Departamento de Informática

PROYECTO FIN DE CARRERA

Diseño e implementación de VanSimFM, un asistente para la generación de escenarios vehiculares en NS-2

Autor: José Munera López

Tutor: José María de Fuentes García-Romero de Tejada

Co-tutor: Jorge Blasco Alís

Leganés, julio de 2011

Título: Diseño e implementación de VanSimFM, un asistente para la generación de escenarios vehiculares en NS-2

Autor: José Munera López

Director: José María de Fuentes García-Romero de Tejada

Co-Director: Jorge Blasco Alís

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

El capítulo más grande de esta memoria debería de ser para los agradecimientos puesto que es muchísima la gente a quien tengo que agradecer su apoyo. Aunque tenga que resumir este apartado y no estén todos los nombres aquí escritos, yo tengo todos muy claros.

Quiero agradecer a mis padres su apoyo incondicional, porque todo lo bueno que hay en mí se lo debo a ellos. A mis tíos por ayudarme siempre que lo he necesitado y transmitirme su alegría siempre que me ha hecho falta.

A Isa, por todo y por más.

Por orden alfabético a Ana, Andrés, Ángel, Beto, César, Gonzalo, Guille, José, Joseda, María José, Mixali, Maris, Miguel, Nacho, Natalia, Patricia, Pablo, Sergio y muchos amigos que ha ido conociendo en estos años tanto en BEST, como en clase y fuera de ella. Gracias por vuestra paciencia y vuestra ayuda.

Y por último a Chema, porque sin su infinita paciencia, ayuda y esfuerzo aún estaría enterándome de qué es eso de las VANET.

Gracias a todos los mencionados y los que me olvido de mencionar por aguantarme cuando menos aguantable estaba, por tener paciencia y ayudarme siempre que lo he pedido y aún cuando no lo he pedido.

Resumen

Las redes vehiculares ad-hoc (VANET) serán, con toda probabilidad, el gran salto de las TIC en la industria automovilística tan pronto como se disponga de modelos de comportamiento que justifiquen las inversiones necesarias para desarrollar los dispositivos hardware y las aplicaciones que den contenido a esas redes.

En la práctica, estos modelos de comportamiento sólo pueden obtenerse, a un coste asumible, a través de aplicaciones de simulación capaces de predecir el comportamiento de una aplicación o protocolo para VANET en diferentes escenarios.

Aunque se dispone de un gran número de simuladores, tanto de movilidad como de red, existen dificultades importantes para su uso de forma confortable:

- Su manejo no es sencillo, y varía bastante de una herramienta a otra.
- La interacción entre simuladores de movilidad y red también resulta compleja.
- Al trabajar cada investigador sobre parámetros diferentes de los demás, a veces resulta difícil o imposible comparar resultados de simulaciones del mismo modelo realizadas con criterios distintos.

Ante esta situación, este proyecto plantea el análisis, diseño y construcción de un sistema que:

- Integre diferentes simuladores disponibles.
- Automatice la interacción entre simuladores de movilidad y de red.
- Simplifique y haga homogénea la creación de experimentos de simulación.
- Permita la incorporación de nuevas funcionalidades y herramientas.

Adicionalmente, el proyecto pretende construir un catálogo de experimentos de simulación con los escenarios más representativos del entorno real de las VANET.

Palabras clave:

VANET, simulación, VanSimFM, experimentos de simulación

Abstract

Ad-hoc vehicular networks (VANET) will become, with a high probability, the big bet of the Information and Communication Technologies into the automotive industry, as soon as models of behavior are made available to justify the costs of developing new hardware devices and applications that build those nets and their contents.

In practice, those behavioral models can only be obtained, at a reasonable cost, by using simulation applications capable of predicting the behavior of a VANET on different scenarios.

Even though such tools already exist, for mobility and network simulations, there are some important difficulties for them to be used in a comfortable way:

- Use is not simple, and quite different from tool to tool.
- The interaction among mobility and network simulation tools is also fairly complex.
- As each person works on parameters different from the others, sometimes is difficult or even impossible to compare simulation results of a single model elaborated by two different set of criteria.

To face such situation, this project proposes the analysis, design and construction of a system that:

- Integrates different simulation tools available.
- Automates the interactions among mobility and network simulation tools.
- Simplify and homogenize the definition of simulation experiments.
- Allows the inclusion of new functionalities/tools.

