aproximarse a la realidad con el máximo de detalle posible, estudiando todos los factores que parecen influir. En cualquier caso es importante que los modelos de movilidad sean fieles a la realidad. Para conseguirlo es necesario tener en cuenta aspectos básicos que influyen en la movilidad:

- **Trazado de calles:** Los vehículos tienen que moverse obligatoriamente dentro de las calles, por lo que la movilidad queda limitada a su trazado.

- **Tamaño de la manzana:** Una manzana puede ser considerada como la menor área rodeada por calles. El tamaño de la manzana determina el número de intersecciones en un área y por tanto establece la frecuencia con la que un vehículo tiene que parar en una intersección y la distancia a la que un vehículo vecino es capaz de detectar la transmisión de otro vehículo.

- **Mecanismos de control de tráfico:** Las formas más comunes de controlar las intersecciones son las señales de stop y los semáforos. Estos mecanismos crean clusters y colas de vehículos en las intersecciones. Como es lógico estas colas provocan una reducción de la media de la velocidad de los vehículos. La disminución de la velocidad favorece que haya menos cambios en el enrutamiento de datos de la red. Pero por otra parte los clusters de vehículos pueden afectar negativamente al funcionamiento de la red por suponer un aumento en la contención del canal.

- **Movimiento interdependiente de los vehículos:** A pesar de que cada vehículo puede tomar sus propias decisiones, éste se mueve influido por los vehículos que le rodean.

- **Velocidad media:** La velocidad de un vehículo establece cómo de rápido cambia su posición, lo que determina a la vez cómo cambia la topología de red. la media de velocidad varía en función de los límites de velocidad establecidos las características y el estado de la calzada, la aceleración y deceleración, y la topología de las calles. Del mismo modo, si hay pocas intersecciones los vehículos aceleran a velocidades mayores que si hay muchas intersecciones

2.4 Taxonomía de los modelos de movilidad

Para tratar los diferentes modelos de movilidad existentes es necesario realizar una clasificación de los mismos [FH08]. A continuación se detalla una posible clasificación usada en la simulación de redes. Esta clasificación está lejos de ser completa para modelos de movilidad. En realidad lo que se pretende con esta clasificación es dividir los modelos de movilidad para vehículos, que es lo realmente interesante en el presente trabajo.

- **Modelo estocástico:** Los vehículos, como se ha indicado antes, están restringidos a circular dentro de las calzadas. Este modelo mueve los vehículos eligiendo aleatoriamente el camino que tomarán. Asimismo, en este modelo, los vehículos viajan a una velocidad aleatoria. Este modelo es el más simple y el que ofrece unos resultados más alejados de la realidad. Es útil en el caso de tener que simular movimiento pero sin necesidad de realismo.

- **Modelo de corriente de tráfico** [EJT06]**:** La movilidad de vehículos es observada en este modelo a través de un punto de vista macroscópico. Su nombre hace referencia a que la corriente de vehículos es tratada como un fenómeno hidrológico. En este modelo la velocidad se calcula atendiendo a dos variables independientes: la distancia recorrida y la diferencia de velocidad entre un coche y aquél que lleva por delante.

- **Modelo de coches que siguen:** El comportamiento de cada coche es calculado en base a la posición, velocidad y aceleración de los coches que lo rodean. Este modelo se empezó a usar en los años 50 y es la forma más común de describir el flujo de vehículos de forma analítica.

- **Modelo de interacción de flujos:** Este modelo se caracteriza por la interacción recíproca y dinámica de los distintos flujos que convergen en una intersección. Es útil para explicar las incorporaciones en una carretera o intersecciones de ámbito urbano.

2.5 Descripción de algunos modelos de movilidad

En este capítulo se van a detallar algunos de los modelos de movilidad existentes y en qué consisten. Pero antes hay que definir la notación que se seguirá para explicar las distintas ecuaciones los modelos que a continuación se van a explicar. En cualquier caso la notación está resumida en la *Figura 5*.

- ***i***: indica el vehículo que se está investigando. Valores mayores que i indican los vehículos que están por delante y valores menores que *i* los vehículos que están por detrás.
- ***j***: funciona de forma análoga a la variable *i*, pero con la salvedad de que los vehículos se encuentran en otro carril.
- x_i: posición del vehículo.
- Δx_i: distancia existente entre los parachoques del vehículo en estudio y del vehículo precedente.
- Δx_{min}: distancia de seguridad.
- v_i :velocidad del vehículo *i*.
- v_{min} v_{max}: velocidad mínima y máxima respectivamente.
- ***a***: aceleración.
- ***b***: deceleración o frenado.
- ***t***: tiempo transcurrido.

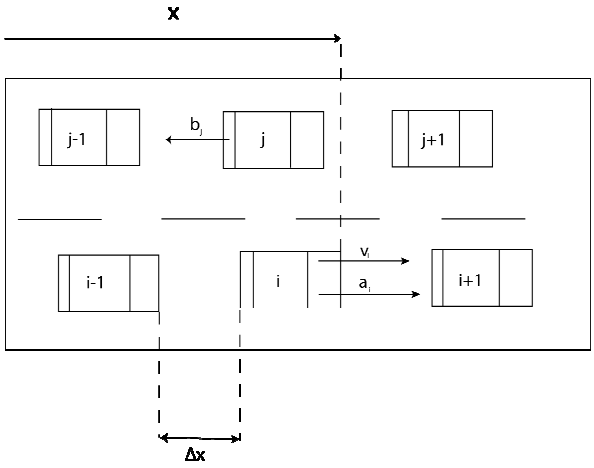

Figura 5 *Esquema de la notación seguida.*

2.5.1 Movimiento a velocidad constante

El modelo de movimiento a velocidad constante (CSM, Constant Speed Motion) es el típico ejemplo de modelo estocástico, por lo que el movimiento de los vehículos es generado bajo parámetros aleatorios. El movimiento de los coches está estructurado en trayectos, que es el movimiento entre dos puntos correspondientes a un origen y un destino. Al principio de un trayecto un vehículo elige su siguiente destino. Una vez elegido el destino el vehículo calcula la ruta basándose principalmente en el camino más corto posible. Este modelo tiene la desventaja de que calcula el movimiento de los vehículos de forma individual. Al no tener en cuenta los vehículos que le rodean,

pueden producirse solapamientos entre vehículos. También hay que señalar que este modelo explica bien el movimiento para vehículos aislados pero, sin embargo, falla para explicar el movimiento de grupos de vehículos.

Puesto que este modelo calcula la velocidad que han de llevar los vehículos a continuación según siguiente ecuación:

$$v_i = v_{\min} + \eta(v_{\max} - v_{\min})$$

Ecuación 1 *Modelo de movimiento a velocidad constante.*

Donde η es una variable aleatoria uniformemente distribuida comprendida entre 0 y 1. Este modelo incluye la posibilidad de realizar paradas durante la duración del viaje de un vehículo. Supone una forma rudimentaria de gestionar las paradas durante un trayecto.

2.5.2 Manhattan

Este modelo también se puede considerar estocástico. El modelo Manhattan añade un plus de complejidad al modelo anterior. En concreto, hace uso de una topología de rejilla de manera que el mapa consiste en calles que se cortan perpendicularmente entre sí. De este modo todas las manzanas son rectangulares. Los nodos en este caso cuando llegan a una intersección eligen aleatoriamente cuál será la siguiente calle que seguirán. Una mejora con respecto al modelo anterior es que los vehículos ya no se solapan. Por contra este modelo presenta como desventaja que falla en la gestión de las intersecciones. Al igual que en el modelo anterior la velocidad es importante pero en este caso depende de la aceleración. Las ecuaciones que describen la velocidad son las siguientes:

$$v_i(t+\Delta t) = \begin{cases} v_{i+1}(t) - a/2, & \text{Si } \Delta x_i(t) \le \Delta x_{\min} \\ \tilde{v}_i(t+\Delta t), & \text{En otro caso} \end{cases}$$

Ecuación 2 *Descripción de la velocidad de un vehículo según el modelo Manhattan.*

La *Ecuación 2* añade una restricción para evitar que los vehículos se solapen, lo que en este caso se considera una colisión con otro vehículo. Esta ecuación implica que si un vehículo ha alcanzado la distancia de seguridad mínima, entonces hay que frenar. Si no se aplica la forma estándar de calcular la velocidad en el modelo.

$$\tilde{v}_i(t+\Delta t) = \min\left\{\max\left\{v_i(t) + \eta a \Delta t, v_{\min}\right\}, v_{\max}\right\}$$

Ecuación 3 *Restricción sobre los límites de velocidad en el modelo Manhattan.*

La *Ecuación 3* propuesta para este modelo supone que la velocidad de un vehículo no puede ser más pequeña que el valor mínimo de v_{min} y tampoco puede superar el valor máximo de v_{max} durante cualquier instante del trayecto. Obsérvese que la *Ecuación 3* está incluida en la *Ecuación 2*, cuando un vehículo no ha alcanzado la distancia mínima de seguridad. En esta ecuación el parámetro η es la misma variable aleatoria uniforme introducida en la *Ecuación 1*. De este modo, el modelo Manhattan añade la aceleración de forma aleatoria a la descripción de la velocidad.

2.5.3 Movimiento de tráfico fluido

El modelo de tráfico fluido (FTM, Fluid Traffic Motion) utiliza una corriente de tráfico para aproximarse desde un nivel microscópico a la realidad. La velocidad se describe como una función monótona decreciente de la densidad de vehículos, forzando a ajustar la velocidad cuando los niveles de congestión alcanzan un nivel crítico. La ecuación que describe la velocidad en este modelo es:

$$v_i(t+\Delta t) = \max\left\{v_{\min}, v_{\max}\left(1-\frac{n/l}{k_{jam}}\right)\right\}$$

Ecuación 4 *Descripción de la velocidad en el modelo FTM.*

Donde k_{jam} es la densidad de vehículos para la que se detecta un atasco, n es el número de vehículos en la calle i y l es la longitud de la calle. Puesto que n/l es la densidad actual de la calle, los vehículos que circulan por calles saturadas se ven obligados a disminuir su marcha. Por otra parte los vehículos que circulan por calles con poco tráfico pueden acelerar hasta el máximo de velocidad permitido.

2.5.4 Modelo Wagner

La mayoría de los modelos asumen que los conductores reaccionan continuamente a los cambios del entorno. El modelo Wagner sostiene que esta visión es incompleta y que pasa por alto dos factores. El primero es que las personas conducen anticipándose y el segundo que el control de los humanos durante la conducción no es continuo sino discreto en el tiempo [W06]. A cada uno de estos puntos en el tiempo en los que el conductor realiza alguna acción son denominados como puntos de acción. Muchos de los puntos de acción suceden con mayor o menor motivo por lo que son considerados como fenómenos aleatorios. Las dos condiciones pueden resumirse en las siguientes ecuaciones:

$$d(v+a_{opt}\tau)+v\tau+\frac{1}{2}a_{opt}\tau^2 \le d(V)+g$$

Ecuación 5 *Condición para conducir de modo seguro*

Donde v es la velocidad del vehículo a_{opt} es la aceleración óptima para ese intervalo de tiempo, τ es el tiempo de reacción del conductor, V la velocidad del vehículo precedente y g la distancia de seguridad para evitar una colisión. Para cada punto de acción debe satisfacerse la *Ecuación 5* para que la conducción sea considerada como segura. Lo que realmente describe esta ecuación es que para cada intervalo de tiempo de reacción la aceleración debe ser tal que el espacio que se recorre durante ese tiempo no debe ser mayor que el espacio que separa el vehículo del conductor con el vehículo precedente.

$$a_{opt} = -\frac{v}{t} - \frac{b}{2} + \sqrt{\left(\frac{v}{t} - \frac{b}{2}\right)^2 + \frac{2bg + V^2 - v^2}{\tau^2}}$$

Ecuación 6 *Aceleración óptima para conducir de modo seguro*

Donde b es la deceleración del vehículo y t el tiempo. Esta ecuación determina la aceleración óptima para un t cualquiera durante un intervalo τ. Esta es la apelación máxima que se puede alcanzar sin comprometer la seguridad. Nótese que este modelo enfatiza bastante en la aceleración debido a que es la única variable que el conductor puede modificar.

2.5.5 Modelo Kerner

También conocido como teoría de las tres fases del tráfico [K98]. Se centra principalmente en las congestiones de vehículos. Para explicarlas define tres estados frente a los dos estados que se usaban clásicamente:

- **Flujo libre:** se da cuando la tasa de vehículos por tiempo es menor que la densidad de vehículos. Si no se cumple esta condición se considera que el tráfico está congestionado. Cuando hay flujo libre, los vehículos pueden circular sin problemas de congestión.
- **Flujo sincronizado:** se da en el tráfico congestionado y se caracteriza por tener una densidad de tráfico que permite viajar con fluidez pero que en ocasiones puede obligar a reducir la marcha. La punta del flujo de tráfico se comporta como en flujo libre pero a una determinada distancia de ésta se crea un cuello de botella.
- **Amplio congestionamiento:** es un caso de tráfico congestionado que se da cuando la densidad de tráfico es extremadamente alta y la velocidad a la que circulan los vehículos es casi nula. Este caso se caracteriza porque en la parte delantera de la congestión los vehículos aceleran cambiando su estado a flujo libre y los vehículos de detrás van frenando cambiando su estado a amplio congestionamiento.

Este modelo también explica cómo se da la transición entre los diversos estados y sus características.

2.5.6 Modelo Krauß

Es un modelo de movilidad basado en los vehículos predecesores. La idea es que los vehículos deben ajustar su velocidad a la del vehículo predecesor para evitar colisiones [KW97]. La velocidad está definida según la siguiente ecuación:

$$v(t+1) = v_1(t) + \frac{g(t) - v_1(t)}{\tau_b(t) + 1} - \eta(t)$$

Ecuación 7 *Velocidad según el modelo Krauß*

Donde v es la velocidad, t el tiempo, τ el tiempo de reacción y η una variable aleatoria entre 0 y 1. En esta ecuación se ve claramente que la velocidad está restringida continuamente por los vehículos de alrededor (apreciable en los parámetros v_1 y $g(t)$).

Este modelo ha sido actualizado y es ofrecido por SUMO además del modelo Krauß original. La nueva actualización contempla el tráfico en múltiples carriles.

2.5.7 Modelo de conducción inteligente

El modelo de conducción inteligente (IDM, Intelligent Driver Model) es una mejora de los anteriores. IDM es un modelo determinista, en el cual la aceleración de un vehículo depende de la suya propia, de la de los vehículos que le rodean y del espacio que tiene con el vehículo precedente [TH99]. IDM propone además la posibilidad de simular el comportamiento de conductores distintos. Para simular este comportamiento vamos a suponer tres tipos de conductores [KTH08]: agresivo, normal y tímido. Asimismo se consideran las diferencias entre conductores de coches y camiones. Este tipo de comportamiento se rige por la siguiente ecuación:

$$\delta = x(v, \Delta v) = \Delta x_{\min} + vT + \frac{v\Delta v}{2\sqrt{ab}}$$

Ecuación 8 *Distancia entre vehículos según el modelo IDM*

La *Ecuación 8* hace referencia a la distancia existente entre vehículos tomando para su cálculo los siguientes parámetros:

- **Velocidad deseada *v*:** Es la velocidad máxima que un conductor desea alcanzar en condiciones en las que la densidad de tráfico permite circular con rapidez. Un conductor normal elegirá la velocidad máxima legal, mientras que un conductor agresivo sobrepasará esta velocidad y finalmente, un conducto tímido circulará por debajo de esta velocidad. Los camiones por su parte, en la mayor parte de países, se ven sometidos a fuertes restricciones de velocidad, por lo que suelen comportarse como conductores normales.

- **Tiempo espacial *T* deseado:** Este factor se refiere a distancia *vT* preferida por un conductor a una velocidad *v*. Es decir, la distancia que hay entre un vehículo y otro a una velocidad *v*. Este factor suele tener un valor típico de 1.4 s.

- **Distancia mínima entre parachoques Δx_{min}:** Cuanto más agresivo es un conductor menor es la distancia entra parachoques que deja. Los conductores de camiones, en cambio, dejan una mayor distancia debido a la distancia que necesitan estos vehículos para frenar.

- **Aceleración deseada *a*:** Describe el comportamiento de un conductor durante las aceleraciones. Como es de esperar, cuanto más agresivo es un conductor mayor es la aceleración aplicada. Hay que tener en cuenta que esta aceleración está supeditada a restricciones físicas, por lo que la aceleración en un camión necesariamente será, en general, menor que la de un coche.

- **Frenado confortable *b*:** Describe la deceleración de un vehículo al aproximarse a un obstáculo en la calzada. Los conductores agresivos se aproximan a gran velocidad, por lo que es necesario que la deceleración aplicada sea superior que en un conductor tímido que se aproxima lentamente. Como se puede comprobar, cuanto más agresivo es un conductor, mayor es el valor de deceleración.

Parámetros IDM	**Normal**	**Tímido**	**Agresivo**	**Camión**
Velocidad deseada v_0 (en Km/h)	120	100	140	85
Tiempo espacial *T* deseado (en s.)	1.5	1.8	1.0	2.0
Distancia de seguridad s_0 (en m.)	2.0	4.0	1.0	4.0
Aceleración *a* máxima (en m/s^2)	1.4	1.0	2.0	0.7
Deceleración *b* (en m/s^2)	2.0	1.0	3.0	2.0

Tabla 1 *Parámetros del modelo IDM para simular 3 clases de comportamiento de conductores y de conductores de camión.* [KTH08]

Es necesario aclarar que esta notación puede cambiar en algunas publicaciones. Por ejemplo SUMO trata especialmente este modelo, llamando al tiempo espacial *HeadWay* y a la distancia de seguridad *MinGap*. Estos cambios en la notación no tienen mayor efecto que el cambio de nombre tal como se puede apreciar en la *Figura 6*.

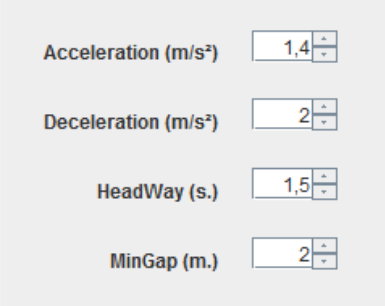

Figura 6 *Notación usada por SUMO para el modelo IDM.*

Dejando de lado el tema de las notaciones y haciendo más énfasis en las variables del modelo, se observa que el modelo IDM se caracteriza por describir el comportamiento de un vehículo a través de su aceleración instantánea por medio de la siguiente ecuación:

$$\frac{dv_i(t)}{dt} = a\left[1-\left(\frac{v_i(t)}{v_{des}}\right)^4-\left(\frac{\delta}{\Delta x_i(t)}\right)^2\right]$$

Ecuación 9 *Aceleración de un vehículo en un tiempo t determinado*

La velocidad deseada v_{des} tiene un valor diferente para cada tipo de conductor y está distribuida uniformemente entre los valores [v_{min}, v_{max}]. De la *Ecuación 9*, hay que destacar que la aceleración va aumentando hasta que alcanza la velocidad deseada (v_{max}) en una calle descrita mediante [1-($v_i(t)/v_{max}$)^4] y la deceleración inducida por el coche precedente es (δ / $\Delta x_i(t)$)^2. Por otra parte la distancia dinámica (δ) determina el frenado, y depende de la velocidad del vehículo tal como se muestra en la *Ecuación 8*.

2.5.8 IDM con gestión en las intersecciones

El IDM con gestión en las intersecciones (IDM-IM, IDM with Intersection Management) Es una mejora del modelo IDM, al que se ha añadido la capacidad de gestionar las intersecciones haciendo énfasis en la interacción vehículo a vehículo. El modelo IDM-IM es capaz de gestionar intersecciones controladas mediante señales de stop y mediante semáforos. La forma de gestionar intersecciones viene dada por la siguiente ecuación:

$$\Delta x_i(t) = \Delta x_{stop}(t) - S, \quad v_{i+1}(t) = 0$$

Ecuación 10 *Gestión de intersecciones según el modelo IDM-IM*

$\Delta x_{stop}(t)$ es la distancia actual hasta la intersección y S es el margen de seguridad. Entendiendo que el margen de seguridad es el espacio que hay desde el centro de la intersección al punto donde se ha de detener el vehículo.

2.5.9 IDM con cambio de carril

El IDM con cambio de carril (IDM-LC, IDM with Lane Changing) añade al modelo IDM la capacidad de que los vehículos adelanten a otros vehículos. Este modelo está basado en el MOBIL changing model, que considera que un vehículo puede cambiar de carril cuando es capaz de acelerar más que el vehículo que le precede. [TH02]. Las siguientes ecuaciones describen este proceso:

$$\frac{dv_i'(t)}{dt} - \frac{dv_i(t)}{dt} \geq p\left(\frac{dv_{j-1}(t)}{dt} - \frac{dv'_{j-1}(t)}{dt}\right) + a_{thr}$$

Ecuación 11 *Descripción de la maniobra de cambio de carril según IDM-LC*

$$\frac{dv'_{j-1}(t)}{dt} > -b_{safe}$$

Ecuación 12 *Frenado de seguridad para el vehículo de detrás*

Donde los caracteres marcados con «'» son los valores que implican un posible adelantamiento. En la primera inecuación en la parte izquierda representa la ventaja para poder adelantar, mientras que la parte de la derecha representa las desventajas para llevar a cabo la misma maniobra. El factor *p* es el factor de amabilidad del conductor, mientras que el umbral de aceleración (a_{thr}) previene el fenómeno de salto de carril en condiciones límite y puede ser ajustada para diferenciar movimientos a izquierda o a derecha [FH08]. La *Ecuación 12* además añade condiciones extra de seguridad para el coche de detrás en el nuevo carril. De hecho el parámetro b_{safe} es el valor de la aceleración del vehículo de detrás.

2.6 Conclusiones

Como se ha podido comprobar, en las VANETs existen dos partes bien diferenciadas. Por un lado, tenemos la movilidad de los vehículos, y por otro tenemos la red que forman los vehículos. Todo esto está encuadrado en un mapa que será por el que circulan los vehículos. Como es de suponer, llevar a cabo la comunicación entre varios vehículos es sumamente complejo por lo que se dividen cada uno de los problemas que surgen y se analizan por separado. Uno de estos problemas es el de los modelos de movilidad. Estos modelos pretenden describir la realidad de un modo preciso, pero lamentablemente aún no son un fiel reflejo de la realidad. Por este motivo, el trabajo futuro está en aumentar el nivel de detalle de estos modelos de movilidad. Para conseguirlo es necesario identificar y describir correctamente todos los factores fundamentales que afectan a la movilidad. Es necesario recalcar la importancia que tiene el desarrollar modelos de movilidad realistas ya que el resultado de estos modelos sirve para simular las redes de vehículos. Por lo tanto, si se crean modelos alejados de la realidad, por una cuestión de acumulación de errores, se obtendrán resultados muy alejados de la realidad a la hora de simular VANETs.

3. Estado del arte

En esta sección se presentarán cuáles son las herramientas que permiten generar y hacer uso de los modelos de movilidad. Primero se hará una breve introducción para explicar las características de estas herramientas. A continuación hay una descripción de algunas de las aplicaciones existentes hoy en día. Finalmente, se presenta un cuadro comparativo de las herramientas descritas.

3.1 Simuladores

Debido al creciente interés creado sobre las VANETs, en los últimos años se han desarrollado multitud de herramientas que ofrecen la posibilidad de generar modelos de movilidad. Estas herramientas dan la posibilidad de generar una simulación con varios tipos de vehículos. Es habitual que el resultado de esta simulación sirva para simular las redes que formarían los vehículos con un simulador de redes. De esta forma con este conjunto de herramientas, podemos simular el funcionamiento de una VANET en un entorno bastante realista.