Additionally, the project will attempt to create a catalog of simulation experiments that includes the most representative scenarios of the VANET's real environment.

Keywords:

VANET, simulation, VanSimFM, simulation experiments

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

En las últimas décadas hemos sido testigos de un gran número de avances tecnológicos en diversas áreas. Las tecnologías de la información se han convertido en uno de los ejes principales de estos avances y así, en menos de 20 años, hemos conseguido tener en un dispositivo móvil más capacidad de comunicación, almacenamiento y procesamiento que en un ordenador personal de hace 15 años, lo que ha hecho que nuestra vida diaria haya cambiado, en algunos casos drásticamente, para abrazar estos avances, permitiéndonos acceder a un gran volumen de información sobre dispositivos muy reducidos en cualquier lugar con cobertura de red móvil.

Sin embargo en el sector automovilístico, a pesar de estar en constante evolución, estos avances en las tecnologías de la información y especialmente en las telecomunicaciones no han llegado con tanta fuerza como, por ejemplo, a los salones de los hogares. Si bien la tecnología se abre paso cada vez más dentro de los vehículos, con GPS u ordenadores de a bordo, aún existe una importante diferencia con los avances tecnológicos que se ven, por ejemplo, en nuestros hogares. A día de hoy la inmensa mayoría de los vehículos que se venden, incluidos los de gama alta, ofrecen como única vía de comunicación con el exterior un teléfono móvil que sólo se puede usar para realizar llamadas o en su defecto una conexión al teléfono móvil del usuario a través de Bluetooth. Esto deja a los vehículos como entes aislados en la carretera desde los que, como mucho, se pueden realizar llamadas telefónicas.

La tecnología llamada a eliminar este “desfase” tecnológico en las comunicaciones en el sector del automóvil son las redes vehiculares ad-hoc o VANET por sus siglas en inglés. Las VANET son un subtipo de redes móviles ad-hoc, o MANET por sus siglas en inglés, en las que los nodos que participan en la comunicación son vehículos equipados con un hardware específico para la participación en comunicaciones sobre VANET llamado *On-Board Unit* (OBU). Aparte de los vehículos, normalmente se asume que en una VANET existirán una serie de nodos estáticos llamados *Road Side Unit* (RSU) que se encuentran ubicados junto a las calles y carreteras y unen las VANET con una infraestructura fija, por ejemplo una red perteneciente a la Dirección General de Tráfico. La alta velocidad a la que se mueven los nodos en una VANET sumada al corto alcance de las OBU hacen que las topologías de este tipo de redes cambie constantemente lo que dificulta el diseño de servicios o hardware para operar sobre VANET.

El objetivo de los primeros sistemas de comunicación vehicular era mejorar la seguridad en la carretera si bien con el paso del tiempo se han ido añadiendo propuestas al catálogo de protocolos y aplicaciones, relacionadas con diversos temas como las comunicaciones de voz a través de IP, la comunicación de incidencias en el tráfico para evitar retenciones de tráfico, aplicaciones comerciales, etc. [1]

En los últimos años las VANET han despertado el interés de la comunidad investigadora debido a que se trata de un nuevo campo en el que aún queda mucho por definir, lo que conlleva nuevos retos y oportunidades que atraen a investigadores especializados en distintas áreas que van desde la definición de protocolos e infraestructuras viarias hasta el desarrollo de nuevas aplicaciones.

Por la propia naturaleza de las VANET se hace muy difícil validar sobre un entorno real nuevas propuestas ya que se requeriría en algunos casos una gran cantidad de vehículos dotados de sistemas de comunicación circulando constantemente en distintas calles o carreteras. Es por esto por lo que la simulación tiene un papel clave en la validación de nuevas propues-

tas en VANET. La simulación de VANET tiene que tener en cuenta dos factores muy importantes:

- la movilidad de los vehículos
- la comunicación entre estos.

Esta doble componente en la simulación de VANET no solo hace que sea una tarea más compleja, sino que también contribuye a la formación de un criterio muy heterogéneo a la hora de simular: podemos encontrar propuestas validadas con el mismo simulador de red pero con distinto origen en la movilidad de los vehículos o viceversa, propuestas validadas con un simulador de VANET, usando sólo un simulador de red y generando manualmente los movimientos, etc. A estas alternativas a la hora de elegir la aplicación para realizar la simulación hay que añadir los distintos parámetros utilizados para configurar estas simulaciones: trazado sobre el que se realizará la simulación, número de coches, trayectos de los distintos coches, número y posición de RSU, tiempo de simulación, etc. Como podemos ver en la práctica hay infinitas combinaciones para simular una propuesta VANET.