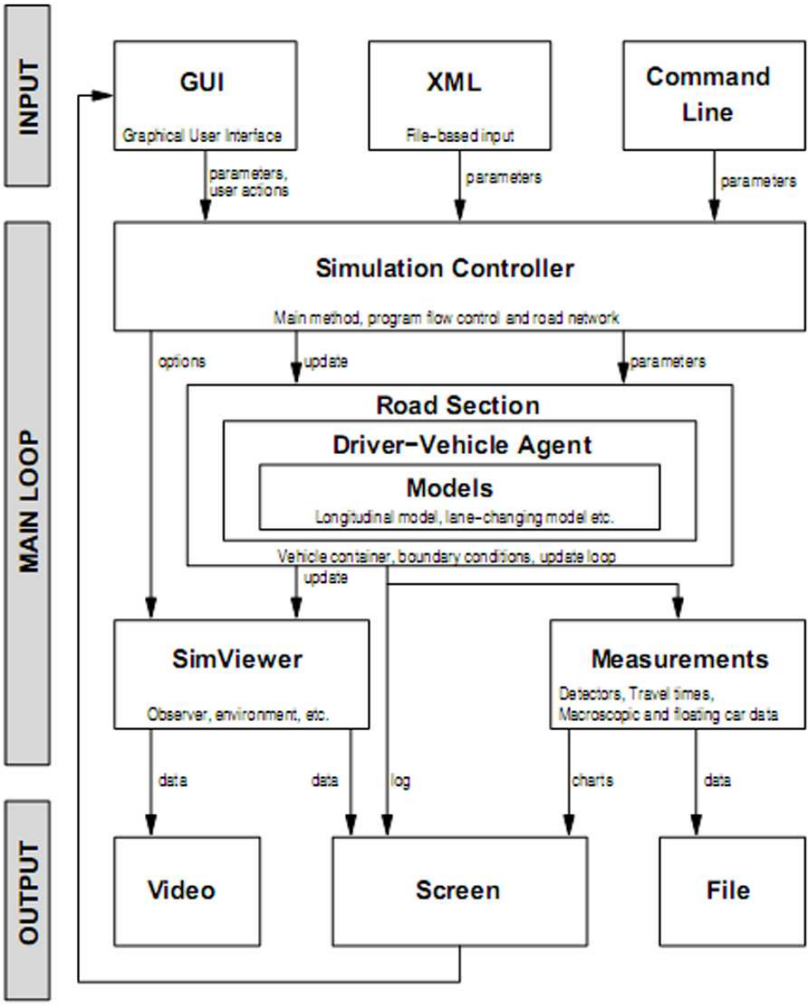

Figura 7 *Diagrama de la arquitectura de un simulador genérico* [KTH08].

La simulación por computador permite manejar un gran volumen de información, simplificando la tarea que supondría determinar el movimiento de los vehículos sobre un mapa. Simulando se consigue ver de una forma rápida y barata el resultado que ofrece un determinado modelo de movilidad. Como es lógico, estos simuladores necesitan de una entrada de datos. Para la introducción de estos datos se puede recurrir a ficheros codificados en XML, a la introducción de datos mediante consola, o mediante una interfaz gráfica de usuario (GUI, Graphical User Interface). Los simuladores tienen un controlador que se encarga de vigilar el funcionamiento del simulador y de las operaciones introducidas por los usuarios [KTH08]. Este control centralizado actualiza la información de los vehículos existentes en el mapa dando como resultado los movimientos de los vehículos. Además, este sistema de control también se va a encargar de actualizar la representación gráfica y de mostrar el resultado de las distintas medidas llevadas a cabo durante la simulación, tanto a nivel macroscópico como a nivel microscópico. Hay varias formas de presentar el resultado de estas simulaciones. La salida de datos puede ser mediante un archivo XML, o como se muestra en la *Figura 7*, mediante la representación gráfica del resultado ya sea en 2D o en 3D. La representación gráfica muestra el resultado de un modo más claro e intuitivo. Este método, además, permite ver en tiempo real el resultado global de la simulación.

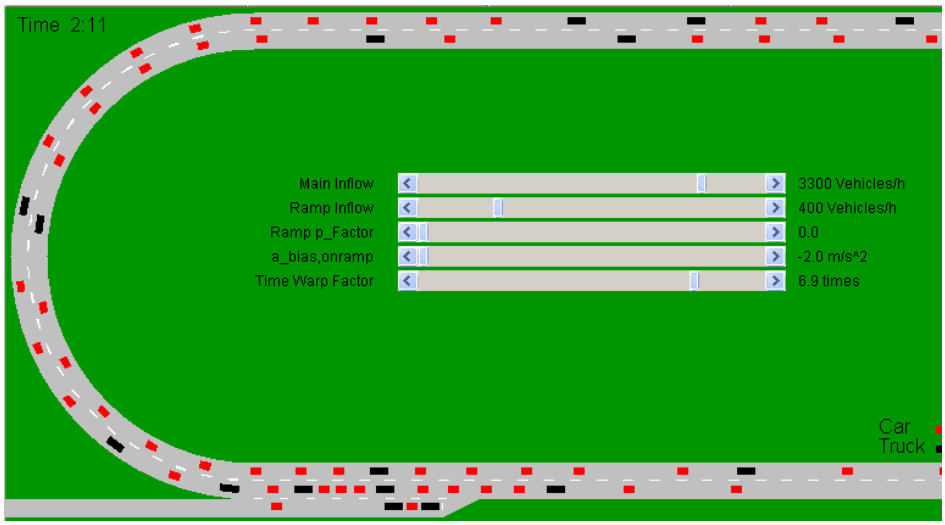

Figura 8 *Visualización en 2D de una simulación. Esta imagen corresponde a un applet disponible en:* ***http://www.traffic-simulation.de/.***

En la actualidad existen muchos simuladores que son clasificados por la mayoría de los autores entre macroscópicos y microscópicos, utilizando la misma terminología que se aplica a los modelos de movilidad

Los simuladores macroscópicos ponen especial atención a características relacionadas con la topología para construir las simulaciones. En concreto los simuladores macroscópicos se atienen a los siguientes puntos para llevar a cabo las simulaciones [HFB06]:

- **Mapa:** Es un punto crucial. La topología de las calles afecta severamente al resultado de la simulación ya que los vehículos tienen limitado su movimiento a las calles. Dentro de los mapas también existen factores determinantes como el número de intersecciones o la longitud de una calle, que pueden hacer que varíe

la topología de red con la consecuente variación de los resultados de la simulación. Atendiendo a la topología de los mapas podemos distinguir los siguientes tipos:

- **Personalizado**: En este tipo de mapas el usuario crea el mapa a la medida de sus necesidades diseñando su propia topología. Se usan para recrear situaciones muy concretas. El problema que presentan estos mapas es su complejidad y el coste de tiempo asociado a su creación.

- **Aleatorio:** Estos mapas se crean aleatoriamente de acuerdo a ciertos patrones predeterminados. Algunas de las formas que suelen tener estos mapas son: forma de rejilla o Manhattan, forma de tela de araña o Spider y mapas con polígonos de Voronoi generados aleatoriamente.

- **Mapas reales:** Son mapas reales obtenidos de bases de datos. Pueden ser obtenidos de de diversos sitios como por ejemplo la base de datos TIGER [TIGER] y OpenStreetMap [OSM].

- **Posición inicial y destino:** Estas posiciones se generan aleatoriamente teniendo en cuenta los denominados puntos de atracción y de repulsión. Los puntos de atracción son lugares donde hay mucha afluencia de vehículos, por ejemplo el centro urbano o un centro comercial. Por el contrario, un punto de repulsión es un lugar del mapa del cual los vehículos se alejan, por ejemplo un barrio residencial a la hora de ir a trabajar. Conociendo estos puntos podemos determinar donde es más probable que un coche quiera iniciar la marcha y a donde quiere ir. Es más probable que un vehículo inicie la marcha en un punto de repulsión para ir a un punto de atracción. Para la generación de estos puntos es necesario tener en cuenta la hora del día y el día de la semana, puesto que los puntos de atracción y repulsión varían en función de estos parámetros.

- **Generación del viaje:** Del mismo modo que se generan los puntos de posición inicial y destino, es necesario generar el camino que seguirá un vehículo para llegar del punto inicial al punto destino. Es común que este camino se genere de forma aleatoria pero aunque así sea, es necesario tener en cuenta las preferencias de los conductores para realizar el viaje y posibles nodos intermedios durante un viaje.

- **Velocidad:** Para representar velocidad de a manera más realista posible, debería ser uniforme, acomodarse a la situación del tráfico y variar poco a poco. En este aspecto hay que tener en cuenta que un vehículo cuando arranca la marcha no pasa de 0 a 50 Km/h instantáneamente, del mismo modo que la frenada no es instantánea, sino progresiva.

Frente a los simuladores macroscópicos, los simuladores microscópicos se basan en el comportamiento individual de cada conductor. De esta forma, la suma de los comportamientos de todos los conductores conformará el resultado de la simulación. Los factores en los que se fijan este tipo de simuladores son:

- **Cambio de carril:** La posibilidad de que un vehículo pueda adelantar a otro puede hacer cambiar la forma en la que se mueven los vehículos.

- **Patrones de movilidad humanos:** La forma de conducir de cada individuo varía ligeramente debido a las diferentes preferencias de cada uno. En conjunto, estas preferencias, pueden hacer que los resultados de la simulación no sean los esperados.

- **Comportamiento en los cruces:** Los conductores cuando están en un cruce no se comportan todos del mismo modo. Por ejemplo, al llegar a un stop un conductor puede esperar a que pase un vehículo, mientras que otro conductor en la misma situación puede considerar que la distancia a la que está el vehículo es suficiente como para que pueda pasar.

Hasta ahora se han visto los factores que implementan los simuladores para efectuar una simulación realista, pero esto no es suficiente. Con el fin de que las herramientas que se encargan de simular VANETs, además de realistas, sean útiles para la comunidad de investigadores, es altamente recomendable que las aplicaciones cumplan los siguientes puntos [MTB07]:

1. Los simuladores deben ser software libre para permitir que la comunidad de investigadores critique la validez de un modelo y su implementación.

2. El código fuente debe ser claro y estar suficientemente documentado para permitir realizar la tarea antes descrita con facilidad.

3. La estructura de los simuladores debe ser modular para poder analizar individualmente cada pieza que forma parte del proceso de simulación.

Además de estas consideraciones, existen otras a tener en cuenta a la hora de desarrollar una herramienta. Una de ellas es el lenguaje. Por ejemplo, si se desarrolla una herramienta en lenguaje C, hay que tener en cuenta que es un lenguaje difícil de depurar y con un código que requiere de bastante documentación para ser entendido perfectamente. Mientras que si se elige Java, podemos superar estos inconvenientes que presenta el lenguaje C. Esto no quiere decir que los simuladores escritos en Java sean mejores *per se*, sino que hay que considerar todos los factores que pueden influir tanto en el funcionamiento de la simulación en sí, como en la legibilidad de la aplicación.

Una vez descritas las características generales que deben tener los generadores de movilidad, se van a comparar algunos de los existentes añadiendo una pequeña descripción de cada uno de ellos.

3.2 SUMO (Simulation of Urban MObility)

Esta herramienta llamada SUMO es distribuida bajo licencia GPL y es desarrollada por el Institute of Transportation Systems con el objetivo de dar apoyo a la comunidad de investigadores en el área del tráfico [SUMO]. Para conseguir que sea una aplicación útil para la investigación se ha puesto especial cuidado en que sea lo más rápida y portable posible.

SUMO es un simulador microscópico que ofrece usar una rápida GUI de OpenGL o bien la línea de comandos. A continuación se describen algunas de sus principales características relacionadas con la simulación. Con SUMO es posible simular varios tipos de vehículos, calles de varios carriles con cambios y cruces de calles de varios tipos. SUMO ofrece gestionar redes en más de 10.000 calles, una velocidad de simulación rápida (por encima de 100.000 actualizaciones/segundo de vehículos en una máquina de 1GHz), interoperabilidad con otra aplicación en tiempo de ejecución usando TraCI (una extensión de SUMO). En cuanto a la simulación de redes hay que destacar que es posible importar muchos formatos de redes como VISUM, Vissim, Shapefiles, OSM, Tiger, RoboCup, XML-Descriptions. En cuanto a las rutas que usan los vehículos para moverse por el mapa, SUMO asigna a cada vehículo una ruta propia y también permite que sean asignadas por el usuario de forma dinámica. Aparte de esto, hay que destacar de SUMO la alta interoperabilidad que ofrece a través de XML y la alta portabilidad debido a que sólo se usa C++ estándar y librerías portables, permitiendo versiones para Windows y Linux.

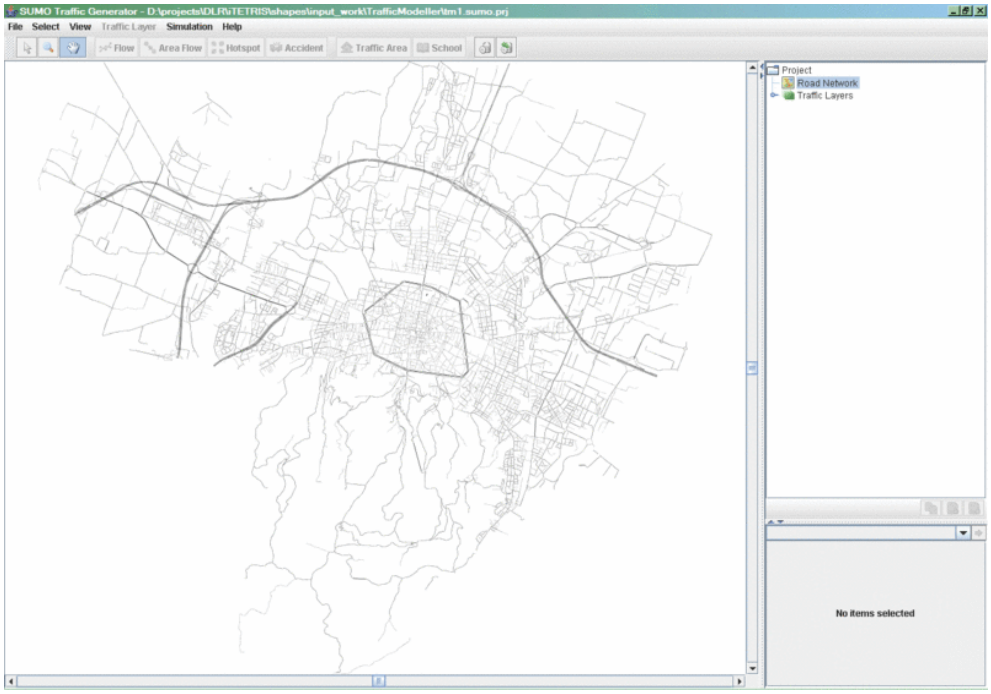

Figura 9 *Interfaz de SUMO.*

3.3 MOVE (MObility model generator for VEhicular networks)

[MOVE] es una herramienta desarrollada con el objetivo de facilitar a los usuarios la tarea de generar modelos de movilidad realistas para simulaciones de VANETs. Esta es una herramienta que trabaja con SUMO, de forma que la salida que ofrece es una traza que puede ser usada por un simulador de redes como NS-2 o QualNet. Además gracias a la interfaz gráfica que tiene MOVE es posible generar rápidamente escenarios realistas sin la molestia de tener que escribir scripts para simular y sin tener que conocer el funcionamiento interno de ninguna de las aplicaciones con las que trabaja.

MOVE se compone de dos partes bien diferenciadas. La primera trata de la simulación de redes y la otra parte trata de la generación de mapas y modelos de movilidad. En cuanto a la simulación de redes permite configurar la salida de datos para los simuladores de redes anteriormente citados. Sobre los modelos de movilidad se puede destacar que esta aplicación permite crear a través de su interfaz varios tipos de mapas. Puede crearse un mapa personalizado, añadiendo cada uno de los nodos que componen el mapa y después la información de las calles que son las líneas que unirán los nodos. MOVE también tiene un modo que permite generar un mapa de forma automática a partir de unos datos que se piden a través de la interfaz. Estos mapas pueden ser de 3 tipos: rejilla, tela de araña o totalmente aleatorio. Otra posibilidad que ofrece la aplicación, es conseguir mapas a través de archivos provenientes de TIGER. Una vez que tenemos los mapas, hay que simular los vehículos. Para ello MOVE permite generar automáticamente el movimiento de los vehículos o bien permite hacerlo manualmente, definiendo por dónde irá el vehículo y cómo se comportará en los cruces. Además, se pueden simular varios tipos de vehículos tales como autobuses urbanos y similares, estableciendo un horario y puntos de parada.

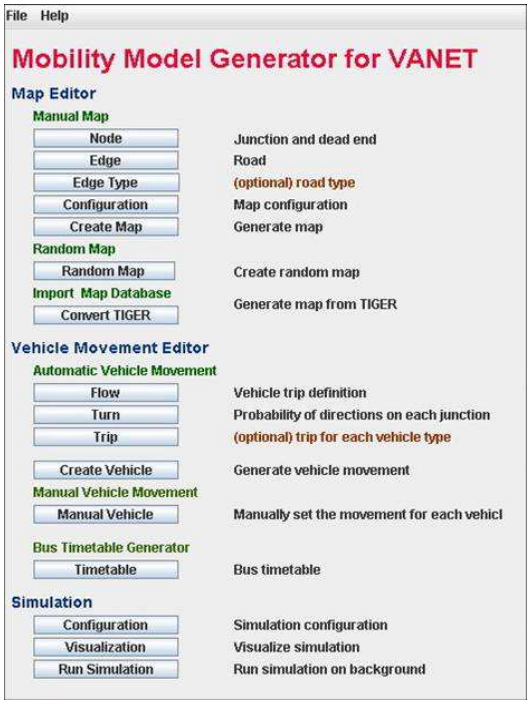

Figura 10 *Interfaz de MOVE.*

3.4 Trans (Traffic and Network Simulator environment)

Otra herramienta interesante es Trans. De esta herramienta se puede resaltar que posee una interfaz gráfica que trabaja con un simulador de tráfico y un simulador de redes (SUMO y NS-2) para generar simulaciones realistas de VANETs [TRANS]. Trans permite que la información que se intercambia durante una simulación pueda modificar el comportamiento de los vehículos durante el transcurso de la simulación, lo que le da un plus de realismo. Trans tiene como características que permite que su salida sirva para NS-2 a partir de mapas importados de TIGER. Esta aplicación aporta más funcionalidades cuando se trabaja con TIGER puesto que permite ciertas manipulaciones sobre el mapa y también permite ver la visualización de la simulación con Google Earth. Aparte, hace uso de la extensión de SUMO TraCI para unir SUMO con NS-2, pero no funciona con la última versión de SUMO. Otras características de Trans es que puede simular más de 3000 vehículos y permite simular incidentes en la calzada, como por ejemplo accidentes.

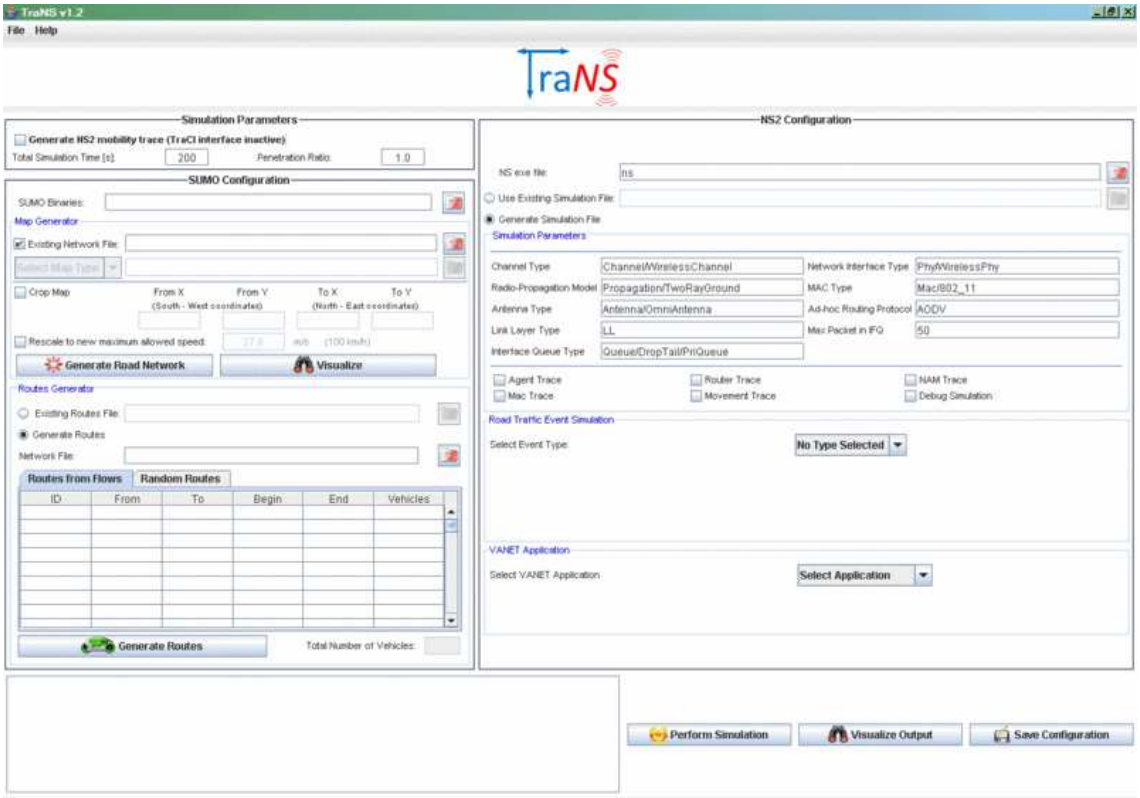

Figura 11 *Interfaz de Trans.*

3.5 STRAW (Street Random Waypoint)

Otra herramienta para la generación de modelos de movilidad es STRAW. Está programada en Java y basada en SWANS (Scalable Wireless Ad-hoc Network Simulator), un simulador de redes gratuito comparable a ns2 y GloMoSim. STRAW extrae los mapas a partir de la base de datos TIGER [STRAW]. Este simulador tiene como característica que realiza la gestión de cruces, tanto con semáforos como con señales de tráfico de una forma avanzada. Gracias a esto, la simulación en los cruces alcanza un nivel bastante realista. De STRAW hay que subrayar que define unas restricciones de movilidad muy precisas, así como un motor que genera tráfico muy realista. Uno de los problemas que presenta esta herramienta es que aunque es capaz de simular calles con varios carriles para cada sentido, los vehículos en cambio, no son capaces de cambiar de carril. Otras desventajas que presenta esta herramienta es la dependencia que tiene con SWANS, la inexistencia de una GUI que permita visualizar las simulaciones con claridad y que las salidas del simulador no pueden ser usadas directamente con un simulador de redes.

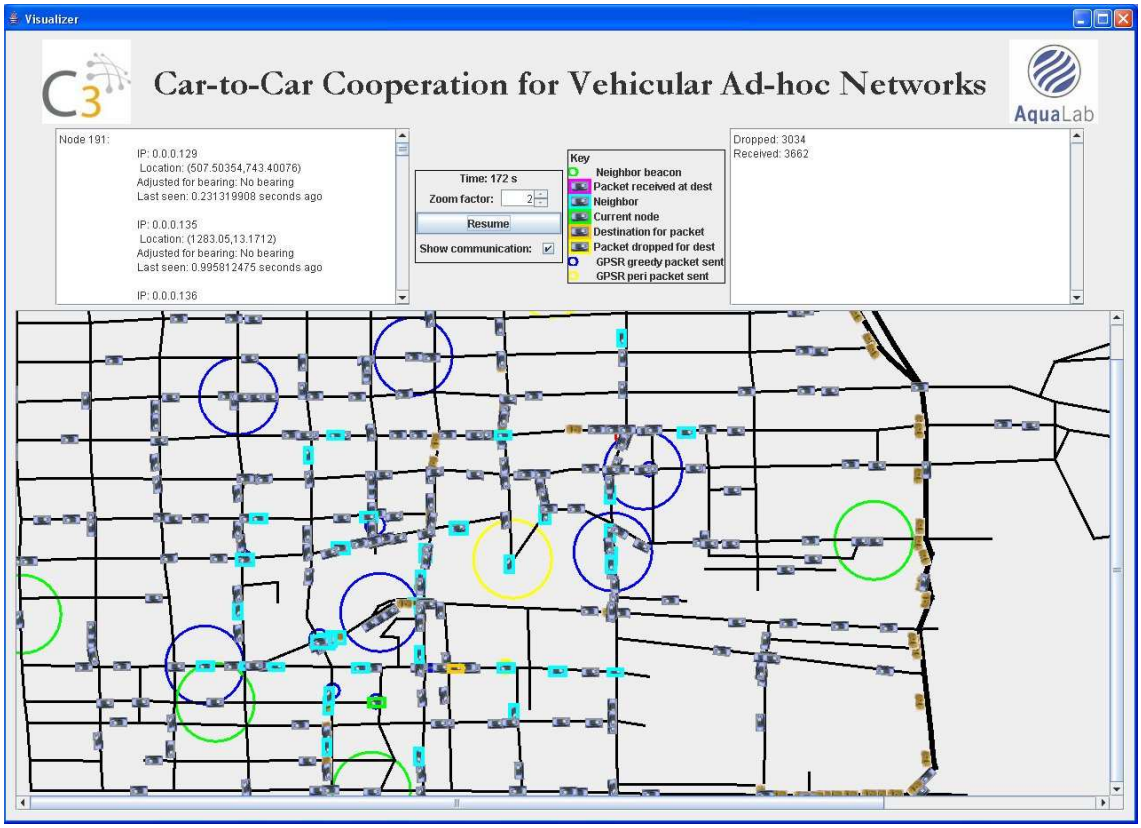

Figura 12 *Interfaz de STRAW.*

3.6 VanetMobiSim

VanetMobiSim es una extensión de CanuMobiSim, un simulador genérico de movilidad [CANU, HFFB]. VanetMobiSim, en cambio, tiene como objetivo extender el modelo de CanuMobiSim hacia un escenario en el que se reproducen simulaciones de vehículos de un modo realista. VanetMobiSim utiliza características tanto de nivel macroscópico, como microscópico. El movimiento en este simulador tiene en cuenta la topología, estructura y características de las calles. También atiende a la presencia de semáforos y señales de tráfico. Además permite el uso de varios tipos de mapas, como mapas extraídos de la base de datos TIGER, generación de mapas aleatorios haciendo uso de polígonos de Voronoi y también mapas definidos por un usuario. En cuanto a la simulación de vehículos esta aplicación realiza una separación física entre los dos flujos de vehículos en cada calle. También permite la introducción de calles con múltiples carriles en cada dirección además de la implementación de señales de tráfico en cada intersección. En cuanto a la generación de tráfico se incluyen todos los aspectos relativos al modelado de la velocidad y aceleración de cada vehículo. Además VanetMobiSim implementa la posibilidad de establecer puntos de interés. Esta herramienta permite crear un punto inicial y uno final para determinar qué viaje seguirá un vehículo durante la simulación. Estos puntos pueden ser determinados aleatoriamente, o atendiendo a puntos de interés.

Este simulador presenta tres formas diferentes de generar caminos, para un solo trayecto. Se pueden calcular bien por el algoritmo Dijkstra, atendiendo a la congestión de las calles, o bien atendiendo a la velocidad que permite cada calle. Aparte de estas características, VanetMobiSim implementa otras funcionalidades más ligadas con la micromovilidad, como el uso de los modelos IDM-IM y IDM-LC para las intersecciones.

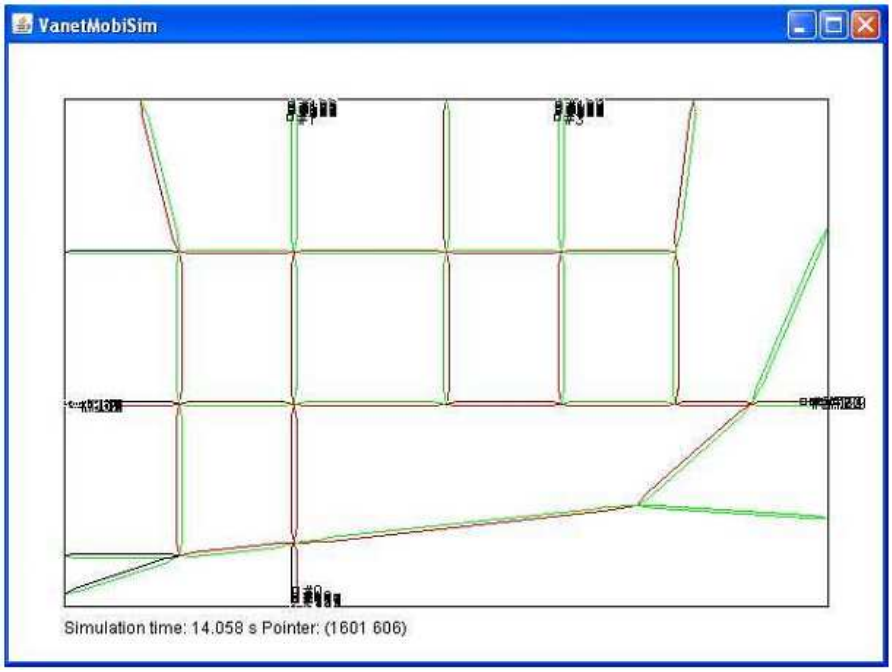

Figura 13 *Interfaz de VanetMobiSim.*

3.7 Comparativa

A continuación se presenta un cuadro comparativo donde se analizan las características de las distintas herramientas descritas, agrupadas en cuatro categorías [MTCCM09].

Este cuadro pretende ser un resumen de lo expuesto hasta ahora y también poner de relieve las principales ventajas e inconvenientes que ofrece cada herramienta.

	SUMO	MOVE	Trans	STRAW	VanetMobiSim
Software					
Portabilidad	Sí	Sí	Sí	Sí	Sí
Freeware	Sí	Sí	Sí	Sí	Sí
Open Source	Sí	Sí	Sí	Sí	Sí
Consola	Sí	Sí	-	-	No
GUI	Sí	Sí	Sí	Sí	Sí
Ejemplos disponibles	Sí	Sí	Sí	-	Sí
Desarrollo continuo	Sí	No	-	No	No
Facilidad instalación	Moderada	Fácil	Fácil	Moderada	Moderada
Facilidad de uso	Difícil	Moderada	Moderada	Moderada	Moderada
Mapas					
Real	Sí	Sí	Sí	Sí	Sí
Personalizado	Sí	Sí	Sí	-	Sí
Aleatorio	Sí	Sí	Sí	-	Sí
Manhattan	No	No	No	No	No
Voronoi	No	No	No	No	Sí
Movilidad					
Random waypoint	Sí	Sí	Sí	No	Sí
STRAW	Sí	Sí	-	Sí	No
Manhattan	Sí	Sí	No	No	No
Downtown	No	No	-	No	No
Modelos de tráfico					
Macroscópico	Sí	Sí	Sí	Sí	Sí
Microscópico	Sí	Sí	Sí	Sí	Sí
Varios carrilles	Sí	Sí	Sí	Sí	Sí
Cambio de carril	Sí	Sí	Sí	Sí	Sí
Varios sentidos	Sí	Sí	Sí	Sí	Sí
Límites de velocidad	Sí	Sí	Sí	Sí	Sí
Señales de tráfico	Sí	Sí	-	Sí	Sí
Gestión intersecciones	Sí	Sí	-	-	Sí
Adelantamientos	Sí	Sí	-	-	Sí
Redes calles largas	Sí	Sí	-	Sí	-
Movimiento libre	Sí	Sí	-	-	-
Varios tipos vehículos	Sí	Sí	-	-	No
Cruces por prioridad	Sí	Sí	-	-	No
Cálculo de ruta	Sí	Sí	Sí	Sí	Sí

Trazas					
Soporte NS-2	No	Sí	Sí	No	Sí
Soporte GloMoSim	No	Sí	-	No	Sí
Soporte Qualnet	No	Sí	-	No	Sí
Soporte SWANS	No	No	Sí	Sí	No
Soporte XML	No	No	-	No	Sí
Importación formatos	Sí	Sí	-	No	Sí

Tabla 2 *Comparativa de varios generadores de modelos de movilidad.*

3.8 Conclusiones

Los generadores de movilidad estudiados SUMO, MOVE, Trans, STRAW y VanetMobiSim son programas con muy buenas características y excelentes para la generación de modelos de movilidad. Como se ha podido comprobar en la *Tabla 2*, existen muchas semejanzas pero también grandes diferencias entre las herramientas.

Un punto en el que coinciden los generadores de movilidad es en las características de software. En general son programas de software libre, fáciles de conseguir y con una facilidad de uso moderada. En este aspecto la excepción es SUMO que tiene una dificultad alta en el uso, pero a cambio sus características son las más completas.

En cuanto a aspectos de la simulación, las cinco herramientas estudiadas tienen características muy similares habiendo muy pocas diferencias entre ellas.

Sí que se encuentran diferencias sustanciales si tenemos en cuenta los formatos con los que es posible trabajar con estos generadores. En este aspecto VanetMobiSim tiene un soporte bastante completo mientras que SUMO y STRAW no tienen muy buen soporte para formatos. Entrando más en detalles, SUMO permite la importación en varios formatos de mapas dejando de lado el soporte con los simuladores de redes. En cuanto a STRAW, sólo trabaja con SWANS lo que quizás pueda limitar la flexibilidad a la hora de trabajar con este generador de movilidad. VanetMobiSim da soporte para todos los formatos estudiados excepto para SWANS. Esto hace que VanetMobiSim sea especialmente flexible y que permita trabajar con los simuladores de redes que más se ajustan al trabajo de cada uno.

Cabe señalar que MOVE trabaja con SUMO y NS-2 gestionando la entrada de datos. Por este motivo es posible considerar a MOVE como un interfaz para estos programas.

4. Análisis y diseño

Este capítulo trata sobre la fase de análisis de la aplicación, y está dividido en tres partes. Primero se presenta el documento de especificación de requisitos en el que se detallan cada uno de los requisitos de la aplicación. En la segunda parte hay una planificación del proyecto explicada en un diagrama de Gantt. Para cerrar el capítulo el lector podrá encontrar una estimación de coste de la aplicación.

4.1 Documento de especificación de requisitos

4.1.1 Introducción

El presente apartado contiene una Especificación de Requisitos de Software de la herramienta de generación de modelos de movilidad del trabajo de fin de carrera titulado “Generación de modelos de movilidad para redes de vehículos a partir de mapas reales”. La estructura del presente documento ha sido realizada siguiendo las directrices referidas en el estándar IEEE Recommended Practice for Software Requirements Specifications IEEE 830-1998 [I830].

4.1.1.1 Propósito

El presente apartado tiene como propósito ser una guía para el desarrollo de la aplicación así como facilitar la posterior implementación. El actual capítulo irá dirigido al equipo de desarrollo de la aplicación, así como al cliente de la misma.

4.1.1.2 Alcance

La aplicación a desarrollar permitirá generar modelos de movilidad a partir de una interfaz de usuario en la cual, los usuarios, elegirán un mapa en el que podrán introducir los elementos que crean convenientes para crear un modelo de movilidad personalizado. Una vez aplicado éste, se generará la traza de movilidad.

4.1.1.3 Definiciones, acrónimos

API: Application Programming Interface es un conjunto de procedimientos que permiten que una aplicación pueda interactuar con otra completamente distinta como una capa de abstracción.

Área de trabajo o workspace: Es la parte de la interfaz en la que el usuario puede ver el mapa y trabajar con él.

C4R: Citymob for Roadmaps es el nombre de la aplicación. Como se podrá observar se usa indistintamente en cualquier momento el nombre real de la aplicación o sus siglas.

Calle: Un conjunto de carriles que tienen la misma dirección. Las calles pueden contener carriles con distinto sentido.

Carril: Cada una da las líneas del mapa por las que puede circular un vehículo. Los carriles están contenidos en calles y tienen la misma dirección que ésta.

Downtown: Es una zona del mapa donde los vehículos tienden a acercarse. Habitualmente suelen ser el centro de las ciudades aunque pueden ser también áreas comerciales o polígonos industriales.

GUI: Graphical User Interface o interfaz gráfica de usuario en castellano, es el entorno visual que permite al usuario interactuar con la aplicación.

OpenStreetMap: (OSM): Es un proyecto que tiene como objetivo ofrecer datos geográficos a todo el mundo de forma libre. Este proyecto está disponible en www.openstreetmap.com

ns-2: Network simulator 2 es un simulador de eventos orientado a la investigación de redes. Este simulador está disponible en: http://www.isi.edu/nsnam/ns/

Ruta: Es el camino que sigue un vehículo desde su partida hasta su llegada.

SUMO: Simulation of Urban Mobility es un simulador de tráfico microscópico de uso libre. Este proyecto está desarrollado principalmente por empleados por el Institute of Transportation Systems en el centro aerospacial de Alemania. SUMO está disponible en sumo.sourceforge.net/

VANET: Vehicular Ad-hoc Networt o red de vehículos ad-hoc en castellano, es una red ad-hoc formada por vehículos conectados entre sí para intercambiar datos.

XML: Acrónimo de Extensible Markup Languaje o lenguaje de marcas extensible en castellano, es un metalenguaje que sirve para almacenar datos de forma estructurada.

4.1.1.4 Referencias

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

4.1.1.5 Resumen

El siguiente subapartado proporciona una idea general de las funcionalidades del producto. Se pretende crear una base sobre los requerimientos técnicos, que serán abordados más adelante.

El tercer subapartado de este capítulo, está dirigido principalmente a los desarrolladores de la aplicación, y en él se explican los pormenores de las funcionalidades de la aplicación expresados de una forma técnica.

Ambas secciones describen el mismo software, pero con la salvedad de que cada uno va dirigido a un público distinto, por lo que en consecuencia se usa un lenguaje distinto.

4.1.2 Descripción global

4.1.2.1 Perspectiva del producto

Este es un producto que forma parte de un sistema mayor que tiene como objetivo la simulación de VANETs. Este programa se sirve de OpenStreetMap para conseguir mapas reales y de SUMO para simular el tráfico.

A continuación se va a presentar un diagrama que muestra cómo se integra esta aplicación con las demás herramientas usadas parte se encuentra el producto:

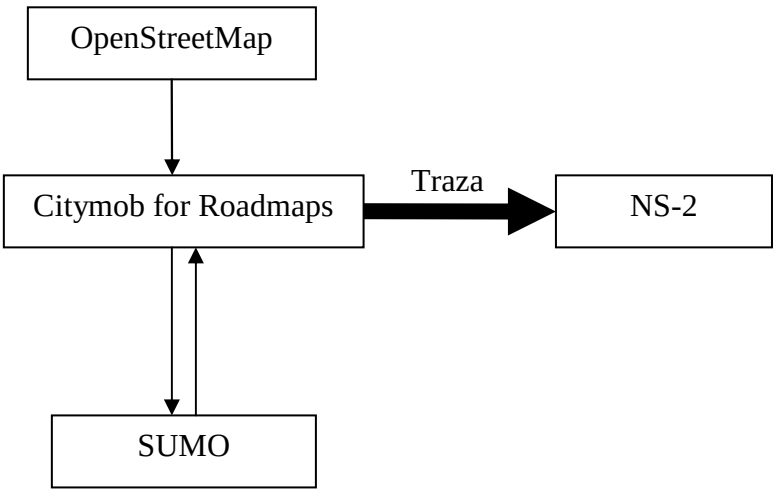

Figura 14 *Relación de Citymob for Roadmaps con otras aplicaciones.*

4.1.2.2 Funcionalidad del producto

El programa se encarga de generar modelos de movilidad mediante una serie de parámetros que son pedidos al usuario mediante una GUI. Una vez conseguidos los datos, pasarán a un simulador de tráfico para que se pueda obtener una traza de la simulación, poder visualizarla en un entorno gráfico, y utilizarla como entrada en un simulador de redes (en concreto, ns-2).

El producto debe permitir:

FUNC_1: Obtener un mapa del proyecto OpenStreetMap mediante la interfaz gráfica de la aplicación.

FUNC_2: Abrir un proyecto creado con este producto.

FUNC_3: Guardar un proyecto.

FUNC_4: Importar archivos al proyecto.

FUNC_5: Exportar el proyecto

FUNC_6: Generar flujos de tráfico, para lo cual el usuario especificará los vehículos que han de componer la simulación y sus características mediante la interfaz.

FUNC_7: Especificar los modelos de movilidad que se usarán sobre el mapa.

FUNC_8: Permitir que sea posible visualizar la simulación creada por el usuario.

FUNC_9: Devolver al usuario una traza de salida compatible con ns-2.

FUNC_10: Generar vehículos con rutas aleatorias.

FUNC_11: Definir centros urbanos y puntos de atracción.

4.1.2.3 Características de los usuarios

Los usuarios a los que se dirige este producto son investigadores que dedican su esfuerzo a la simulación de VANETs. Estos usuarios utilizan habitualmente este tipo de software y realizan numerosas simulaciones por lo que se les puede considerar expertos manejando software con características similares a este producto. A pesar de conocer el funcionamiento del software, es una tarea muy costosa en tiempo, y además es posible que no estén familiarizados con el funcionamiento de XML y que tengan dificultades a la hora de editar estos ficheros. Este producto viene a solucionar en la medida de lo posible este problema, así como a reducir el tiempo necesario para la generación de escenarios para simular VANETs, pudiendo crear modelos de movilidad con toda libertad y olvidándose de la molestia de editar archivos.

4.1.2.4 Restricciones

Este producto trabaja tanto con OpenStreetMap como con SUMO y por tanto, está restringido a trabajar con los formatos que ofrecen estas aplicaciones. Las restricciones más importantes vienen de SUMO, ya que cualquier error en un archivo de proyecto paraliza por completo la creación de una traza.

4.1.2.5 Suposiciones y dependencias

El funcionamiento de ciertas funciones como llamar a SUMO para visualizar simulaciones, está supeditado a la compatibilidad con SUMO. Esto quiere decir que si

SUMO cambia algunas de sus características es posible que la aplicación no funcione como es esperado. La versión de SUMO con la que se realiza este producto es la 0.11.1.

Citymob for Roadmaps accede a la base de datos de OpenStreetMap para seleccionar los mapas que se usarán. Por eso también depende de que esta base de datos siga siendo compatible con el producto y con SUMO. La aplicación usará para comunicarse con OpenStreetMap la API v0.6, aunque será posible cambiarla cómodamente si hubiese algún cambio en el futuro.

4.1.2.6 Evolución previsible del sistema

Es previsible que esta aplicación pueda mejorarse aumentando la flexibilidad que ofrece a la hora de crear modelos de movilidad. Para ello sería necesario ampliar otras funcionalidades de SUMO.

Además es posible mejorar ciertas características que hagan más amigable e intuitivo usar la aplicación, como

4.1.3 Requisitos específicos

4.1.3.1 Requisitos de las interfaces externas

Interfaz de usuario

El producto se dividirá en una serie de ventanas en las cuales el usuario podrá trabajar a su antojo. Los requisitos de las distintas ventanas se detallan a continuación:

Ventana Principal

- Contendrá una menú desplegable con las opciones disponibles
- Contendrá una barra de herramientas con las siguientes opciones:
    - Nuevo escenario
    - Abrir escenario
    - Guardar escenario
    - Descargar mapa de OpenStreetMap
    - Crear vehículos aleatorios
    - Nuevo vehículo
    - Nueva ruta
    - Crear downtown
    - Visualizar simulación
    - Asistente

- Contendrá un espacio de trabajo donde se cargará el trabajo del usuario. Ésta será la parte de la aplicación en la que aparecerá el mapa con el que se está trabajando y donde el usuario tendrá que señalar qué partes del mapa quiere usar.
- Cuando se haya cargado un mapa en la aplicación debe aparecer una barra que permita hacer zoom con el ratón de una forma intuitiva.

Ventana elegir mapa

- Contendrá un mapa descargado desde OpenStreetMap
- Una barra de desplazamiento para realizar zoom sobre el mapa
- Información sobre la porción de mapa que se está visualizando

Ventana opciones

- Contendrá una caja de texto en la que se pedirá la ruta de SUMO y un botón para buscar la ubicación de SUMO.
- Contendrá una caja de texto que contendrá información sobre la API de OpenStreetMap.

Ventana de vehículos y tipos de vehículo

- La ventaja tendrá dispuestos los datos en forma de tabla de forma que se puedan visualizar y editar.
- Los datos se que se hayan introducido previamente deben aparecer reflejados en la tabla.
- Se podrán introducir nuevos datos.
- Se podrán editar los datos seleccionados

Ventana de rutas

- Sólo servirá para introducir nuevas rutas
- En el diálogo aparecerá una caja de texto par introducir un identificador, una lista con los carriles introducidos por el usuario y un combobox con los posibles carriles.
- Será obligatorio introducir un identificador para que la ruta sea válida
- Los carriles que aparecerán en el combobox deberán ser contiguos al último carril introducido por el usuario.

Ventana de downtown

- Este diálogo se abrirá acto seguido de que el usuario seleccione un conjunto de calles para ser un downtown.
- En el diálogo aparecerá un identificador sin el cual no se podrá crear un downtown.
- Un valor de atracción que estará comprendido entre 0 y 1.
- El diálogo permitirá cancelar la operación de creación de un downtown.

Ventana de modelos de movilidad

- En el diálogo aparecerá una lista con los modelos disponibles con la posibilidad de seleccionar uno con un radiobutton.
- Como cada modelo requiere un conjunto distinto de parámetros, en función del modelo seleccionado, se pedirán los parámetros que corresponden para su creación.

Ventana de vehículos aleatorios indefinidos

- Se requerirá una caja de texto donde aparezcan el número de vehículos a generar
- Deberá aparecer una tasa de creación de vehículos fuera de downtowns. Esta tasa deberá tener un valor entre 0 y 1.

- Tendrá que haber una caja de texto para introducir el tiempo en el que aparece en la simulación.

Ventana de vehículos aleatorios definidos

- Se requerirá una caja en la que se pueda escribir el carril donde se generarán vehículos aleatorios. Esta caja también podrá ser rellenada seleccionando un carril por medio del ratón.
- Se requerirá una caja con las mismas condiciones del punto anterior para la posición final de los vehículos aleatorios.
- Deberá haber una caja de texto donde se podrán introducir el número de vehículos aleatorios a crear.
- Será necesaria una caja de texto donde introducir el tiempo de aparición en la simulación de los vehículos aleatorios.

Ventanas de vistas de datos

- Todas las vistas de datos deben permitir ver los datos al que corresponde la vista, permitiendo editarlos y eliminarlos si fuera necesario.
- Las vistas no permitirán introducir nuevos datos.

Ventana de creación de traza

- Constará de una caja de texto que pedirá el número de trazas a crear.
- Constará de una caja de texto que determinará el tiempo final de la simulación.

Ventana de asistente

- Constará de un botón “Next” que permitirá avanzar
- Constará de un botón “Back” que permitirá retroceder
- La ventana de asistente constará de un botón “Cancel” que permitirá abandonar el asistente en cualquier momento preguntando al usuario si de verdad quiere abandonar el asistente
- La ventana de asistente constará de un panel que irán cambiando según se vaya avanzando en el asistente.
- Los botones “Next” y “Back” se podrán deshabilitar en cualquier momento con el objetivo de evitar inconsistencias tanto en la aplicación como en el proyecto que se está creando.

Ventana about

- Deberá aparecer el logotipo de la aplicación.
- Deberá aparecer el logotipo del Grupo de Redes de Computadores.
- Deberá aparecer el logotipo de la Fundación Universitaria Antonio Gargallo.
- Deberá aparecer el logotipo de la Caja de Ahorros de la Inmaculada.
- Deberán aparecer los nombres de los colaboradores.
- Deberá aparecer un aviso legal sobre la aplicación.

Interfaz Hardware

Al ser una aplicación destinada a instalarse en equipos personales, es necesario que la aplicación funcione en cualquier tipo de hardware. El principal objetivo es que

funciones en equipos domésticos, pues es previsiblemente el tipo de hardware que más se va a usar para esta aplicación.

Interfaz software

Al ser una aplicación que trabaja con otras aplicaciones hay que crear usar interfaces software para llevar a cabo una correcta comunicación. Las distintas interfaces software identificadas son:

- Servidor de mapas de OpenStreetMap
- SUMO

Interfaz de comunicaciones

Puesto que la aplicación ha de conectarse a servidores externos, va a ser necesario el uso del protocolo HTTP para establecer las comunicaciones.

4.1.3.2 Requisitos funcionales

El sistema deberá permitir:

REQ_1: Obtener un mapa conectándose a los servidores de www.openstreetmap.org y permitir navegar por el mismo.

Para navegar por el mapa se podrá:

- Acercar o alejar la vista actual del mapa con la ruleta del ratón o con una barra deslizante que se mostrará en el espacio de trabajo.
- Mover el mapa con el cursor.
- Seleccionar la porción de mapa deseada.