La elaboración y ejecución de un experimento de simulación para VANET puede descomponerse en seis pasos básicos: Definición del mapa sobre el que se ejecutará la simulación de movilidad, configuración de la simulación de movilidad, ejecución de la simulación de movilidad, configuración de la simulación de red, ejecución de la simulación de red y finalmente el procesamiento de los datos obtenidos de la simulación de red. La Figura 1 muestra estos pasos ordenados secuencialmente.

Figura 1 - Pasos a seguir para realizar un experimento de simulación VANET

1.2 Motivación

Como se ha indicado en la sección anterior, existe un gran número de posibilidades diferentes a la hora de simular VANET, este amplio abanico de posibilidades unido a la falta de un criterio común a la hora de configurar y realizar las simulaciones está teniendo como resultado que las propuestas que se presentan en la comunidad investigadora tengan, en un gran número de casos, configuraciones de simulación tan dispares que dificultan e incluso hacen imposible una valoración de las mismas o una comparación contra otras propuestas similares.

Dada la gran importancia que tiene a día de hoy la simulación en la validación de aplicaciones y protocolos para VANET el hecho de no disponer de un criterio común de simulación puede llevar a importantes problemas en la comparación y validación de protocolos y aplicaciones para VANET lo que, sumado a la importancia de algunas de estas aplicaciones y protocolos, hace que esta situación se convierta en un problema lo suficientemente importante como para recibir la atención de la comunidad investigadora.

Como se puede ver en la Figura 1, la generación de experimentos de simulación para VANET requiere un primer paso que supone la configuración y, según la herramienta, ejecución de una simulación de movilidad. La configuración y ejecución de este tipo de simulaciones es compleja y requiere un mínimo grado de familiarización con la herramienta que se va a usar. Es posible que este hecho sea el que haga que los autores de nuevas aplicaciones y protocolos para VANET recurran a la solución que les sea más fácil para resolver este primer paso. En algunos casos será el uso de una herramienta específica que ya se conozca, mientras que en otros casos la solución pasará por definir un modelo de movilidad muy simple al precio de perder fidelidad y calidad en la simulación de movilidad.

Se percibe por tanto la necesidad de una herramienta que simplifique al máximo la generación de experimentos de simulación para VANET, permitiendo así que los creadores de nuevas aplicaciones y protocolos generen la movilidad para sus simulaciones sin tener un conocimiento específico de la herramienta de simulación de movilidad, pero que a la vez permita a usuarios ya familiarizados con la herramienta generar simulaciones de movilidad más elaboradas. Para esto la herramienta podrá utilizar uno o varios simuladores de movilidad a través de una aplicación que simplifique su uso, y un simulador de red único, que debe contar con el respaldo de la comunidad investigadora y ser usado por un número representativo de investigadores en sus simulaciones.

De esta forma se podrá contar con una calidad mínima en la generación de movilidades para la simulación de aplicaciones y protocolos para VANET además de tener unos resultados comparables al ejecutar las simulaciones con un mismo simulador de red y un conjunto delimitado, común y conocido de simuladores de movilidad.

1.3 Objetivos

El principal objetivo perseguido con el desarrollo de este proyecto es el análisis, diseño e implementación de una herramienta que simplifique la creación de experimentos de simulación para VANET. Adicionalmente se busca que los experimentos que genere dicha herramienta puedan contar con diversas fuentes para la generación de movilidad y que además permita la ejecución de los experimentos de simulación que se generen con ella.

Por tanto, la funcionalidad básica que se busca cubrir con el sistema resultante de este proyecto abarca las cuatro primeras etapas vistas en la Figura 1. Adicionalmente, como ya se ha mencionado, se buscará que el sistema pueda realizar las dos últimas etapas del proceso de simulación.

Se ha fijado como objetivo adicional la creación de un catálogo de experimentos de simulación para su uso directo en la validación de propuestas VANET. Este catálogo deberá reunir las configuraciones más comunes empleadas en la simulación de VANET y permitir su modificación para su uso con diferentes propuestas.