REQ_2: Obtener una copia de la parte del mapa seleccionado en la aplicación. Deberá aparecer una reproducción del mismo en la zona de trabajo. En esta reproducción es importante que queden reflejados los distintos carriles por los cuales pueden circular los vehículos.

REQ_3: Añadir vehículos al mapa mediante la GUI. Para esto se facilitarán los siguientes datos:

- Identificación
- La ruta que seguirá el vehículo
- Tiempo en el que aparecerá el vehículo en la simulación
- Tipo de vehículo
- Color

REQ_4: Definir distintos tipos de vehículos. Para esto el usuario facilitará los siguientes datos:

- Identificador
- Aceleración
- Deceleración

- Longitud del vehículo
- Velocidad máxima
- Imperfección de los conductores a la hora de conducir. Este valor será un número real comprendido entre 0 y 1 inclusive.
- Tipo del vehículo admitido por SUMO

REQ_5: Definir una ruta de viaje. Para esta tarea se exigirán los siguientes parámetros:
- Identificador de la ruta
- Una lista ordenada de los carriles que se seguirán durante la ruta

REQ_6: Definir un downtown. Para ello el usuario seleccionará la parte del mapa que crea conveniente. El downtown se verá reflejado en la interfaz gráfica mediante un recuadro de líneas discontinuas. Antes de dibujar el recuadro se pedirán los siguientes datos al usuario:
- Identificador del downtown.
- Atracción que será un número de 0 a 1 que indicará la atracción que ejerce el downtown sobre los vehículos.

REQ_7: Crear vehículos aleatorios con origen y destino definidos. Estos vehículos se moverán aleatoriamente por el mapa, viéndose influidos por los distintos downtowns que puedan existir. Los datos que serán requeridos a este fin son:
- Punto de salida
- Punto de llegada
- Número de vehículos generados
- Tiempo de aparición en la simulación

REQ_8: Crear vehículos aleatorios indefinidos. Estos vehículos se moverán aleatoriamente por el mapa viéndose afectados por los distintos downtowns que puedan existir. Adicionalmente, el punto de salida de estos vehículos está relacionado intrínsecamente con los downtowns. De este modo, cuanto mayor sea la atracción de un downtown, mayor será la posibilidad de que un vehículo aparezca en ese downtown. Para la creación de este tipo de vehículos se le requerirá al usuario:
- El número de vehículos a generar
- La tasa de creación de vehículos dentro de los downtowns. Este se serán un valor entre 0 y 1 que determinará que porción de los vehículos aparece dentro de un downtown.
- El tiempo de aparición en la simulación.

REQ_9: Crear la traza de la simulación con SUMO y hacerla compatible con NS-2. Para esto será imprescindible que el usuario facilite la ruta de instalación de SUMO.

REQ_10: Será posible visualizar el modelo de movilidad una vez creado a través de la aplicación de SUMO para visualizar trazas. Para que el usuario pueda visualizar el modelo creado la aplicación deberá configurar la aplicación de SUMO sin que el usuario tenga que configurar nada si no lo desea.

REQ_11: Definir qué modelo de movilidad se seguirá en la simulación. Al usuario le serán requeridos una serie de parámetros en función del modelo elegido. Se podrá elegir entre una de las siguientes opciones:

- Modified Krauß model
- Original Krauß model
- Peter Wagner 2009 model
- Kerner model
- Intelligent Driver Model

REQ_12: Se requerirá la creación de un asistente que permita a los nuevos usuarios familiarizarse con la aplicación y crear modelos de movilidad de una forma rápida y siendo guiados paso a paso.

4.1.3.3 Requisitos del desarrollo

REQ_13: La aplicación será desarrollada íntegramente en lengua inglesa, tanto el código fuente como la interfaz de usuario.

REQ_14: Sólo se soportará una terminal de trabajo.

REQ_15: Sólo habrá un usuario que pueda estar utilizando a la vez este producto.

REQ_16: Ante situaciones de error, éstas se resolverán informando al usuario mediante los correspondientes mensajes de aviso en caso de que la aplicación no pueda recuperarse del error.

4.1.3.4 Restricciones del diseño

REQ_17: La información de entrada será suministrada por archivos bajo formato XML o por el usuario mediante la GUI.

REQ_18: Los distintos ficheros que conformarán la aplicación estarán en XML.

REQ_19: Los archivos de mapa y ruta serán compatibles con SUMO.

REQ_20: Los archivos de traza generados por la aplicación serán compatibles con el simulador de redes ns-2.

REQ_21: La aplicación debe ser multiplataforma y debe funcionar, al menos, en equipos con los sistemas operativos Windows y Linux.

4.2 Planificación

Para llevar a cabo la planificación se ha tenido en cuenta cada uno de las tareas que son necesarias para llevar a buen puerto el desarrollo de una aplicación. De este modo, se ha dividido la planificación en análisis, desarrollo, implementación y pruebas. Las fases de análisis, diseño y pruebas contienen las tareas habituales en estas fases por lo que se va a obviar la explicación de cada una de ellas. En cambio, la fase de desarrollo está dividida, a su vez, en más fases que corresponden a prototipos.

Finalmente en este apartado también se muestra un diagrama de Gantt que clarifica y resume la planificación seguida para la realización de la aplicación.

4.2.1 Prototipos

Cada una de las fases del desarrollo añade de forma incremental funcionalidades a la aplicación. Hay que tener en cuenta que cada una de estas fases corresponde a un prototipo que es plenamente funcional.

A continuación se van a presentar cada una de las fases explicando en qué consiste cada una de ellas:

Fase 1 – Creación de la GUI

El objetivo de esta fase es crear una GUI de forma que, aunque todavía no haya funcionalidades, se pueda intuir cuál va a ser el resultado final pudiendo navegar por la interfaz. Para llevar a cabo esta tarea hay que construir la ventana principal de la aplicación con sus correspondientes menús. Además, a esta ventana habrá que añadir la ventana desde la que se pueden visualizar e importar los mapas y los paneles que aparecen en el espacio de trabajo del usuario.

Fase 2 – Importar y cargar mapas

Esta fase trata de dotar al prototipo conseguido en la Fase 1 de la posibilidad de cargar mapas directamente desde OpenStreetMap y transformarlos al formato de mapas de SUMO.

Fase 3 – Mostrar mapas

Una vez realizado el prototipo anterior, se trata de poder mostrar al usuario los mapas que ya se pueden cargar ya sea de archivo o directamente desde el mapa que ofrece la aplicación. Además de esto, también se pretende que los distintos objetos que aparecen en el mapa se puedan seleccionar de forma que el usuario pueda interactuar con ellos.

Fase 4 – Elementos de movilidad

Este prototipo se diferencia de los anteriores en que debe permitir añadir vehículos y rutas al mapa cargado en el espacio de trabajo, para representar de un modo lo más visual posible los vehículos que conforman el modelo y sus rutas asociadas.

Fase 5 – Modelos de movilidad

En esta fase se pretende añadir la posibilidad de agregar centros de atracción al mapa cargado por el usuario. En esta fase también se pretende añadir los algoritmos que definen los parámetros de movimientos de cada coche como la aceleración, velocidad, etc.

Fase 6 – Resultado de la simulación

Este prototipo se centra en crear los resultados devueltos al usuario. Así pues, se trata de terminar las funcionalidades añadiendo la posibilidad de generar una traza o visualizar la simulación.

4.2.2 Diagrama de Gantt

En este apartado se van a ver las diferentes planificaciones para el proyecto. Se ha creído interesante mostrar las distintas configuraciones de la planificación y también reproducir la que realmente se ha seguido para ver las diferencias existentes.

Planificación inicial

El siguiente diagrama muestra la planificación realizada antes del desarrollo de la aplicación. Este diagrama se ha diseñado para ser desarrollado por una sola persona, por eso no es posible llevar a cabo tareas paralelas, de modo que todas las tareas se han planificado de un modo secuencial.

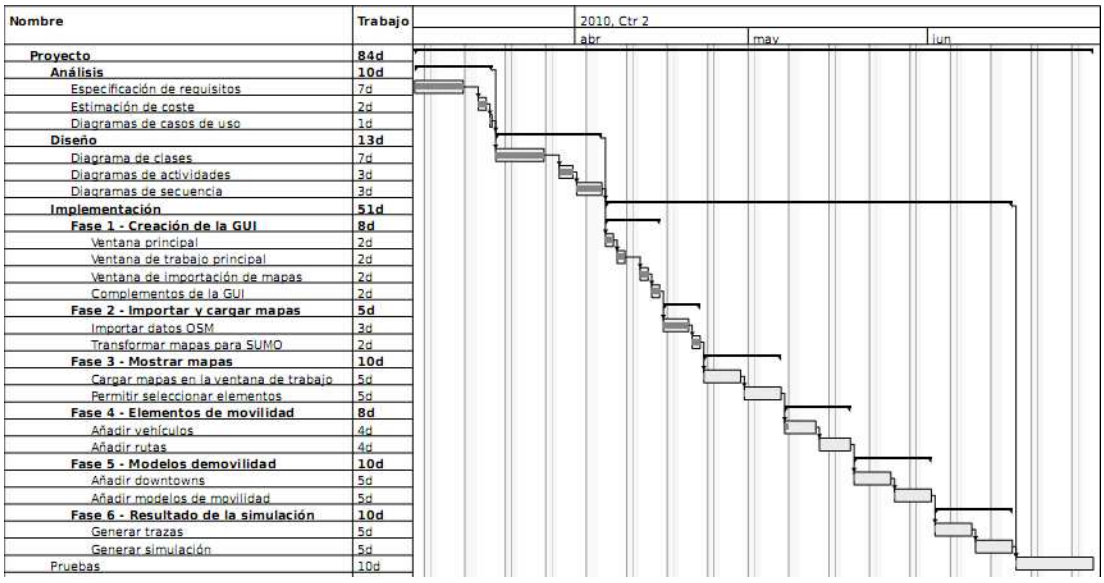

Figura 15 *Diagrama de Gantt de la planificación inicial.*

Planificación para varios desarrolladores

Además se muestra la planificación que correspondería a varios desarrolladores, permitiendo realizar tareas en paralelo. Como se podrá observar el tiempo necesario para acabar la aplicación se recorta considerablemente.

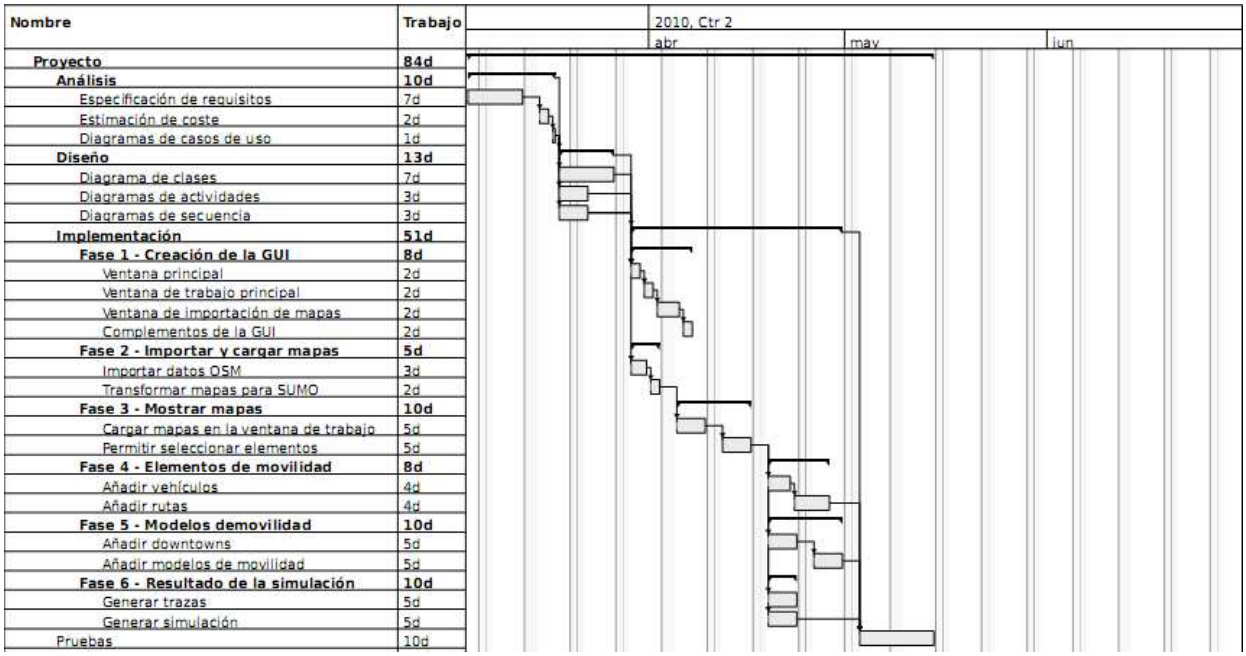

Figura 16 *Diagrama de Gantt de la planificación para varios desarrolladores.*

Calendario seguido

A continuación se va a mostrar el diagrama de Gantt correspondiente al calendario que realmente se ha seguido para el desarrollo de la aplicación. Con esto se pretende hacer una comparación entre lo planeado y lo real para llevar a cabo una valoración de la estimación.

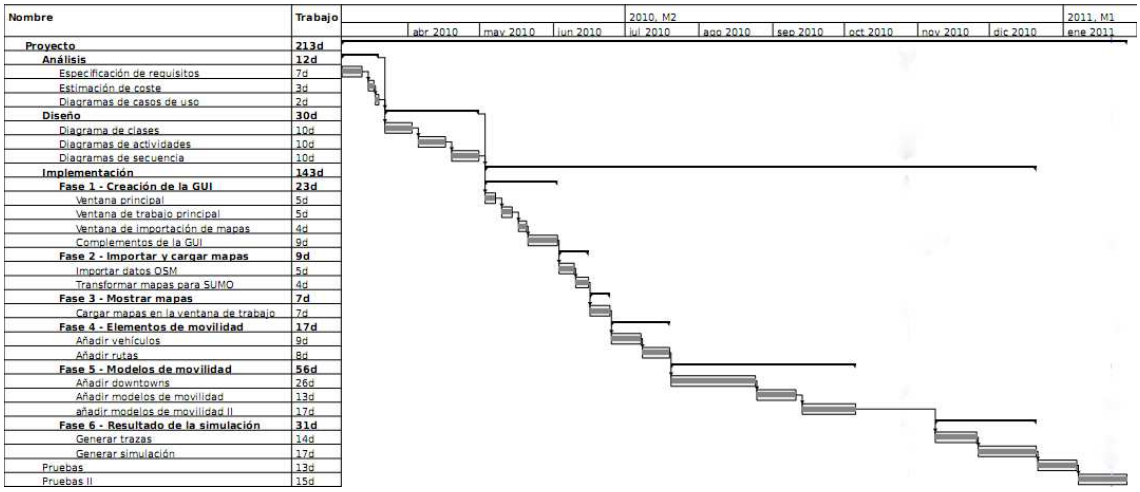

Figura 17 *Diagrama de Gantt sobre el calendario seguido.*

4.2.3 Estimación de coste

En el desarrollo de aplicaciones informáticas, al igual que en toda actividad industrial, es importante conocer los gastos para calcular la rentabilidad de un producto. Para ello es necesario realizar la estimación de coste correctamente para evitar tener pérdidas y sobre todo para obtener valiosos beneficios económicos.

Estimación con la métrica puntos función

Una vez conocidos los requisitos de la aplicación, es necesario hacer una previsión del coste de la misma. Así los desarrolladores y los clientes pueden tener una idea aproximada del coste y evitar sorpresas. En este caso se ha elegido llevar a cabo la estimación por el método de puntos función [GH00]. Este método consiste en asignar una serie de puntos a la aplicación informática según la complejidad de los ficheros que maneja y de los procesos que tratan la información de los ficheros. Siempre tratando de considerarlo desde el punto de vista del usuario.

Para comenzar la estimación debemos obtener una serie de datos que nos facilitarán el cálculo. Primero obtendremos los datos funcionales. Estos datos están divididos en dos subconjuntos: los Internal Logic File (ILF) y los External Interface File (EIF).

Un ILF es un grupo de datos lógicamente relacionados, identificables por el usuario y mantenidos a través de procesos dentro de los límites de la aplicación. Un EIF, en cambio, es un grupo de datos lógicamente relacionados, identificables por el usuario y usados por la aplicación pero mantenidos dentro de los límites de otra aplicación.

Una vez identificados los ficheros, hay que identificar nuevos elementos: los Data Element Type (DET) y los Record Element Type (RET). Los DETs corresponden a atributos o campos de los ficheros, incluyendo las claves foráneas. Los RETs corresponden a campos del fichero que referencian a otros ficheros.

La complejidad de los datos que maneja la aplicación es calculada en base a una matriz – *Tabla 3* – que asigna valores en función de los DET's y RET's que cada fichero maneja.

RET's	1-19 DET's	20-50 DET's	51+ DET's
1	Baja	Baja	Media
2 – 5	Baja	Media	Alta
6+	Media	Alta	Alta

Tabla 3 *Matriz de complejidad para ILF's y EIF's*

Todos estos datos se relacionan en la *Tabla 4* que describe la situación de cada fichero y su complejidad.

Nombre de fichero	Tipo	Número de DET's	Número de RET's	Complejidad
Configuración	ILF	3	0	Baja
Traza de SUMO	ILF	1	0	Baja
Downtown	ILF	3	1	Baja
Log	ILF	1	0	Baja
Mapa de SUMO	ILF	1	0	Baja
Proyecto de C4R	ILF	4	4	Baja
Mapa de OSM	EIF	1	0	Baja
Vehículos aleatorios	ILF	10	0	Baja
Rutas	ILF	21	1	Baja
Formas de C4R	ILF	5	1	Baja
Proyecto de SUMO	ILF	2	2	Baja

Tabla 4 *Estimación de la complejidad para ILF's y EIF's.*

Para continuar hay que identificar las distintas transacciones funcionales, divididas en entradas (EI – External Inputs), salidas (External Outputs) y consultas (EQ – External Inquiri). Como es de suponer, las entradas hacen referencia a los procesos de entrada de datos en el sistema, las salidas hacen referencia a los procesos de salida de datos del sistema y las consultas hacen referencia a los procesos de recuperación de datos internos del sistema. Al igual que en el paso anterior hay que determinar la complejidad de cada una de las transacciones. A tal efecto, hay también unas matrices de complejidad que siguen la misma mecánica vista anteriormente. La principal diferencia es la existencia de un nuevo parámetro File Types Referenced (FTR) y el número de ficheros a los que accede cada transacción.

Matriz de complejidad para las EI's			
FTR's	**1 – 4 DET's**	**5 – 15 DET's**	**16+ DET's**
0 – 1	Baja	Baja	Media
2	Baja	Media	Alta
3+	Media	Alta	Alta
Matriz de complejidad para las EO's			
FTR's	**1 – 5 DET's**	**5 – 19 DET's**	**20+ DET's**
0 – 1	Baja	Baja	Media
2 – 3	Baja	Media	Alta
4+	Media	Alta	Alta
Matriz de complejidad para las EQ's			
FTR's	**1 – 4 DET's**	**5 – 15 DET's**	**16+ DET's**
0 – 1	Baja	Baja	Media
2	Baja	Media	Alta
3+	Media	Alta	Alta

Tabla 5 *Matriz de complejidad para EI's, EO's y EQ's.*

A continuación se puede observar la matriz de complejidad para cada una de las transacciones que realiza la aplicación con cada fichero.

Nombre de la transacción	Tipo	FTR's	DET's	Complejidad
Inicio aplicación	EI	1	3	Baja
Nuevo proyecto	EO	1	1	Baja
Abrir proyecto	EI	6	44	Alta
Guardar proyecto	EO	6	44	Alta
Importar archivo	EI	1	21	Media
Exportar proyecto	EO	1	2	Baja
Crear traza	EO	1	1	Baja
Visualizar traza	EQ	1	2	Baja
Seleccionar mapa	EI / EO	2	2	Baja / Baja
Vehículos aleatorios indefinidos	EI	2	31	Alta
Vehículos aleatorios definidos	EI	2	31	Alta
Añadir vehículos	EI	1	5	Baja
Añadir tipos de vehículos	EI	1	7	Baja
Añadir Ruta	EI	1	2	Baja
Añadir downtown	EI	2	8	Media
Añadir modelo de movilidad	EI	1	5	Baja
Configurar opciones	EI	1	3	Baja
Ver vehículos aleatorios	EO	1	5	Baja
Ver Downtowns	EO	1	2	Baja
Ver rutas	EO	1	2	Baja
Volcar vehículos aleatorios	EQ	2	31	Alta
Crear traza	EO	1	5	Baja
Convertir traza a ns-2	EO	1	5	Baja
Calcular ruta	EQ	3	7	Alta
Calcular distancia	EQ	2	4	Baja
Calcular camino más corto	EQ	2	5	Media

Tabla 6 *Estimación de la complejidad para EI's, EO's y EQ's.*

Llegados a este punto y con todos los datos, se procede a calcular los puntos función. Para ello hay que completar la siguiente tabla con los datos obtenidos anteriormente. El resultado obtenido se llama Unadjusted Function Points (UFP).

	Baja		**Media**		**Alta**		**Total**
EI	7	x3	2	x4	3	x6	**47**
EO	9	x4	0	x5	1	x7	**43**
EQ	2	x3	1	x4	2	x6	**22**
ILF	10	x7	0	x10	0	x15	**70**
EIF	1	x5	0	x7	0	x10	**5**
						UFP	**187**

Tabla 7 ***Cálculo de los puntos función.***

El siguiente paso es obtener las características generales del sistema (GSC). A cada característica del sistema se le asigna un valor entero entre 0 y 5, donde 0 significa que no influye y 5 una fuerte influencia. La suma de la puntuación de cada característica da como resultado el grado total de influencia (TDI).

Característica	**Valor**
Comunicaciones de datos	1
Funciones distribuidas	0
Rendimiento	0
Configuraciones fuertemente utilizadas	1
Frecuencia transacciones	0
Entrada de datos on-line	1
Eficiencia del usuario final	2
Actualización on-line	0
Reutilización	2
Procesos complejos	2
Facilidad de instalación	1
Facilidad de operación	0
Instalación en distintos lugares	5
Facilidad de cambios	4
TDI	**19**

Tabla 8 Cálculo del GSC

A partir del TDI calcularemos el valor del factor de ajuste (VAF) que se calcula con la siguiente fórmula:

*VAF = (TDI * 0,01) + 0,65.*

Por lo tanto, *VAF = (19 * 0,01) + 0,65 = 0,84*

Ahora obtenemos los puntos función de la nueva aplicación (AFP) multiplicando los puntos función por el factor de ajuste:

*AFP = 187 * 0,84 = 157,08*

Suponiendo que la productividad es un punto función por día, entonces se obtiene un resultado de 158 días para desarrollar la aplicación.

Asumiendo que cada día de desarrollo implica 8 horas de trabajo y que el coste de producción de 1 hora de aplicación es de 30 €, entonces el coste estimado de la aplicación es de:

*158 días * 8 h./día * 30 €/h. = 37.920 €*

4.3.2 Coste real de la aplicación

En el punto anterior se ha calculado cuál es el coste de la estimación, pero ese coste no es válido si no corresponde con la realidad. Por eso es interesante calcular el coste real de la aplicación para ver la diferencia y analizar la fuente de errores para evitar que en un futuro pueda volver a ocurrir.

Para calcular el coste real se va a seguir una técnica parecida a la estimación. Primero hay que conocer el número de días que se ha tardado en finalizar el desarrollo de C4R. Este dato lo podemos obtener del apartado 4.2.2 – *Figura 17* –. Según el diagrama de Gantt, el desarrollo de la aplicación ha costado 213 días. Tal como se puede apreciar en el diagrama, durante el desarrollo ha habido tiempos de parada. Por este motivo y porque no todos los días la dedicación a la aplicación ha sido la misma, es difícil calcular con exactitud el verdadero coste. Lo que sí se puede calcular con bastante aproximación es la dedicación diaria, ya que los días que se ha trabajado en el proyecto se ha invertido un número parecido de horas. Como resultado de todo esto obtenemos que la media de dedicación es de 5 h/día. Por lo tanto:

*213 días * 5 h./día * 30 €/h. = 31.950 €*

4.4 Conclusiones

A lo largo del capítulo se han visto las distintas estimaciones de tiempo y coste y también cuál ha sido el resultado real. Como se puede comprobar existen divergencias en los resultados.