Por lo tanto, y de acuerdo a lo expuesto en esta sección los objetivos de este proyecto quedan definidos de la siguiente forma:

- Análisis, diseño e implementación de una herramienta que simplifique la generación de experimentos de simulación de VANET.
- Creación de un catálogo de experimentos de simulación que contenga las configuraciones de red y escenarios de movilidad más representativos de entre los que podemos encontrar en un entorno real.

1.4 Organización del presente documento

Con el objetivo de facilitar la lectura del documento en esta sección presenta una breve descripción del contenido de cada una de las partes que forman este documento, compuesto por seis capítulos y tres anexos:

Capítulo 1, Introducción y objetivos: En este capítulo se ofrece una introducción a las VANET junto con las motivaciones de este proyecto fin de carrera y los objetivos que persigue.

Capítulo 2, Estado del arte: en esta capítulo se muestra un análisis de las principales alternativas para realizar simulaciones de VANET disponibles hoy en día junto con las prácticas de simulación más comunes entre la comunidad investigadora obtenidas a partir del estudio de más de 30 publicaciones.

Capítulo 3, Análisis: En este capítulo se expone una perspectiva general del sistema junto con el estudio de la arquitectura que tendrá el sistema, definiéndose los componentes que la integrarán. El capítulo también incluye un estudio de las tecnologías a utilizar en el desarrollo del sistema junto con la especificación de los casos de uso, los requisitos de software y las pruebas de aceptación definidas para comprobar el cumplimiento de estos requisitos.

Capítulo 4, Diseño detallado: En este capítulo se detallan los componentes definidos en el capítulo de análisis mostrando sus diagramas de clases. Adicionalmente se mostrarán un conjunto de diagramas de secuencia que permitan comprender las distintas interacciones que tendrán lugar entre los distintos componentes y clases del sistema.

Capítulo 5, Implementación y pruebas: En este capítulo se presentan los detalles más destacables de la implementación del sistema junto con los resultados de las pruebas de aceptación llevadas a cabo.

Capítulo 6, Uso de VanSimFM para la generación de experimentos: En este capítulo se incluyen los experimentos de simulación creados con VanSimFM, sus características y las decisiones de diseño tomadas para crearlos.

Capítulo 7, Conclusiones y líneas futuras: Este último capítulo expone las conclusiones obtenidas con el desarrollo del proceso en distintos ámbitos así como alguna de las líneas de desarrollo futuras a seguir para extender la funcionalidad del sistema o mejorar su funcionamiento.

Anexo 1, Gestión del proyecto: En este anexo se detalla la planificación del proyecto junto con el seguimiento del mismo. Adicionalmente se muestra el presupuesto del proyecto junto con las desviaciones detectadas en el mismo.

Anexo 2, Manual de usuario: En este anexo se muestra el manual de usuario en el que se detalla el funcionamiento del sistema paso a paso.

Anexo 3, Plantillas: En este anexo se muestran las plantillas utilizadas para recoger distintos datos a lo largo de la memoria.

Capítulo 2

Estado del arte

2.1 Panorámica de herramientas existentes

En esta sección se ofrece una visión general de las principales soluciones disponibles hoy en día para realizar simulaciones de VANET.

Actualmente existe una gran variedad de posibilidades a la hora de realizar simulaciones de propuestas para VANET. Podemos clasificar estas opciones en tres tipos de acuerdo a sus funcionalidades: existen simuladores de movilidad, que se centran en la generación de trazas de movilidad; simuladores de red, que se centran en el intercambio de datos entre nodos de red; y por último simuladores integrados que ofrecen las funcionalidades de los dos tipos vistos anteriormente. La integración entre las simulaciones de movilidad y de red se puede realizar con una única aplicación que realiza las dos simulaciones o, como hacen las soluciones federadas, incorporando un simulador de red y otro de movilidad. La Figura 2 clasifica las principales herramientas de simulación de acuerdo al tipo de simulador al que pertenecen y los pasos que cubren del proceso de simulación de aplicaciones y protocolos para VANET.