En cuanto al tiempo estimado ha habido una desfase de 55 días más, lo que supone un error de estimación del 25,82%. Aunque teniendo en cuenta que los días estimados son de 8 horas y los reales de 5 horas, lo más justo es comparar entre horas de desarrollo. En tal caso, las horas planificadas son un total de 1.264 horas y las reales son 1.065, lo que supone un 15,74% de error. Esto significa que el proyecto ha sido desarrollado en menos tiempo del que cabría esperar. El error en la estimación es achacable a varios factores. El primer factor sería la ausencia de historia previa. Al no contar con datos de estimación de proyectos similares es fácil subestimar tareas que a

priori parecen sencillas. Asimismo, la inexperiencia a la hora de realizar una estimación para proyectos de esta envergadura y características también influyen. Otro factor influyente es la dedicación. El hecho de que durante el transcurso del desarrollo hayan aparecido otros quehaceres al desarrollador ha hecho ralentizar tareas que estaban previstas para un tiempo mucho menor y el no poder dedicar el tiempo planificado al proyecto.

En cuanto al coste, el desfase ha sido de 5.970 € menos, lo que significa un error del 15,74%. En este caso, como el coste está íntimamente relacionado con el tiempo, los factores que han influido han sido los mismos, de modo que como se podrá observar hay una relación entre el error de estimación temporal y económico.

5. Implementación

Este apartado trata sobre distintos aspectos de la implementación. En él se justifican las decisiones tomadas durante la fase de desarrollo y la estructura de la solución adoptada.

5.1 Casos de uso

Una vez descritas las especificaciones, se pueden crear los diagramas de casos de uso, que permiten clarificar las diferentes posibilidades que tiene el usuario frente a la aplicación.

Figura 18 *Casos de uso de Citymob for Roadmaps.*

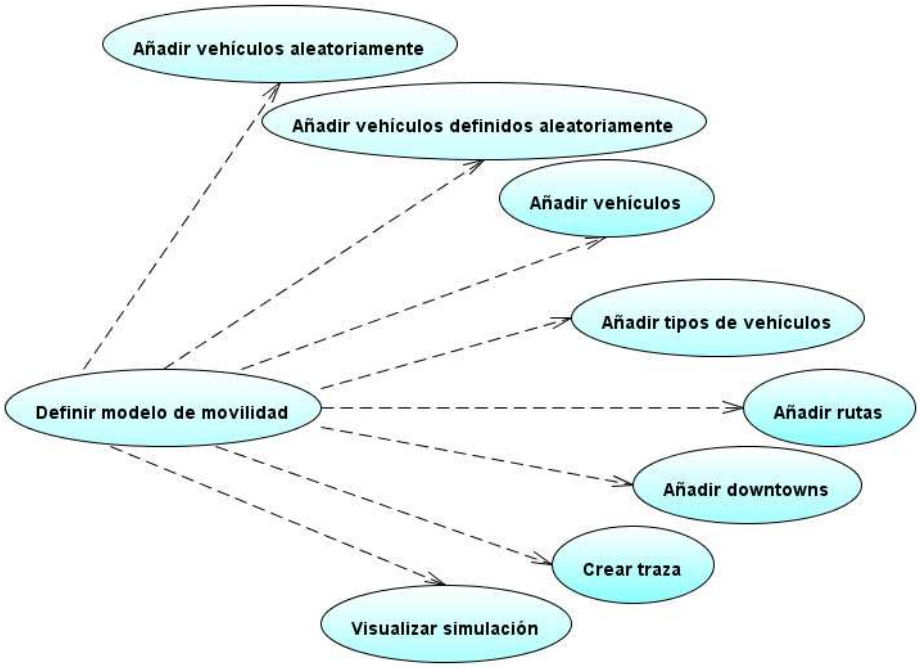

Figura 19 *Desarrollo de la burbuja "Definir modelo de movilidad".*

Los diagramas han sido divididos en partes por cuestiones de comodidad y sencillez para entender su contenido.

El diagrama de la *Figura 18* muestra las opciones de las que dispone el usuario cuando inicia la aplicación. Como se puede ver, el usuario sólo puede configurar las opciones de la aplicación, crear un proyecto nuevo o abrir un proyecto. Si el usuario decide crear un proyecto, después deberá obtener un mapa y a continuación pasar a definir los parámetros de un modelo de movilidad. Si el usuario decide, por el contrario, abrir un proyecto ya existente se saltará la fase de obtener mapa puesto que ya ha sido obtenido cuando se creó el proyecto.

En la *Figura 19* se muestra el desarrollo de la burbuja "Definir modelos de movilidad". Este diagrama es bastante simple, puesto que se limita a listar las opciones que permiten definir un modelo de movilidad y que quedan disponibles una vez hay un mapa cargado con el que trabajar.

De los casos de uso, lo que quizás más llame la atención es que para generar satisfactoriamente un modelo de movilidad con la aplicación, hay que seguir una serie de pasos. Primero hay que crear el proyecto, después se obtiene el mapa y finalmente ya se puede proceder a operar con normalidad. Este proceso se ha diseñado así para dar solución al problema que supone cumplir el *REQ_17* del documento de especificación de requisitos, visto en el capítulo 4.1.

Al crear un proyecto es necesario que el usuario indique el lugar donde se guardará el proyecto, porque la descarga del mapa se realizará en ese directorio, para que pueda ser convertido a formato compatible con SUMO. Los mapas se pueden obtener importándolos de uno ya existente en el equipo, o bien descargándolos a través de la funcionalidad que permite descargar mapas del proyecto OpenStreeMap.

Sólo se permitirá obtener un mapa durante la fase de creación del proyecto. La forma de implementar esta característica es simple. Después de crear un proyecto nuevo, cuando es necesario descargar el mapa, se deshabilitan todas las funcionalidades que no sean importar o descargar mapas y no se habilitarán hasta que exista un mapa cargado en el programa. La razón de deshabilitar la opción de descargar mapa después de tener uno creado, es que todos los datos de vehículos funcionan sólo para un determinado mapa. Esta solución a priori puede parecer una desventaja pero en realidad es una ventaja. Es un modo de evitar que un usuario cambie el mapa una vez definido y deje inservible el proyecto. Sólo aclarar que el proyecto quedaría inservible porque los modelos de movilidad están relacionados directamente con los mapas, valiendo sólo para un mapa. El único inconveniente de esta forma de trabajar está en el caso de que un usuario descargue un mapa desde OpenStreetMap y luego se dé cuenta de que el mapa no es el mapa que estaba buscando. Si el problema radica en que necesita un mapa más grande o más pequeño, los datos almacenados se podrán guardar y luego ser importados a un nuevo proyecto sin ningún tipo de problema. El mayor problema que puede darse con este sistema es que se haya descargado un mapa con una ubicación radicalmente distinta. En tal caso no es posible hacer nada y el usuario debería rehacer su trabajo para el mapa nuevo.

5.2 Estructura de la aplicación

La estructura que se ha seguido para la construcción de C4R es la que se puede ver en la siguiente imagen:

Figura 20 *Estructura de Citymob for Roadmaps.*

Como se puede apreciar en la *Figura 20* la aplicación está diseñada siguiendo una arquitectura de tres capas donde la capa de presentación, la capa de negocios y la capa de datos están perfectamente separadas permitiendo un mejor mantenimiento. Las capas de presentación y datos como se verá posteriormente, son realmente sencillas por lo que no hace falta subdividirlas. En cuanto a la capa de negocio, sí que es necesario hacer una división funcional de sus componentes. Tal y como aparece en la figura, la lógica de negocio está dividida en tres partes. La primera parte, la calculadora de rutas, es la encargada de crear rutas consistentes, es decir, asegura que los vehículos sigan rutas acordes con la realidad y los parámetros introducidos por el usuario, evitando fenómenos que se pueden dar en las simulaciones, como por ejemplo la teleportación de vehículos entre calles. La segunda parte de la capa de negocio, la gestión de modelos de movilidad, se encarga de almacenar los datos obtenidos a través de la GUI y ponerlos a disposición del resto de la aplicación de un modo ordenado y consistente. La última parte, la gestión de formas, se refiere a la gestión de los elementos que aparecen dibujados en la interfaz. El objetivo de tener una serie de elementos que se encarguen de gestionar las formas es aumentar la mantenibilidad y extensibilidad de la aplicación. De esta forma, es posible cambiar la apariencia de los objetos mostrados en pantalla sin alterar el funcionamiento de la aplicación. Otra característica de este apartado es que sirve de enlace entre la GUI y los datos guardados en disco de forma que cualquier alteración en los mismos no obligaría a rediseñar los elementos de las restantes capas.

5.3 Interfaz gráfica

La interfaz gráfica es una parte muy importante en cualquier aplicación porque es la parte con la que el usuario trabaja. Esto tiene como consecuencia que una interfaz mal diseñada puede convertir a una aplicación con infinitas posibilidades en una aplicación de mala calidad, por el simple hecho de que la interfaz convierta la tarea a realizar en un auténtico infierno. Por eso hay que poner especial atención a la hora de diseñar la interfaz. En C4R, se ha llevado a cabo esta tarea con mucho cuidado poniendo especial atención a la mantenibilidad, puesto que al ser una aplicación que trabaja con otras aplicaciones, es susceptible de cambios. Por consiguiente, hay que minimizar el impacto y coste de futuros cambios en los requisitos.

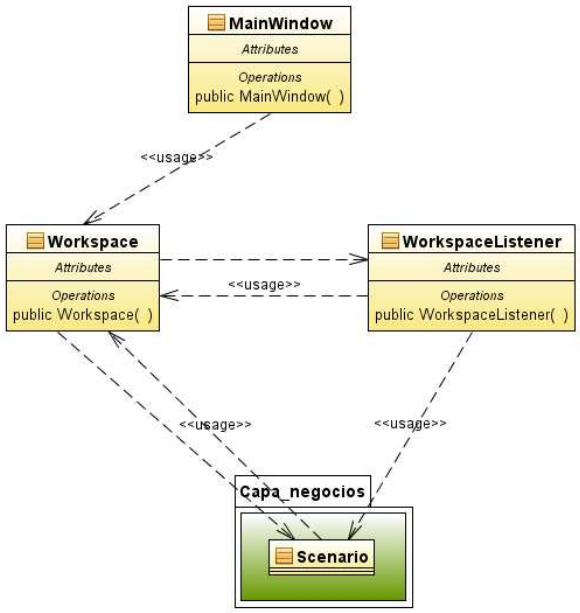

Figura 21 *Diagrama de clases de la ventana principal de Citymob for Roadmaps.*

La *Figura 21* corresponde al diagrama de clases de la ventana principal de C4R. La apariencia general de la ventana está desarrollada en la clase *MainWindow*. Esta clase tiene todos los componentes de la ventana y las llamadas correspondientes a la activación de cada componente. Los componentes almacenados son principalmente los botones de la barra de herramientas y menús, pero hay un componente especial que es importante, es el área de trabajo del usuario y ha sido diseñado siguiendo el patrón Modelo Vista Controlador (MVC) con el objetivo de optimizar la operabilidad. Tal como aparece en el diagrama, la vista corresponde a la clase *Workspace*. *Workspace* es el espacio de trabajo del usuario, el lugar donde se carga el proyecto y donde el usuario puede interactuar con él. Siguiendo el diagrama descrito en la *Figura 21*, se puede identificar la clase *WorkspaceListener* como el controlador y clase responsable de manejar todos los eventos que suceden en el área de trabajo. Finalmente, el modelo corresponde a la clase *Scenario*. Esta clase se puede considerar ya parte de la lógica de negocio, puesto que trabaja con datos lógicos e interactúa con las demás clases de la capa de lógica de negocio.

Otro aspecto de la GUI, son las distintos diálogos que aparecen en función de la opción elegida por el usuario. A la hora de diseñar estas ventanas se ha intentado aprovechar al máximo el código, pero hay ventanas en las que no ha sido posible hacerlo por su naturaleza. A pesar de esto, lo que sí se ha conseguido ha sido que todos los diálogos que sirven para archivos de SUMO y las vistas de datos, compartan código mediante el mecanismo de herencia tal como se puede apreciar en la siguiente figura.

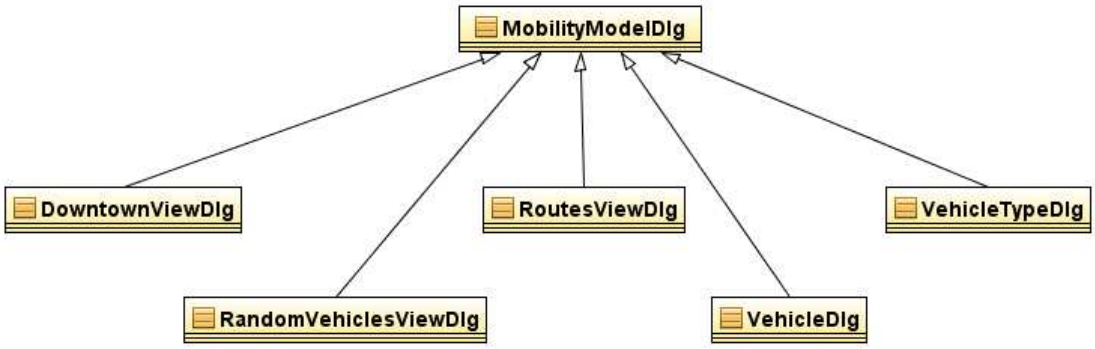

Figura 22 *Diagrama de clases de los diálogos de Citymob for Roadmaps.*

Heredando los distintos diálogos de uno común conseguimos mejorar el mantenimiento y que los diálogos tengan una apariencia uniforme, lo que supone una ayuda al usuario porque tarda menos tiempo en familiarizarse con el sistema.Teniendo en cuenta que la misión principal de estos diálogos es tomar datos del usuario, una tarea crítica que ha sido integrada en la interfaz es la validación de los datos. Los datos son tomados en distintos formularios y validados. En caso de que los datos fueran incorrectos, la GUI informa al usuario. En cambio si los datos son correctos, se cierra la ventana de diálogo. Este proceso se puede ver representado en el siguiente diagrama.

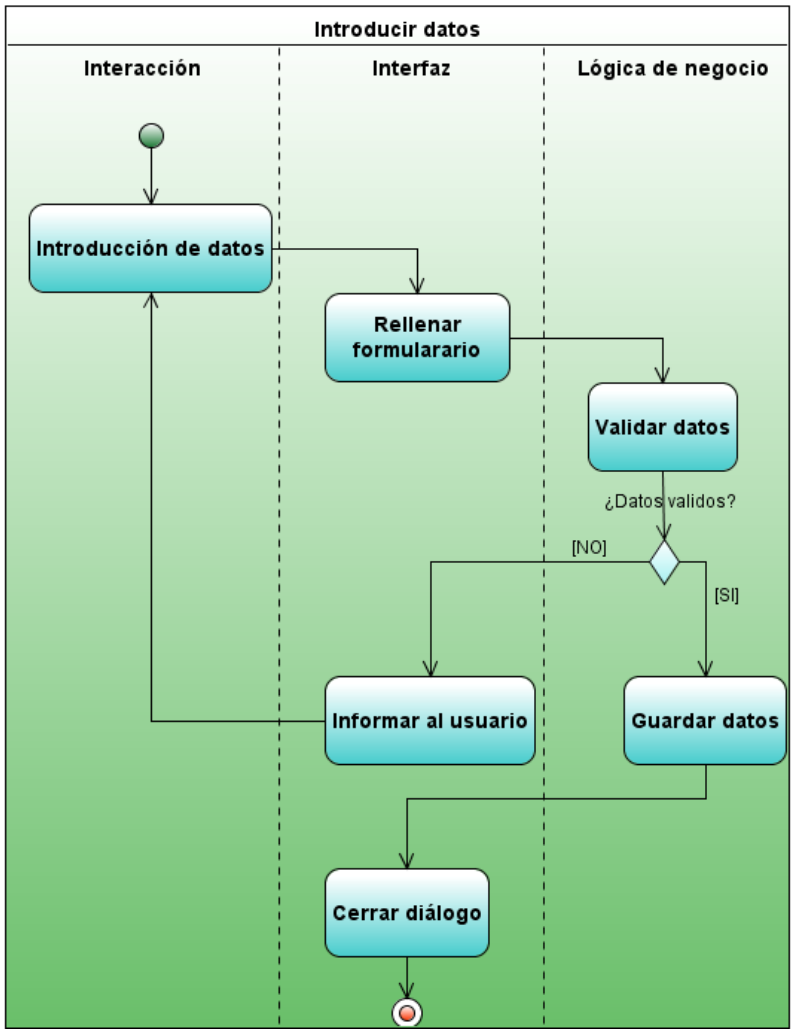

Figura 23 *Diagrama de actividades de la validación de datos.*

5.4 Lógica de negocio

Como se ha visto anteriormente la capa de lógica de negocio esta dividida de forma funcional en varios fragmentos. En este apartado se pretende ampliar la información sobre la lógica de negocio, profundizando en su diseño y las razones por las que se ha elegido ese diseño.

La primera parte a tratar es la de cálculo de rutas. A este fin se ha creado una clase llamada *RoutesCalculator* que se encarga de generar rutas a partir de los datos suministrados por el usuario. Esta calculadora tiene tres operaciones importantes. La primera genera una pareja de puntos aleatoriamente cumpliendo dos requisitos: que los dos puntos de la pareja no pueden ser el mismo y que la posición que ocupan en el mapa debe atender a los criterios probabilísticos relativos al valor de atracción de los distintos centros urbanos repartidos por el mapa. La segunda operación consiste en calcular la ruta más corta entre dos puntos. Finalmente, la tercera operación consiste en calcular la distancia que separa dos puntos por la ruta más corta. Estas tres operaciones además

necesitan una serie de parámetros para poder funcionar. Los únicos datos necesarios son: una representación del mapa en forma de grafo, los distintos centros urbanos y la información que facilita el usuario, relativa a los vehículos aleatorios. Una vez que se tiene todos estos datos, se vuelcan en un archivo para que SUMO pueda crear la traza.

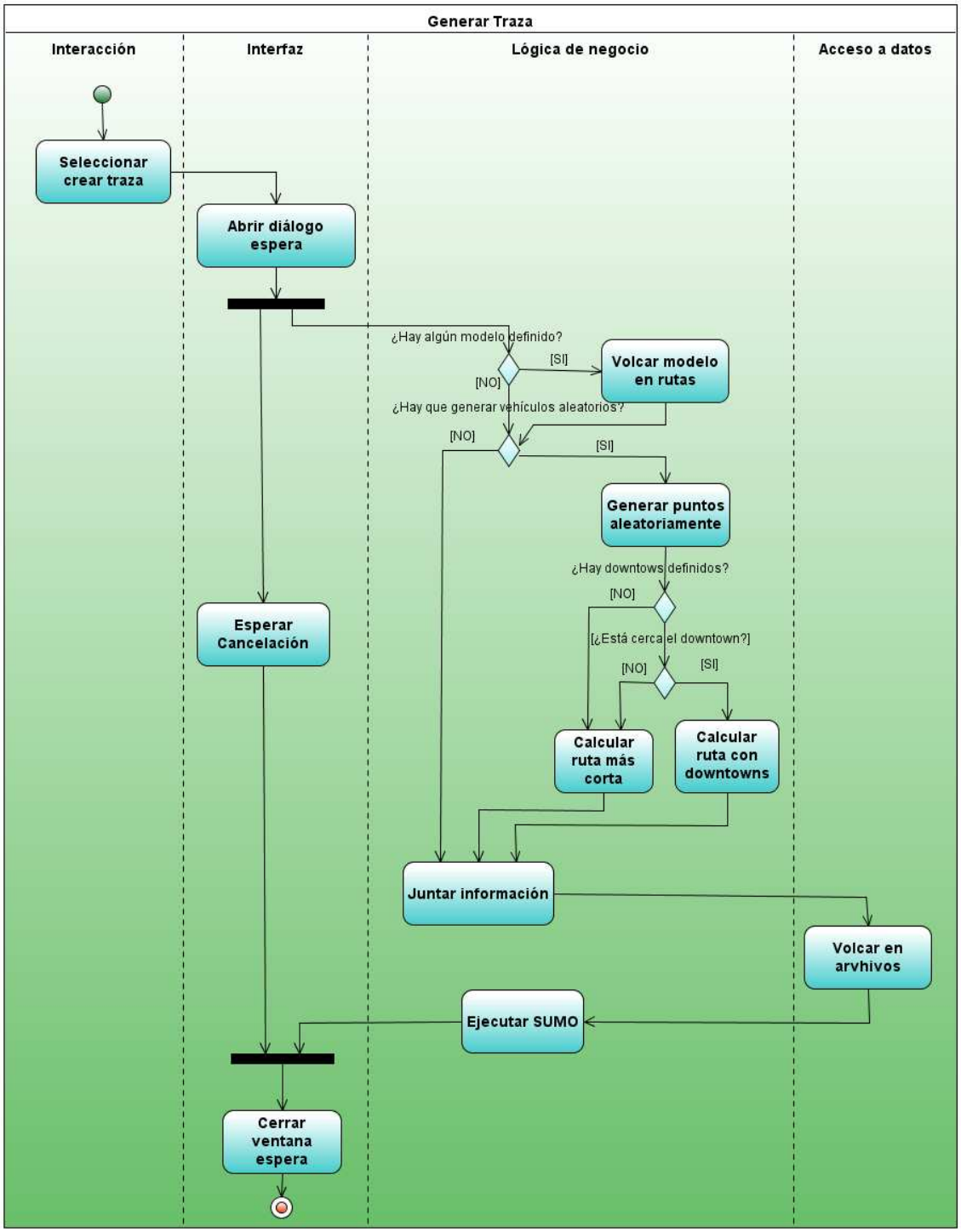

Figura 24 *Diagrama de actividades de la creación de rutas aleatorias.*

A pesar de la claridad de los diagramas de actividad hay algunos aspectos clave relativos a la implementación que no quedan reflejados.

Cada ruta aleatoria es generada individualmente, y el proceso que se sigue consta de varios pasos. El primero consiste en generar un origen y un destino de ruta. Una vez que tenemos un origen, se calcula el camino más corto mediante el algoritmo de Dijkstra. Llegados a este punto, hay que comprobar si alguno de los downtowns está situado cerca de la ruta más corta. Esta última acción implica que hay que definir cerca. Para esta versión del algoritmo de creación de rutas se ha definido cerca arbitrariamente, como una ruta tal que pasa por al menos un downtown y la distancia total que recorre el vehículo es menor que el doble de la distancia que recorre con el camino más corto. Esta definición ha sido implementada en una función de forma que en futuras versiones resulte cómodo cambiar la definición a otras necesidades. Volviendo al proceso, se observa uno a uno cada centro urbano si está cerca. Si no lo está, la ruta es el camino más corto, si lo está entonces se calcula la probabilidad ponderada de las distintas rutas y dejándolo preparado para que SUMO las simule aleatoriamente de acuerdo a los cálculos del algoritmo.

La segunda parte de la lógica de negocio se refiere al almacén de los datos obtenidos del usuario y su transformación en datos útiles para poder crear trazas. Para llevar a cabo esta tarea se ha dividido esta parte de la capa en paquetes.

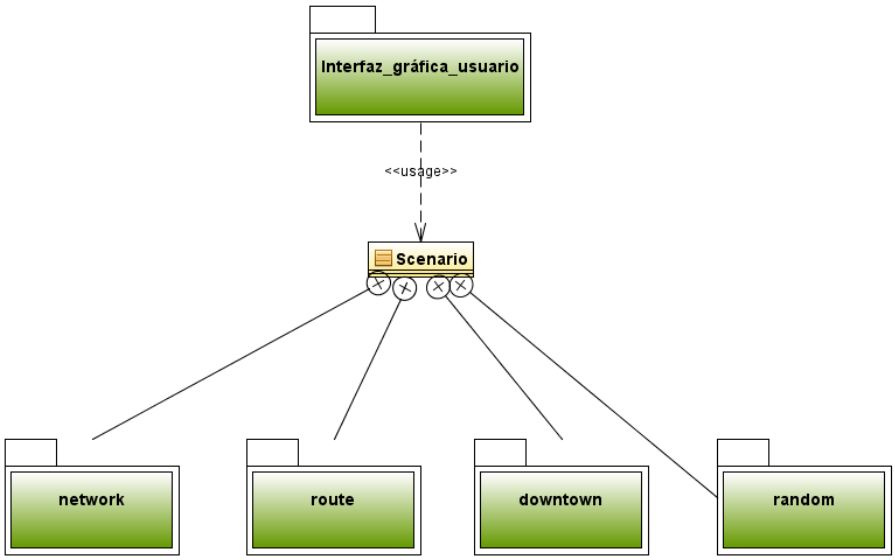

Figura 25 *Diagrama de paquetes del almacén de datos.*

Tal como se aprecia en la *Figura 25*, cada uno de los paquetes está integrado dentro de la clase *Scenario*. Esta característica es debida a que la gestión de los contenedores de información es llevada a cabo por *Scenario*. Otra característica es la estructura de los paquetes, la cual se muestra en la siguiente imagen.

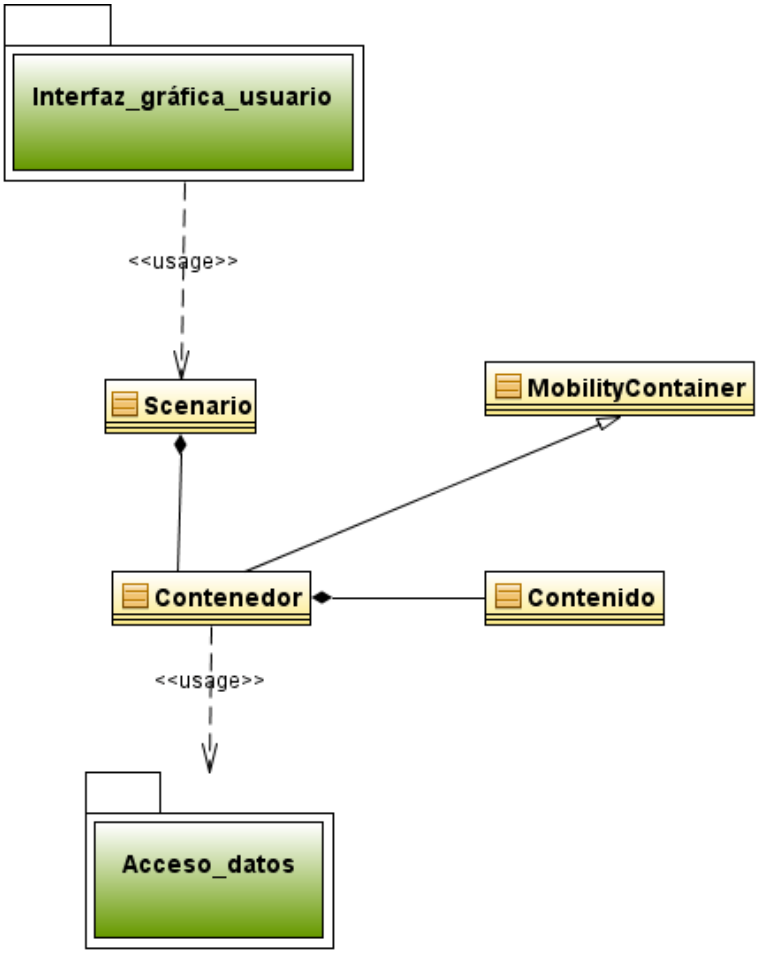

Figura 26 *Diagrama de clases de los paquetes del almacén de datos.*

Como se puede comprobar la estructura de la *Figura 26* es muy simple. Esta estructura se compone de una clase contenedor que hereda de la clase *MobilityContainer*, que ofrece atributos y métodos que comparten todos los contenedores. También hay una clase contenido que es cada uno de los elementos que se guardan en un tipo de contenedor. Otra característica importante es que la relación que tiene una clase contenedor con *Scenario* es de 1:1 porque no existen contenedores de contenedores. Finalmente, es necesario señalar que todos los paquetes que aparecen en la *Figura 26* tienen esta estructura. Esta decisión de diseño fue tomada pensando en la extensibilidad. Usando esta estructura es posible crear un paquete totalmente independiente e integrarlo en la aplicación en poco tiempo, la única condición es que tiene que heredar de *MobilityContainer*.

La última parte de la lógica de negocio corresponde a la gestión de figuras. La forma en la que está diseñado es muy simular a la vista en los paquetes de la gestión de datos sólo que con alguna pequeña variación.

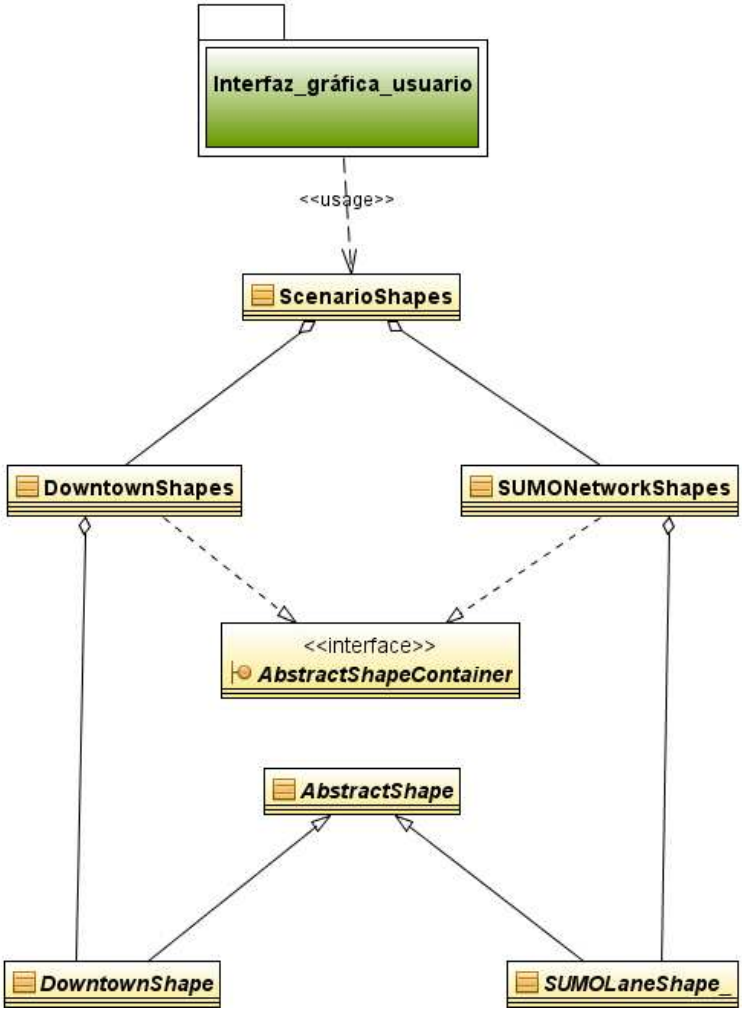

Figura 27 *Diagrama de clases de la gestión de formas.*

Observando el diagrama de clases se puede ver el parecido con la *Figura 26*. Este conjunto tiene una clase llamada *ScenarioShapes*, cuya función es servir de unión y gestionar las figuras de cada proyecto en conjunto. Esta clase, a su vez, contiene referencias a las clases *DowntownShapes* y *SUMONetworkShapes*. El objetivo de estas dos clases es permitir el almacenamiento de un tipo concreto de figuras. En este caso, para servir de contenedor de figuras es necesario implementar la interfaz *AbstractShapeContainer* que contiene la definición de los métodos necesarios para poder representar cualquier conjunto de figuras. Hasta el momento C4R sólo representa dos tipos de figuras, los centros urbanos y los carriles, que están representadas por *DowntownShape* y *SUMOLaneShape* respectivamente. Cada una de estas figuras tiene que heredar de *AbstractShape* para aprovechar al máximo el código y evitar problemas a la hora de dibujar una figura en el área de trabajo.

Este diseño tiene como ventaja que permite añadir una nueva figura o modificar una existente de un modo sencillo y sin afectar al resto de la aplicación. De esta forma es posible cambiar la apariencia de de una forma rápida y elegante.

5.5 Acceso a datos

Como se ha descrito anteriormente, en C4R los datos se guardan en archivos XML que son almacenados en una carpeta que elige el usuario. Este sistema ha sido elegido por su sencillez, comodidad y compatibilidad con SUMO de forma automática. Por este motivo, no es previsible que la forma de almacenamiento varíe, pero aún así es necesario prever un posible cambio para evitar problemas durante la fase de mantenimiento.

El sistema de acceso a datos, al ser tan simple, se compone única y exclusivamente de dos clases muy sencillas.

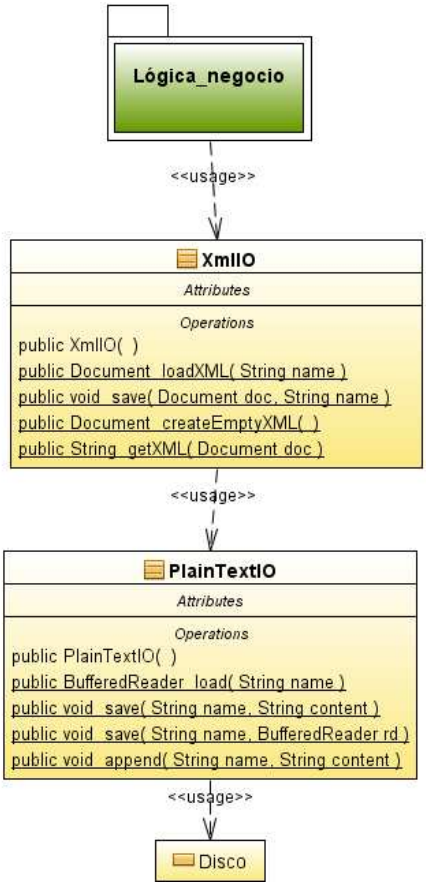

Figura 28 *Diagrama de clases del acceso a datos.*

Como se puede apreciar, las dos clases están compuestas por métodos estáticos, que permiten ser invocados sin tener que instanciar el objeto. El modo de funcionamiento es muy simple tanto para guardar como para cargar archivos. En primer lugar para guardar información en formato XML, un método cualquiera de la lógica de negocio llama al método *save*, facilitando el nombre de archivo y el contenido a guardar. En esta clase se añaden datos de cabecera y finalmente se llama a *save* en *PlainTextIO* que se encarga de guardarlo en disco. El proceso es distinto para cargar un archivo debido al funcionamiento de la librería SAX en Java. Para leer un archivo XML hay que llamar al método *loadXML* de *XmlIO*. Si el texto es plano, se puede leer análogamente llamando al método *load* de la clase *PlainTextIO*.

5.6 Asistente

Una parte importante de la aplicación es el asistente. El asistente tiene un diseño que merece la pena ser explicado, porque debe trabajar en las tres capas y aprovechar el código de las tres. El asistente está concebido como una simplificación de la interfaz de modo que el usuario puede crear un modelo de movilidad de una forma rápida y sin conocer la aplicación. A este fin, se ha diseñado una parte de la interfaz para que el usuario se pueda mover por ella lo más libremente posible y que pueda desarrollar un modelo de movilidad lo más completo posible.

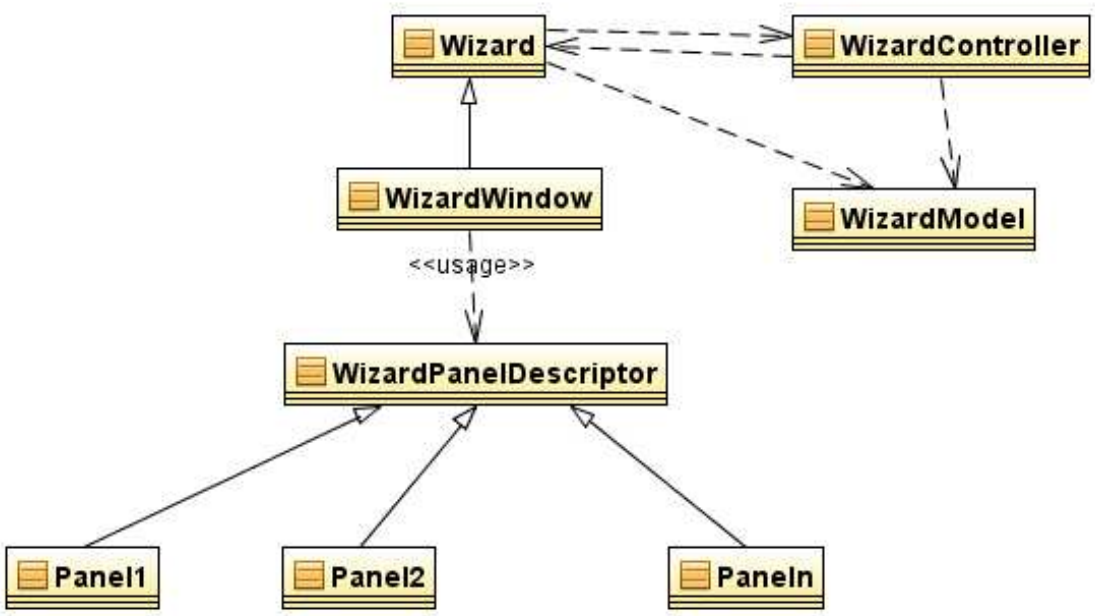

Figura 29 *Diagrama de clases del asistente.*

Como se puede observar en la *Figura 29*, el asistente ha sido diseñado siguiendo un patrón MVC, donde la clase *WizardModel* tiene el rol de modelo, la clase *Wizard* tiene el rol de vista y la clase *WizardController* el de controlador. La clase *WizardWindow* configura la apariencia de la ventana del asistente. Siguiendo esta estructura, se puede reutilizar el código de manera que se puede crear cualquier tipo de asistente, creando una clase nueva, al igual que se ha hecho con *WizardWindow*. La clase WizardPanelDescriptor se encarga de crear una descripción de los distintos paneles que irán apareciendo a lo largo del asistente. La idea es que cada vez el usuario avance o retroceda, vea una serie de paneles con características similares y que guarden la información, de tal forma que si se vuelve sobre sus pasos sigan en los formularios los mismos datos que se facilitaron, sin tener que volver a introducirlos otra vez. Para crear un panel nuevo sólo hay que hacer que herede de la clase *WizardPanelDescriptor* y establecer los criterios de cuándo aparecerá en pantalla.

6. Manual de usuario

En este manual se explica cómo usar Citymob for Roadmaps y muestra cómo explotar al máximo las posibilidades del programa. Dentro del manual están todos los pasos para crear trazas de movilidad y poder usarlas para realizar simulaciones. Hay que dejar claro que este manual trata sólo del uso del software.

6.1 Citymob for Roadmaps

Citymob for Roadmaps (C4R) es un generador de trazas de movilidad para redes de vehículos. Esto significa que es un software que permite simular el tráfico en distintas localidades, valiéndose para ello, de mapas de ubicaciones reales. Citymob for Roadmaps es una herramienta software libre, ya que su finalidad es ser usado por la comunidad científica. Se permite tener acceso al código fuente para determinar la validez de sus algoritmos y a la vez, permite realizar adaptaciones y correcciones por los propios usuarios si son requeridas.

6.2 Licencia

La licencia de Citymob for Roadmaps es la GNU General Public License versión 3 [GPL3]. Se ha elegido este tipo de licencia porque facilita la distribución del software cumpliendo el objetivo de llegar a todos los usuarios que sea posible. Además este tipo de licencia obliga a distribuir una copia del código fuente con el binario de modo que la comunidad científica puede determinar la validez de la solución. Por si esto fuera poco, también se permite modificar el código fuente, lo que permite mejorar la aplicación por terceras personas que no estén vinculadas con la aplicación.

Los términos de uso completos de la licencia se pueden encontrar en la carpeta de C4R en un archivo llamado “license.txt”. También es posible encontrar una traducción no oficial (sin validez legal) al castellano en el archivo “licencia.txt”. También es posible encontrar el *disclaimer* en el apartado *about* de la aplicación y a continuación:

> *Citymob for Roadmaps (C4R) is a software which uses SUMO to create mobility traces based on real maps extracted from OpenStreetMap.*
> *It is specially designed for VANET research.*
>
> *Copyright (C) 20011 Alberto Pardo Calvo*
>
> *This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.*

This program is distributed in the hope that it will be useful,but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS

FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Linking Citymob for Roadmaps statically or dynamically with other modules is making a combined work based on Citymob for Roadmaps. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

In addition, as a special exception, the copyright holders of Citymob for Roadmaps give you permission to combine Citymob for Roadmaps program with free software programs or libraries that are released under the GNU LGPL and with code included in the standard release of Citymob for Roadmaps under the GPL license (or modified versions of such code, with unchanged license). You may copy and distribute such a system following the terms of the GNU GPL for Citymob for Roadmaps and the licenses of the other code concerned, provided that you include the source code of that other code when and as the GNU GPL requires distribution of source code.

Note that people who make modified versions of Citymob for Roadmaps are not obligated to grant this special exception for their modified versions; it is their choice whether to do so. The GNU General Public License gives permission to release a modified version without this exception; this exception also makes it possible to release a modified version which carries forward this exception.

IMPORTANTE: Esta traducción tiene carácter meramente informativo y carece de validez legal.

Citymob for Roadmaps (C4R) es un programa que usa SUMO para crear trazas de movilidad basadas en mapas reales extraídos de OpenStreetMap. Está especialmente diseñado para la investigación de redes VANET.

Copyright (C) 2011 Alberto Pardo Calvo

Este programa es software libre. Puede redistribuirlo y/o modificarlo bajo los términos de la Licencia Pública General de GNU según es publicada por la Fundación para el Software Libre, bien de la versión 2 de dicha Licencia o bien (según su elección) de cualquier versión posterior.

Este programa se distribuye con la esperanza de que sea útil, pero SIN NINGUNA GARANTÍA, incluso sin la garantía MERCANTIL implícita o sin garantizar la CONVENIENCIA PARA UN PROPÓSITO PARTICULAR. Véase la Licencia Pública General de GNU para más detalles.

Debería haber recibido una copia de la Licencia Pública General junto con este programa. Si no ha sido así escriba a la Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.

Al enlazar Citymob for Roadmaps estática o dinámicamente con otros módulos se está haciendo una obra combinada basada en Citymob for Roadmaps. Así, pues, los términos y condiciones de la Licencia Pública General de GNU cubren la combinación en conjunto.

Además, como excepción especial, los titulares del copyright de Citymob for Roadmaps le conceden permiso para combinar el programa Citymob for Roadmaps con programas de software libre o bibliotecas publicadas bajo la LGPL de GNU y con código incluido en la edición estándar de Citymob for Roadmaps bajo la licencia GPL (o versiones modificadas de dicho código, siempre que la licencia no haya cambiado). Puede copiar y distribuir el sistema así construido, según los términos de la GPL de GNU para Citymob for Roadmaps, y las licencias del resto del código implicado, siempre que incluya el código fuente de ese otro código, en el momento y de la manera en que la GPL de GNU exija la distribución del código fuente.

Advierta que quien haga modificaciones de Citymob for Roadmaps no está obligado a garantizar esta excepción especial en sus versiones modificadas; queda a su discreción hacerlo o no. La Licencia Pública General de GNU concede permiso para publicar una versión modificada sin esta excepción; esta excepción hace también posible publicar una versión modificada que contenga dicha excepción.

6.3 Características

El usuario podrá encontrar usando Citymob for Roadmaps las siguientes características:

- Sistema multiplataforma (Windows y Linux).
- Generación de modelos mediante una potente interfaz gráfica.
- Trabajar con mapas reales.
- Vehículos con rutas definidas por el usuario.
- Vehículos con rutas aleatorias.
- Flujos de vehículos entre dos puntos.
- Puntos de atracción.
- Movimiento de los vehículos definido por un determinado modelo.
- Modelos de movilidad creados mediante un cómodo asistente.
- Crear trazas de movilidad.
- Exportar trazas de movilidad a formato NS-2.
- Exportar proyecto a formato SUMO.
- Importar archivos de otros proyectos de Citymob for Roadmaps existentes.
- Visualizar la simulación.

Citymob for Roadmaps permite simular el movimiento de vehículos. Para esta tarea se vale de un sistema para generar rutas de forma aleatoria o usar rutas definidas por el usuario. Igualmente, también permite señalar puntos con tendencia a concentrar tráfico. Finalmente, este generador crea archivos con trazas de movimiento y permite visualizar la traza de movimiento.

6.4 Requisitos mínimos

Componente	Requisitos mínimos
Procesador	Procesador Pentium III, compatible con PC a 600 MHz
Disco duro	10 GB
RAM	1.5 GB
Resolución de pantalla	1024x768
Sistema operativo	Windows o Linux
Versión de java runtime	Java 6 SE
Versión de SUMO recomendada	0.11.1
Conexión a Internet	Conexión de 56K

Tabla 9 *Requisitos mínimos para ejecutar Citymob for Roadmaps*

La capacidad mínima del disco duro puede ser mayor en algunos casos. En algunas circunstancias, si se van a crear modelos en mapa grandes y con muchos vehículos, el tamaño de la traza puede ser superior a 10 GB, pero no es frecuente.

6.5 Instalación

Instalación en Windows

1. Abrir el CD.
2. Copiar la carpeta C4R donde se desee.

Intalación en Linux

Para llevar a cabo la instalación es necesario tener instalados los siguientes paquetes: Proj, GDAL, Xerces-c, Fox. En caso de no tenerlos instalados, lo recomendable es instalarlos desde los repositorios. Si no están en los repositorios se pueden conseguir en el CD, en la carpeta llamada **Libraries**

1. Entrar desde el intérprete de comandos a la carpeta del CD llamada SUMO
2. tar xzf sumo-src-0.11.1.tar.gz
3. cd sumo-0.11.1
4. ./configure
5. make
6. Copiar la carpeta C4R a donde se desee.

6.6 Ejecutando Citymob for Roadmaps

Lo primero es abrir la aplicación y familiarizarse con el entorno. La ventana está dividida en varias zonas, las cuales, van a ser nombradas a continuación. Estos son los nombres que se van a usar a lo largo del manual de usuario.

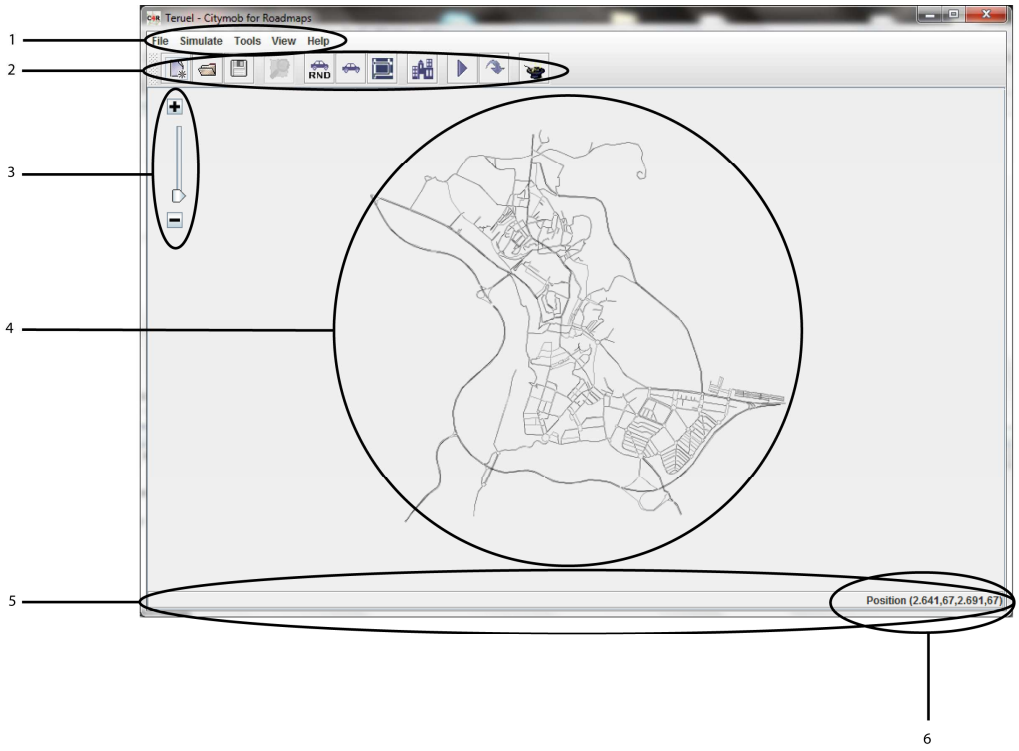

Figura 30 *Composición de la ventana principal de Citymob for Roadmaps.*

Áreas de la ventana principal:

1. Menú de opciones
2. Panel de botones
3. Barra de zoom
4. Área de trabajo
5. Barra de estado
6. Posición del cursor

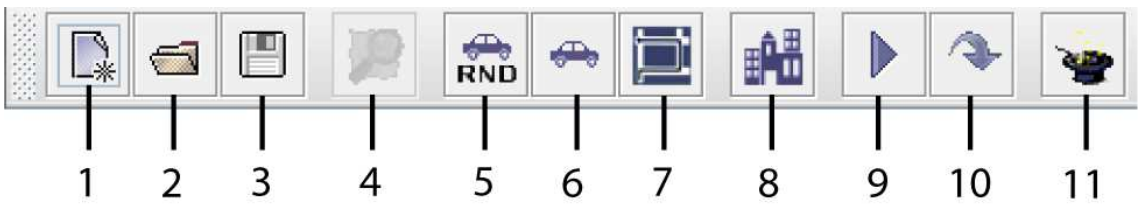

Figura 31 *Detalle de la barra de botones de Citymob for Roadmaps.*

Nombre de los botones:

1. Nuevo proyecto
2. Abrir proyecto
3. Guardar proyecto
4. Seleccionar mapa de OpenStreetMap
5. Añadir vehículos aleatorios
6. Añadir vehículos manualmente
7. Añadir rutas manualmente
8. Añadir puntos de atracción
9. Visualizar traza
10. Crear traza
11. Abrir asistente

6.6.1 Comprobar configuración

La primera tarea que hay que realizar antes de crear cualquier modelo de movilidad es comprobar que la aplicación está configurada correctamente para trabajar con todos sus componentes. Para ello hay que seguir los siguientes pasos:

1. Clic en menú **Tools**, opción **Options.**
2. En caso de disponer de un sistema operativo Windows hay que comprobar que en la caja de texto de **SUMO installation path** aparece la ruta de instalación de SUMO. De no ser así hay que hacer clic en el botón con un icono de carpeta y buscar la ubicación de SUMO.
3. Comprobar que en la caja de texto de **OpenStreetMap API** hay un texto como este: "http://api.openstreetmap.org/api/0.6/map?bbox=". Esta es la API usada actualmente. En caso de que cambie la API, indicar la nueva. Ésta se puede obtener buscándola en http://wiki.openstreetmap.org/wiki/API.
4. Pulsar **Accept.**

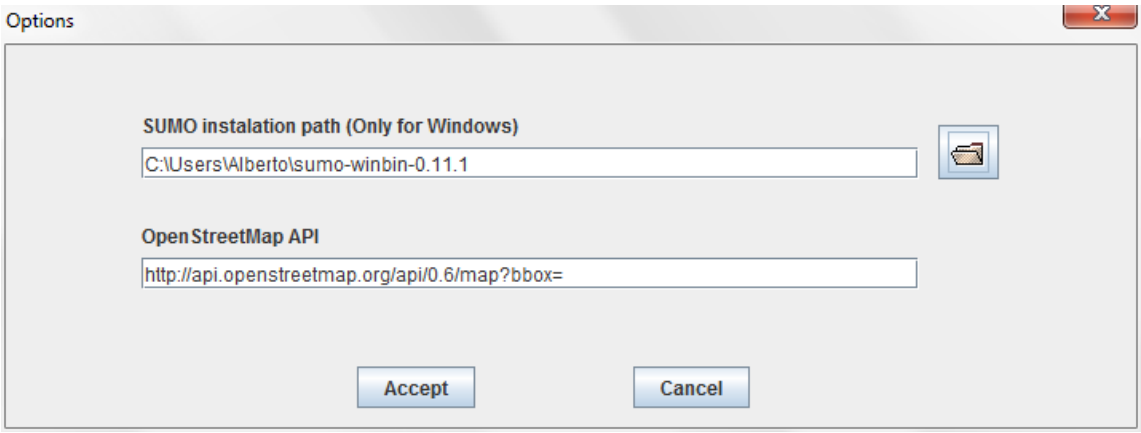

Figura 32 *Detalle del diálogo de opciones de Citymob for Roadmaps.*

6.6.2 Crear proyecto nuevo

1. Hacer clic en el botón **Nuevo proyecto** o menú **File** opción **New project**.
2. Seleccionar la ubicación del nuevo proyecto.
3. Seleccionar un nombre para el proyecto.
4. Pulsar **Guardar**.

Existe un error con la versión 0.11.1 de SUMO que impide trabajar con rutas que tienen al menos un carácter de espacio. Por este motivo es altamente recomendable evitar carpetas que contengan espacios en su ruta. Este problema está solucionado en la 0.12.2 pero aparecen otros nuevos, por lo que no se recomienda migrar de versión.

6.6.3 Abrir proyecto

1. Hacer clic en el botón **Abrir proyecto** o menú **File** opción **Open project**.
2. Buscar el archivo de proyecto guardado previamente.
3. Pulsar **Abrir**.

6.6.4 Guardar proyecto

1. Hacer clic botón **Guardar proyecto** o menú **File** opción **Save**.

6.6.5 Importar archivo

Esta opción permite añadir archivos de otros proyectos, ya sean de SUMO o de Citymob for Roadmaps al proyecto actual. Para ello bastará con seguir estos pasos:

1. Hacer clic en el menú **File** opción **Import.**
2. Ir a la ubicación donde esté el archivo que se quiera importar y seleccionarlo.
3. Pulsar **Abrir.**

6.6.6 Exportar archivo

Esta opción permite exportar un proyecto a formato SUMO. Para realizar esta tarea hay que hacer lo siguiente:

1. Hacer clic en el menú **File** opción **Export**.
2. Seleccionar el lugar donde se quiere ubicar el archivo.
3. Escribir nombre para el proyecto exportado.
4. Pulsar **Guardar**.

6.6.7 Salir

Siempre que se pretenda salir de la aplicación o cerrar el proyecto actual, Citymob for Roadmaps pedirá al usuario una confirmación preguntando además si desea guardar los cambios.

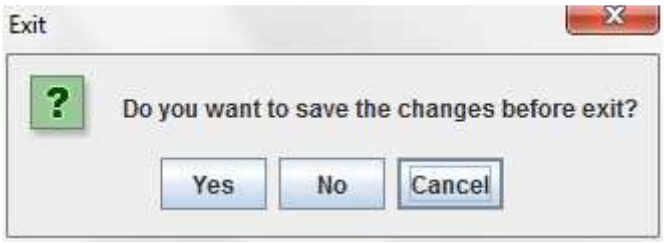

Figura 33 *Detalle de diálogo al salir de Citymob for Roadmaps.*

Como se puede apreciar en la *Figura 33* hay varias opciones cada una de las cuales se detalla a continuación:

1. Yes: guarda los cambios y cierra el proyecto actual.
2. No: no guarda cambios y cierra el proyecto actual.
3. Cancel: cancela la acción de cerrar el proyecto.

6.6.8 Cargar mapa en el proyecto actual

Para realizar este proceso previamente hay que crear un proyecto nuevo y no haber cargado un mapa para este mismo proyecto. Puede también cargar un mapa importándolo de otro proyecto haciendo uso de la opción **Import** explicada anteriormente. También puede descargar el mapa de OpenStreetMap.

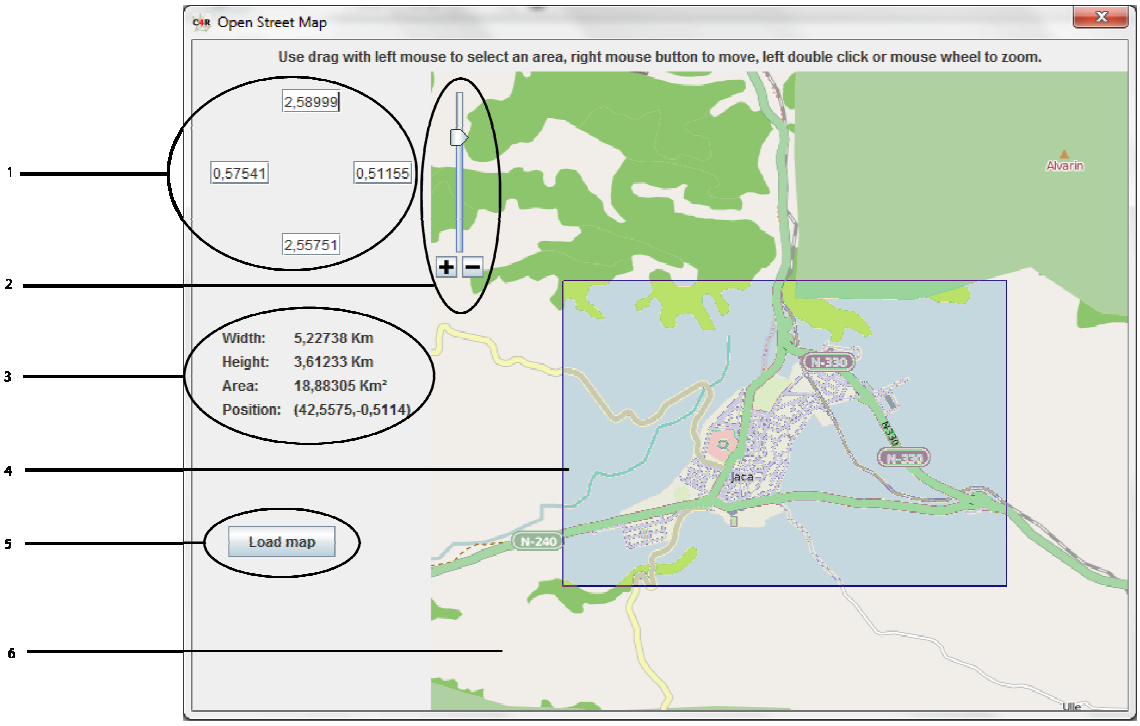

Figura 34 *Ventana de descarga de mapa de Citymob for Roadmaps.*

1. Panel de coordenadas de los límites de la selección
2. Barra de zoom
3. Panel de información sobre la selección
4. Selección
5. Botón de cargar mapa
6. Mapa

Para descargar un mapa de OpenStreetMap hay que seguir los siguientes pasos:

1. Hacer clic en el botón **Seleccionar mapa de OpenStreetMap** o seleccionar menú **Tools** opción **Select map**.
2. Moverse por el mapamundi en busca del lugar deseado. Para moverse arriba, abajo, izquierda, derecha, hay que arrastrar el cursor manteniendo el botón derecho pulsado. Para acercarse o alejase se puede usar la barra de zoom o la ruleta del ratón.
3. Una vez en la posición adecuada y con el zoom adecuado puede seleccionarse la parte que se desee para descargarlo al proyecto. Es importante tener en cuenta que OpenStreetMap no permite descargar mapas de cualquier tamaño, por lo que si el zoom no es lo suficientemente grande no permitirá descargar el mapa. Existen dos formas para seleccionar una porción de mapa. La manera más simple de realizar una selección consiste en arrastrar el cursor con el botón izquierdo pulsado. Cuando esto suceda, la parte seleccionada se coloreará con un tono azulado. Otra forma, consiste en introducir en el panel de coordenadas las longitudes y latitudes límite para la selección. Las cajas de texto coinciden con los puntos geográficos de modo que la caja de más arriba será la latitud de corte por el norte, la caja de la izquierda será la longitud de corte por oeste, la caja de la derecha será la longitud de corte por el oeste y la caja de abajo será la latitud de corte por el sur.
4. Una vez realizada la selección se pulsa el botón **Load map**. Acto seguido aparecerá un diálogo de descarga del mapa que permite su cancelación en cualquier momento. Una vez terminada la descarga se podrá ver el mapa dibujado en el área de trabajo.

6.6.9 Añadir vehículos aleatoriamente

1. Hacer clic en el botón **Añadir vehículos aleatorios** o en menú **Tools** opción **Add vehicles randomly**.
2. Introducir el número de vehículos a generar.
3. Indicar la tasa de vehículos que aparecerán dentro de un downtown (debe ser un número comprendido entre 0 y1).
4. Indicar el momento de aparición de los vehículos en la simulación.

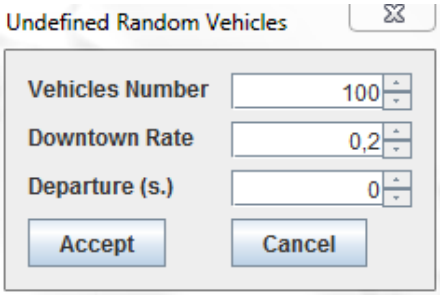

Figura 35 *Detalle del diálogo de vehículos aleatorios.*

6.6.10 Añadir vehículos aleatorios definidos

1. Hacer clic en el menú **Tools** opción **Add defined vehicles randomly.**
2. Selecionar una **Start position**. Se puede hacer escribiendo el valor para el carril en la correspondiente caja o haciendo clic sobre el carril en el mapa. Para saber que la selección ha sido efectuada correctamente se comprobará observando que en la caja ha aparecido el nombre del carril seleccionado y que el carril ahora es de color azul con una bandera roja en el extremo.
3. Realizar lo mismo que en el paso anterior pero para **End position.**
4. Determinar el número de vehículos a generar.
5. Tiempo de aparición de los vehículos en la simulación.
6. Pulsar el botón **Add vehicle**.
7. Si se quieren introducir más vehículos se puede volver a repetir el proceso desde el paso 1. Una vez acabado se pulsa el botón **Close Panel.**

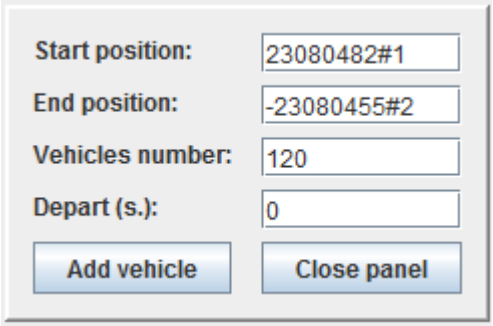

Figura 36 *Panel para insertar vehículos aleatorios definidos.*

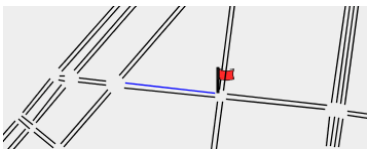

Figura 37 *Detalle de un carril seleccionado.*

6.6.11 Crear tipos de vehículos

1. Hacer clic en el menú **Tools** opción **Add vehicle type.**
2. Completar los datos de la tabla. Merece hacer especial mención al parámetro sigma que es un valor entre 0 y 1 que mide la imperfección del conducto al volante.
3. Para añadir más tipos pulsar **Add.**
4. Para eliminar tipos, seleccionar una tupla y pulsar **Remove.**
5. Pulsar **Accept.**

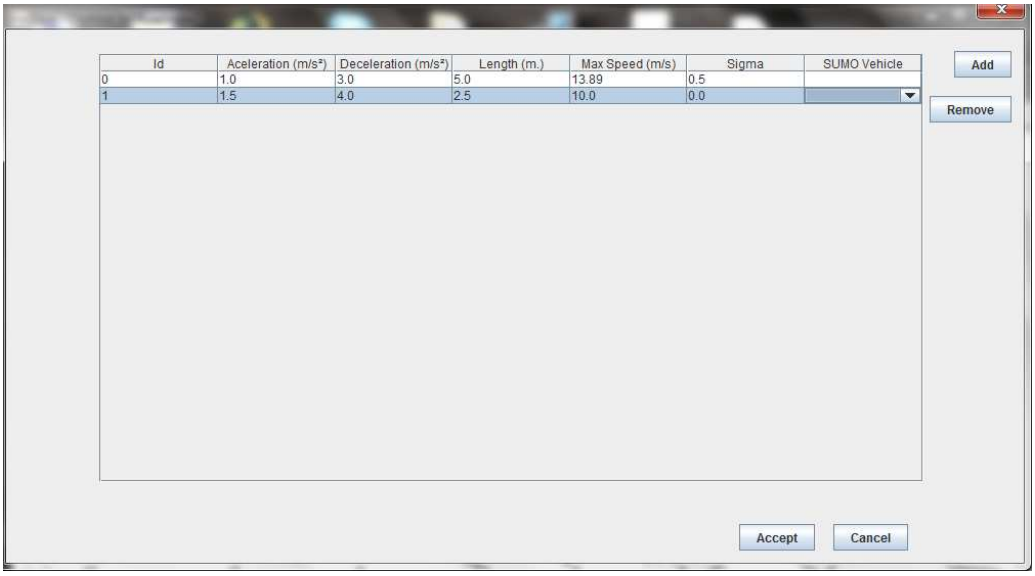

Figura 38 *Diálogo para insertar tipos de vehículos.*

6.6.12 Crear rutas manualmente

1. Hacer clic en el botón **Añadir rutas manualmente** o menú **Tools** opción **Add route.**
2. Introducir un identificador de ruta.
3. Seleccionar un carril del combobox y pulsar **Add.**
4. Si se quiere borrar un carril se pulsa el botón **Remove**. Este botón sólo borra el último carril para evitar inconsistencias.
5. Repetir el paso 3 hasta que se alcance el fin de la ruta.
6. Pulsar **Accept.**

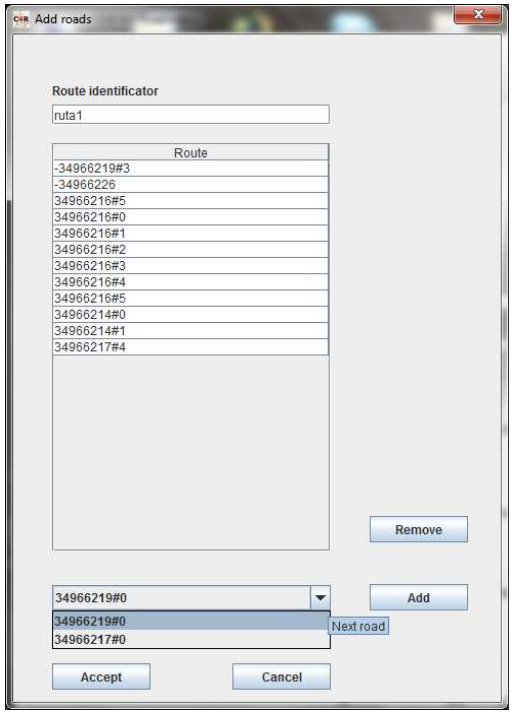

Figura 39 *Diálogo para introducir rutas manualmente.*

6.6.13 Definir vehículos manualmente

1. Hacer clic en el botón **Añadir vehículos manualmente** o en menú **Tools** opción **Add vehicle.**
2. Rellenar los campos que aparecen en la tabla.
3. Si se quieren añadir más vehículos, puede hacerse presionando el botón **Add**.
4. Para borrar un vehículo de la tabla hay que seleccionar la tupla que se quiera borra y pulsar **Remove.**
5. Pulsar **Accept**.

6.6.14 Añadir centros de atracción

1. Hacer clic en el botón **Añadir puntos de atracción** o en el menú **Tools** opción **Add downtown**. En ese momento cambiará la forma del cursor activándose el modo downtown.
2. Para crear un centro de atracción hay que arrastrar el cursor por el área donde se quiera crear.
3. Rellenar el diálogo con un identificador y el valor de atracción del centro de atracción. El valor de atracción es un número entre 0 y 1.
4. Pulsar **Accept**.

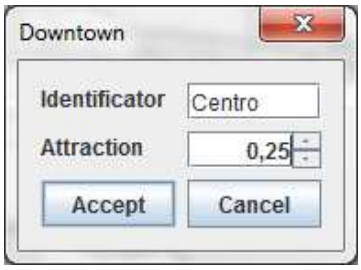

Figura 40 *Diálogo pare crear un centro de atracción.*

Figura 41 *Detalle del aspecto de un downtown en C4R.*

En la *Figura 41* se puede ver el recuadro que representa un downtown en el mapa. Hay que extremar la precaución al crear un downtown, puesto que está formado por todos los carriles que están dentro del recuadro aunque no estén contenidos por completo. Es decir, si se selecciona parcialmente un carril, automáticamente incluye el carril completo en el downtown.

6.6.15 Seleccionar modelo

1. Hacer clic en el menú **Tools** opción **Add mobility model.**
2. Seleccionar un modelo de la lista.
3. En función del modelo seleccionado se pedirán distintos valores mediante cajas de texto. En cualquier caso hay que dar los valores que se crean oportunos.
4. Pulsar **Accept**.

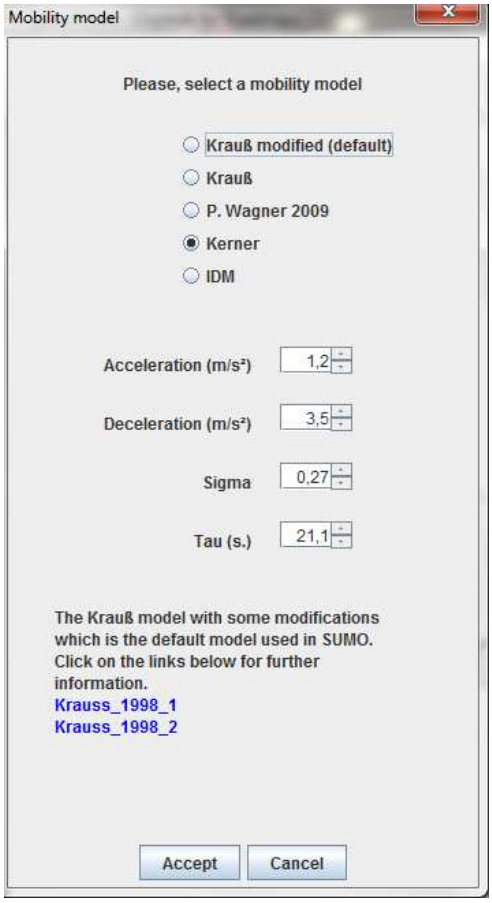

Figura 42 *Diálogo para seleccionar un modelo de movilidad.*

6.6.16 Vistas

En el menú **View** hay varias opciones. Todas las opciones tienen en común que funcionan de la misma manera. La única diferencia radica en el tipo de datos que presenta la vista. Las vistas permiten ver y borrar datos, pero no permiten añadir. Para añadir existen los procedimientos que han sido explicados en apartados anteriores. El procedimiento que se va a explicar es común para todas las vistas.

1. Seleccionar menú **View** y la opción con el tipo de datos que se está buscando.
2. Para borrar algún dato, hay que seleccionar la fila que se desea borrar y pulsar **Remove**.
3. Pulsar **Accept**.

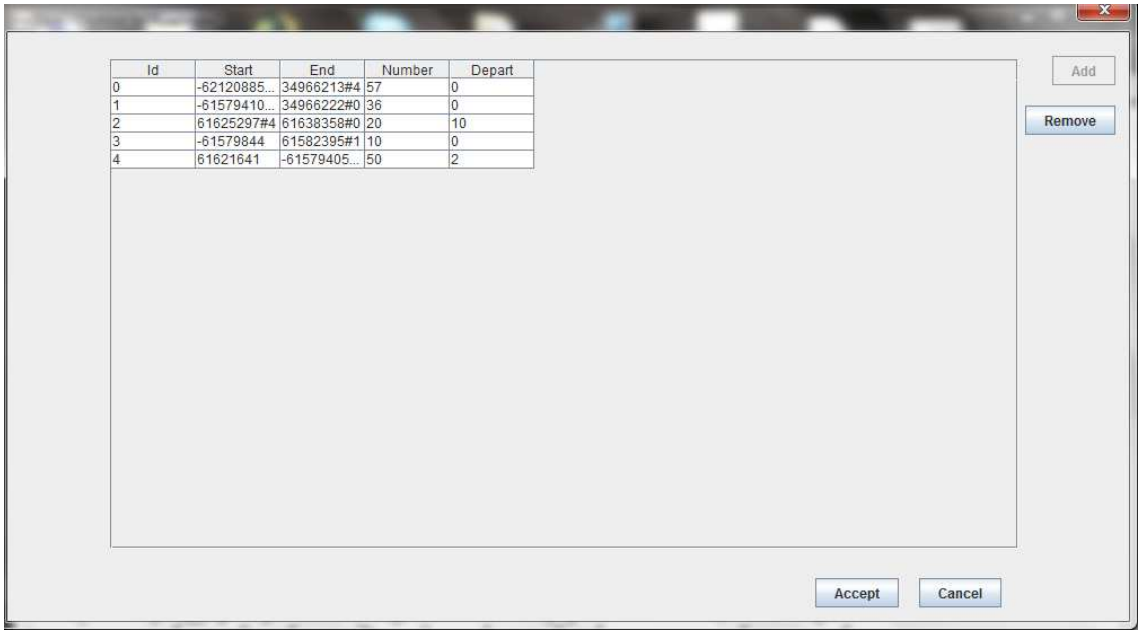

Figura 43 *Imagen de una vista de datos en Citymob for Roadmaps.*

6.6.17 Crear traza

Esta opción crea una traza con el movimiento de los vehículos. La traza es generada en el mismo directorio que el usuario especificó al crear el proyecto. La traza se puede encontrar en formato SUMO y en formato NS-2. Para crear la traza hay que seguir os siguiente pasos:

1. Clic en el botón **Crear** Traza o menú **Simulate** opción **Create trace**.
2. Especificar el tiempo de final de simulación.
3. Especificar el número de trazas a crear.

Dependiendo del tamaño del mapa y de la cantidad de vehículos a generar, la traza puede tardar horas. En cualquier caso es posible cancelar a creación de la traza en cualquier momento.

Figura 44 *Diálogo de opciones de traza.*

Dado que los investigadores realizan múltiples simulaciones para validar sus trabajos, se ha añadido la opción de crear varias trazas distintas con la misma configuración. Esta opción facilita este trabajo evitando que el usuario tenga que seleccionar la opción de generar traza tantas veces como trazas necesite.

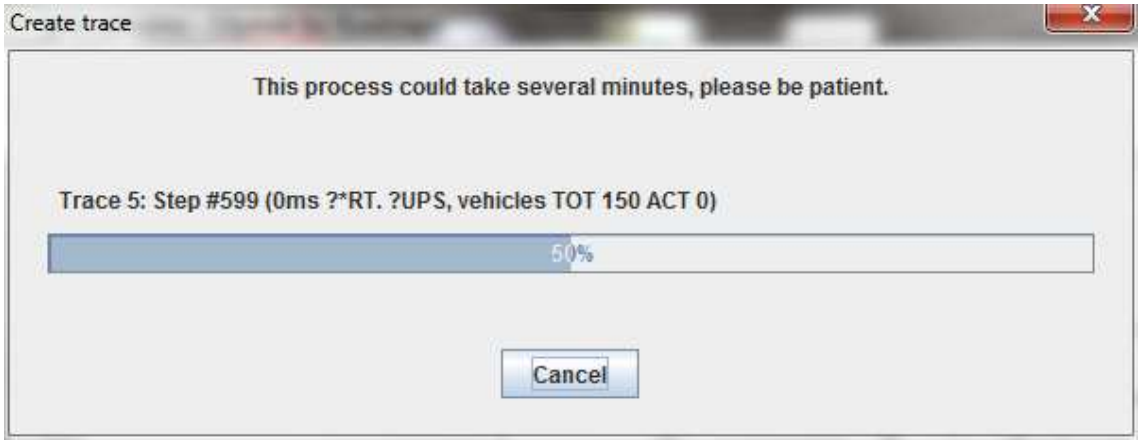

Figura 45 *Diálogo que aparece al crear una traza en Citymob for Roadmaps.*

6.6.18 Visualizar traza

1. Hacer clic en el botón **Visualizar traza** o en el menú **Simulate** opción **Visualize trace**. Entonces se abrirá una aplicación de SUMO con el mapa cargado y con los vehículos en movimiento.

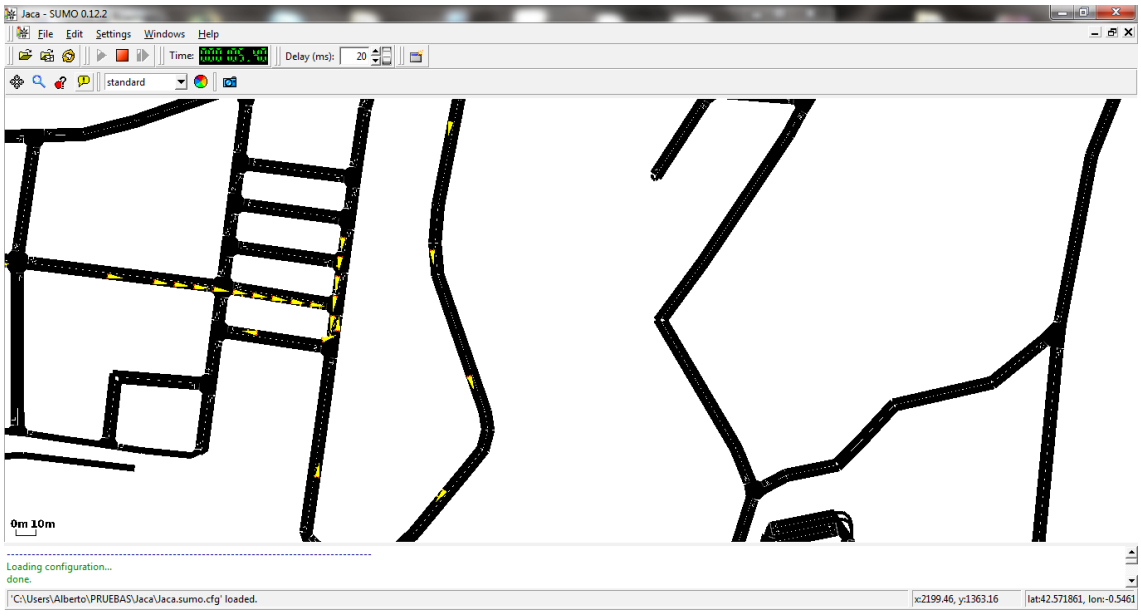

Figura 46 *Pantalla de visualización mediante SUMO.*

6.6.19 Asistente

Además de todas estas funcionalidades, Citymob for Roadmaps tiene la posibilidad de seguir un asistente que ayuda a crear modelos de movilidad en poco tiempo y que son prácticamente igual de funcionales que los creados sin asistente. La idea es que un usuario pueda seguir el asistente sin conocer absolutamente nada de la aplicación. Aún así, se van a describir los pasos a seguir en el manual de usuario.

1. Hacer clic en el botón **Abrir asistente** o seleccionar menú **About** opción **Wizard**. Aparecerá la pantalla de bienvenida.

2. Hay que pulsar **Next** para continuar”

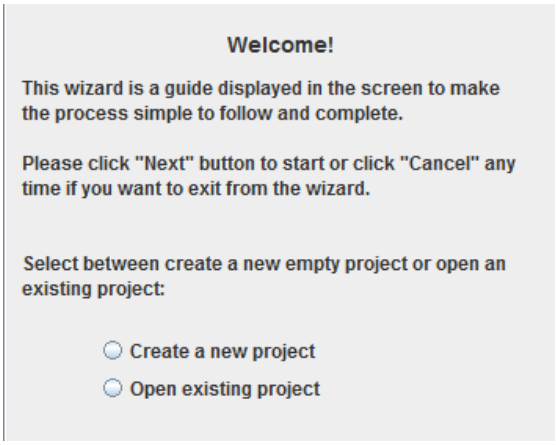

Figura 47 *Pantalla del asistente preguntando sobre un nuevo proyecto.*

3. Elegir entre crear un proyecto nuevo o abrir uno existente. Después de seleccionar el proyecto hay que pulsar **Next**.

    3.1 Si se elige nuevo proyecto, hay que seguir el proceso descrito en 6.6.2 y elegir entre descargar un mapa nuevo o importarlo.

        3.1.1 En caso de elegir descargar se podrá abrir el diálogo de descargar mapa y siguiendo el proceso descrito en 6.6.8 podrá descargarse el mapa.
        3.1.2 En caso de elegir importar mapa podrá hacerse eligiendo la ruta en la que esta ubicado el mapa.

    3.2 Si se elige abrir uno existente, hay que seguir el proceso descrito en 6.6.3.

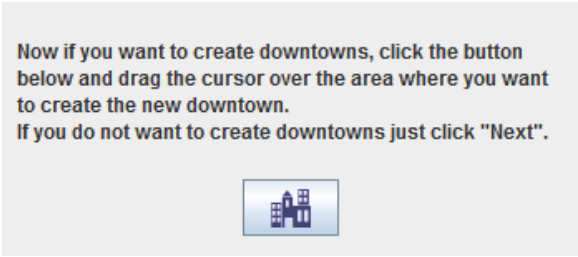

Figura 48 *Pantalla del asistente que permite crear downtowns.*

4. Crear un centro de atracción tal como se explica en el apartado 6.6.14.

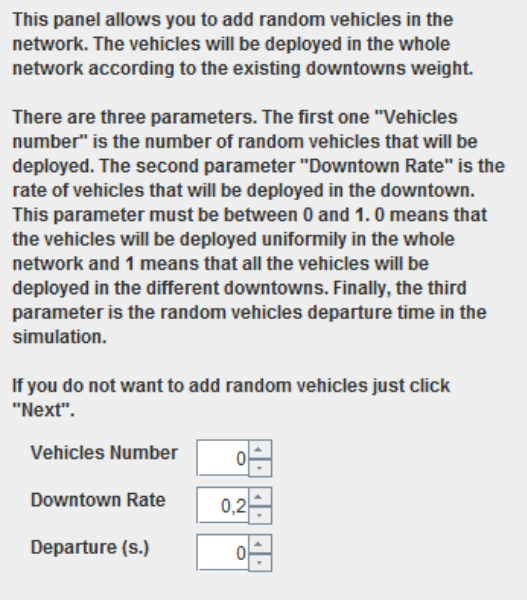

Figura 49 *Pantalla del asistente para crear vehículos aleatorios.*

5. Añadirse vehículos aleatoriamente del mismo modo que en el apartado 6.5.9.

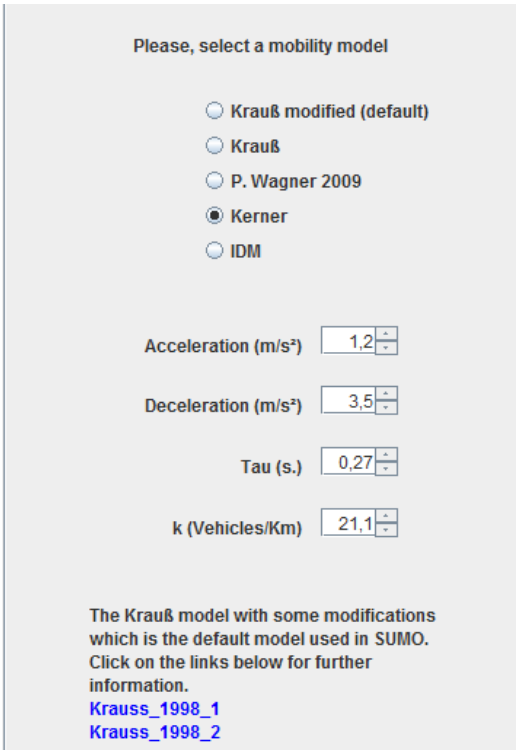

Figura 50 *Pantalla del asistente para insertar modelos.*

6. Definir un modelos de movilidad igual que en el apartado 6.6.15.

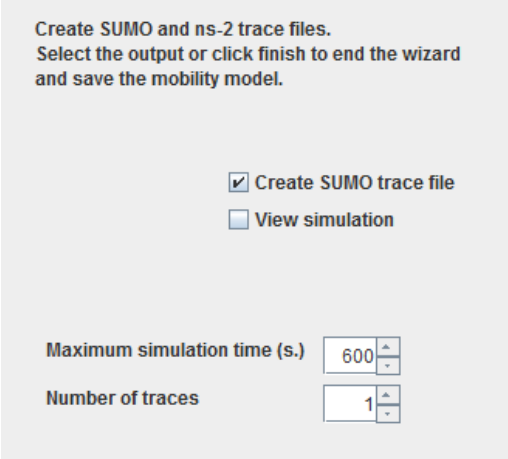

Figura 51 *Pantalla del asistente preguntando sobre el formato de salida.*

7. Seleccionar la forma de salida. Se podrá elegir crear un archivo de traza y/o ver la simulación mediante un programa externo. Una vez elegido se hace clic en **Finish** y ya está creado el modelo para ser usado.

6.6.20 About

La opción about muestra información sobre la aplicación. Para abrir esta ventana de información hay que hacer lo siguiente:

1. Clic en el menú **Help** opción **About**.

Figura 52 *Ventana de about de Citymob for roadmaps*

Desde esta ventana se pueden obtener los siguientes datos:

- Versión de la aplicación
- Disclaimer sobre la licencia
- Entidades que han colaborado en su desarrollo
- Personas que han formado parte del desarrollo

También es posible acceder a las páginas web de las distintas entidades con tan solo hacer clic sobre el logotipo.

6.7 Resolución de problemas

6.7.1 Problemas en la configuración

Problema

Al iniciar la aplicación aparece un mensaje de error con el mensaje "Cannot find configuration file c4r.cfg Please check if the file exists".

O bien

Al iniciar la aplicación aparece un mensaje de error con el mensaje "The configuration file c4r.cfg could not be correct". Please go to menu "Tools" option "options" to configure correctly".

Solución

El archivo de configuración de Citymob for Roadmaps probablemente no habrá sido creado o estará corrupto. Para solucionar el problema hay que seguir los siguientes pasos:

1. Clic en menú **Tools**, opción **Options.**
2. En caso de disponer de un sistema operativo Windows hay que comprobar que en la caja de texto de **SUMO installation path** aparece la ruta de instalación de SUMO. De no ser así hay que hacer clic en el botón con un icono de carpeta y buscar la ubicación de SUMO.
3. Comprobar que en la caja de texto de **OpenStreetMap API** hay un texto como este: "http://api.openstreetmap.org/api/0.6/map?bbox=". Esta es la API usada actualmente. En caso de que cambie la API, buscar la nueva buscando en http://wiki.openstreetmap.org/wiki/API.
4. Pulsar **Accept.**

Problema

Cuando se intenta descargar un mapa o cuando se intenta crear una traza aparece un mensaje de error con el texto "Cannot run program "(Una ruta)": CreateProcess error=2, El sistema no puede encontrar el archivo especificado".

O bien.

Cuando se intenta visualizar una traza aparece un mensaje de error con el texto "An error ocurred while attemping to read from the file (Un nombre de archivo)".

Solución

Estas incidencias tienen el mismo origen. Lo primero es comprobar la versión de SUMO instalada. Si la versión es distinta de la 0.11.1 no hay garantía de que la aplicación funcione como es debido. Por lo tanto la solución es trabajar con la versión 0.11.1 Si la versión instalada es la correcta entonces hay que revisar el archivo de configuración. Para ello hay que seguir los siguientes pasos:

1. Clic en menú **Tools**, opción **Options**.
2. En caso de disponer de un sistema operativo Windows hay que comprobar que en la caja de texto de **SUMO installation path** aparece la ruta de instalación de SUMO. De no ser así hay que hacer clic en el botón con un icono de carpeta y buscar la ubicación de SUMO.
3. Pulsar **Accept**.

Problema

Al darle al botón **Load map** en la ventana de descarga de mapas devuelve un error con un texto que parece una dirección de Internet.

Solución

1. Clic en menú **Tools**, opción **Options**.
2. Comprobar que en la caja de texto de **OpenStreetMap API** hay un texto como este: "http://api.openstreetmap.org/api/0.6/map?bbox=". Esta es la API usada actualmente. En caso de que cambie la API, buscar la nueva buscando en http://wiki.openstreetmap.org/wiki/API.
3. Pulsar **Accept**.

6.7.2 Problemas con los mapas

Problema

Cuando se abre la ventana de descarga de mapa, aparece un mosaico de aspas rojas en lugar del mapamundi.

Solución

Lo que esta ocurriendo es que el equipo no tiene acceso a Internet. La solución es arreglar los problemas de conectividad.

Problema

Al abrir la ventana de descarga de mapas o durante la descarga aparece un mensaje de error con el texto “Connection refused: connect”.

Solución

Este error significa que, lamentablemente, los servidores de OpenStreetMap no están operativos. Puede ir a la página www.openstreetmap.org en busca de más información sobre el problema. Habitualmente esto suele suceder de cuando los servidores están cerrados por mantenimiento.

6.7.3 Otros problemas

Problema

Aparece el mensaje “Imposible convert the map to SUMO format. Further information in <ruta>”.

Solución

Este error aparece cuando no se ejecuta satisfactoriamente algún proceso de SUMO. Lo primero es comprobar que se está trabajando con la versión 0.11.1. Es posible que con otras versiones la aplicación no sea plenamente compatible. Una vez comprobado esto es importante comprobar que no haya espacios en la ruta donde se almacena el archivo. De ser así hay que cambiar el proyecto a una ruta que no contenga espacios.

7. Conclusiones

Como se ha visto a lo largo del documento existen diferentes modelos de movilidad para redes de vehículos que permiten simular el tráfico rodado de una ciudad. Estos modelos son bastante realistas, pero dada la complejidad del tráfico de los vehículos y todos los factores que afectan al mismo, estos modelos sintéticos no reflejan la realidad completamente. Será labor de futuros trabajos identificar las variables que permitan realizar de una forma más realista las simulaciones. De este modo se podrían obtener resultados más fiables, mejorando la calidad de las investigaciones relacionadas con VANET's.

A lo largo del documento también se comparan distintos generadores de movilidad. Estas aplicaciones tienen muchos puntos en común. En general son aplicaciones software libre gratuitas con la posibilidad de trabajar con varios tipos de mapas, con diversos modelos de movilidad y tráfico para elegir, pero tienen la limitación de que no soportan varios tipos de formatos y las trazas rara vez son exportables para otro tipo de aplicaciones.

Teniendo en cuenta las características que presentan los modelos de movilidad y los generadores, se han creado la especificaciones que darán paso a la aplicación Citymob for Roadmaps. Esta aplicación ha nacido con la vocación de cubrir algunas necesidades que aún no estaban resueltas por los generadores de movilidad. Una mejora que introduce Citymob for Roadmaps es la posibilidad de introducir datos interactuando directamente con el mapa que aparece en pantalla. Además estos mapas pueden corresponder a uno real, gracias al sistema que permite descargar mapas del proyecto OpenStreeMap. También se ha conseguido crear downtowns de forma que SUMO pueda interpretarlos y trabajar con ellos sin necesidad de modificar en absoluto SUMO. Otro de los hitos relevantes de Citymob for Roadmaps es la posibilidad de exportar trazas de movilidad de SUMO a formato NS-2. A día de hoy es posible hacer esta transformación usando distintas herramientas pero lamentablemente ninguna lo hacía del modo en el que era requerido, por lo que se ha decidido desarrollar un sistema para llevar a cabo esta tarea.

El trabajo futuro para esta aplicación puede centrarse en aumentar el número de opciones para añadir a un modelo movilidad. SUMO permite bastantes opciones que aún no han sido explotadas en Citymob for Roadmaps. Otra modificación que podría ser atractiva es añadir la posibilidad de trabajar con NS-2. La versión actual de C4R exporta una traza de formato SUMO a NS-2 pero sería interesante que el programa trabajase con NS-2 para poder simular VANETs desde una sola aplicación. Otra posible línea de trabajo podría ser añadir una opción que ofrezca estadísticas y gráficas sobre el movimiento de los vehículos en un intento de automatizar la tarea de investigación. En cualquier caso, lo que se busca como trabajo futuro es mejorar el sistema para conseguir una simulación con un movimiento de vehículos lo más realista posible.

Bibliografía

[CANU] VanetMobiSim. Disponible en: http://vanet.eurecom.fr/

[EJT06] S. Erlingsson, A. M. Jonsdottir, T. Thorsteinsson, "Traffic Stream Modelling of Road Facilities", Transport Research Arena, Göteborg, Suecia, Junio, 2006.

[EURECOM] Centro de investigación, www.eurecom.fr/

[FH08] M. Fiore, J. Härri, "The Networking Shape of Vehicular Mobility", MobiHoc'08, Hong Kong SAR, China, Mayo, 2008.

[GH00] D. Garmus, D. Herron, "Function Point Analysis: Measurement Practices for Successful Software Projects". Ed. Addison-Wesley, Diciembre, 2000.

[GPL3] Licencia de software libre. Disponible en www.gnu.org/licenses/gpl.html

[HFB06] J. Härri, F. Filali, C. Bonnet, "Mobility Models for Ad Hoc Networks: A Survey and Taxonomy", Marzo, 2006.

[HFB09] J. Härry, F. Filali, C. Bonnet, "Mobility Models for Ad Hoc Networks: A Survey and Taxonomy", 2009.

[HFFB] J. Härri, M. Fiore, F. Filali, C. Bonnet, "DEMO: Simulating Realistic Mobility Patterns for Vehicular Networks with VanetMobiSim", disponible en http://vanet.eurecom.fr/

[I830] IEEE. "IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications", IEEE Computer Society, 1998.

[JD08] D. Jiang, L. Delgrossi, "IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Enviroments", Vehicular Technology Conference, 2008, VTC Spring 2008.

[K98] B. S. Kerner. "Experimental Features of Self-Organization in Traffic Flow". Physical Review Letters, 81,3797-3400. 1998.

[KTH08] A. Kesting, M. Treiber, D. Helpbing, "Agents for Traffic Simulator", Abril, 2008.

[KW07] S. Krauß, P. Wagner, C. Gawron. "Metastable States in a Microscopic Model of Traffic Flow". Physical Review E. volume 55, number 304, 55-97. 1997.

[LA05] T. D. C. Little, A. Agarwal, "An information Propagation Scheme for VANETs", MCL Technical Report No. 07-01-2005, Boston University, Boston MA, Julio, 2005.

[MOVE] MOVE (MObility model generator for VEhicular networks). Disponible en http://lens1.csie.ncku.edu.tw/MOVE/index.htm

[MK04] P. Mohapatra, S. V. Krishnamurthy, “Ad hoc Networks Technologies and Protocols”, Springer, 2004.

[MTCCM09] Francisco J. Martinez, C.K. Toh, Juan-Carlos Cano, Carlos T. Calafate, Pietro Manzoni, "A Survey & Comparative Study of Simulators for Vehicular Ad Hoc Networks (VANETs)", Wireless Communications and Mobile Computing journal; 2009.

[MTB07] G. De Marco, M. Tadauchi, L. Barolli, “CAVENET: Description and Analysis of a Toolbox for Vehicular Networks Simulation”, IEEE Xplore, 2007, Disponible en: http://ieeexplore.ieee.org/

[OSM] Proyecto dedicado a crear y ofrecer datos geográficos. Disponible en www.openstreetmap.org

[SE07] S. Eichler, “Performance Evaluation of the IEEE 802.11p WAVE Communication Standard”, Proceedings of the 1st IEEE International Symposium on Wireless Vehicular Communications (WiVeC), Baltimore, USA, Septiembre , 2007.

[STRAW] STRAW (Street Random Waypoint). Disponible en: http://www.aqualab.cs.northwestern.edu/projects/STRAW/index.php

[SUMO] SUMO (Simulation of Urban MObility). Disponible en: http://sumo.sourceforge.net

[TH02] M. Treiber, D. Helbing, “Realistiche Mikrosimulation von Straßenverkehr mit einem einfachen Modell”, Arbeitsgemeinschaft Simulation (ASIM), Ronstock, Alemania, Septiembre 2002.

[TH99] M. Treiber, D. Helbing, “Explanation of Observed Features of Self-Organization in Traffic Flow”, Preprint cond-mat/9901239, 1999.

[TIGER] Topologically Integrated Geographic En coding and Referencing system. Es una base de datos de mapas de EE.UU. Disponible en: http://www.census.gov/geo/www/tiger/

[TRANS] Trans (Traffic and Network Simulator environment). Disponible e: http://trans.epfl.ch/

[W06] P. Wagner. “How human drivers control their vehicle”, The European physical journal B. 52., 427-431. 2006.