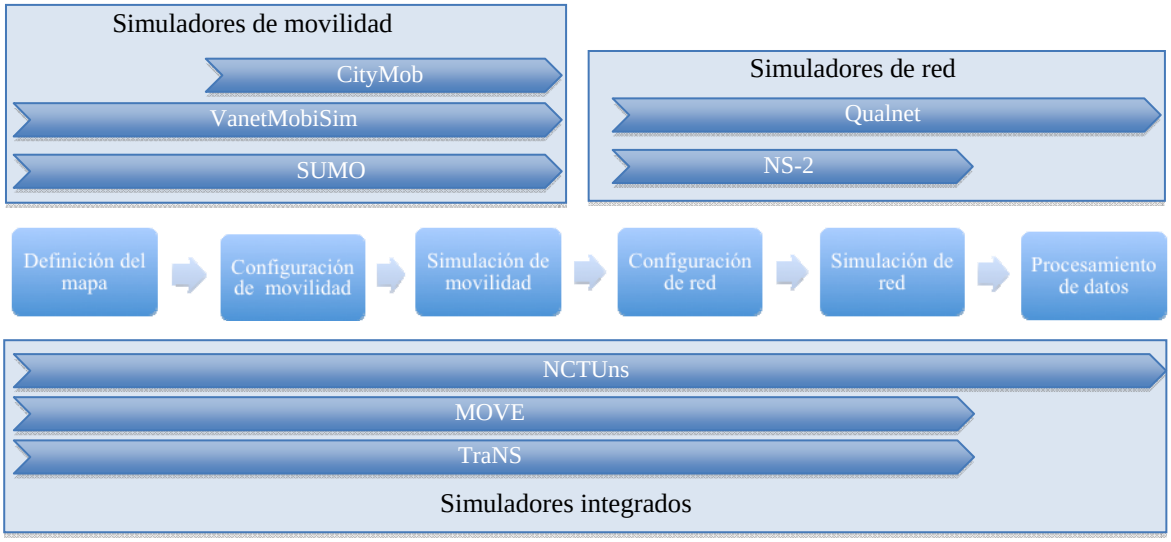

Figura 2 - Clasificación y alcance de los principales simuladores

Como se puede ver en la Figura 2, SUMO [2], CityMob [3] y VanetMobiSim [4] entran en la categoría de simuladores de movilidad. SUMO y VanetMobiSim generan trazas de movilidad mediante simulaciones de movilidad sobre mapas especificados por el usuario y de acuerdo a una serie de parámetros especificados. A diferencia de VanetMobiSim, SUMO cuenta con la opción de poder ejecutarse con interfaz gráfica. Por otra parte CityMob sólo utiliza mapas de tipo rejilla, siendo su principal objetivo la generación de trazas de movilidad y no tanto la simulación.

Todas ellas son herramientas de código abierto y libre distribución capaces de generar trazas de movilidad que pueden ser utilizadas para la generación de experimentos de simulación del simulador de red NS-2.

Respecto a los simuladores de red, las principales opciones disponibles utilizadas en la comunidad investigadora son NS-2 [5] y Qualnet [6]. Ambos cuentan con funcionalidades

para simular cualquier tipo de red de comunicación, incluidas VANET. Mientras que Qualnet proporciona herramientas para el procesado posterior de los datos resultantes de la simulación, NS-2 no cuenta con esta funcionalidad.

Finalmente, los simuladores integrados con más aceptación en la comunidad investigadora son: NCTUns [7], MOVE [8] y TraNS [9]. NCTUns es, desde su última versión, una herramienta comercial que realiza las simulaciones de movilidad y de red dentro de la propia aplicación, adicionalmente ofrece la posibilidad de procesar los datos resultantes de la simulación. La configuración de las simulaciones de movilidad se ha de hacer manualmente y sobre mapas creados manualmente o importados a partir del formato TIGER [10]. Por otro lado TraNS y MOVE son soluciones federadas que utilizan SUMO para la simulación de movilidad y NS-2 para la simulación de red. La configuración de la simulación de movilidad en ambas soluciones está marcada por el uso de SUMO como simulador de movilidad y si bien ofrece una gran gama de opciones para la configuración de la simulación resulta muy compleja para un usuario no familiarizado con la aplicación. La principal diferencia entre MOVE y TraNS es que este último permite la interacción entre el simulador de red y el de movilidad durante la ejecución de la simulación.

Un ejemplo para ilustrar la complejidad de creación de simulaciones con MOVE podría ser la definición de un escenario de movilidad simple. La Figura 3 muestra la pantalla para la definición de la movilidad de una simulación con MOVE.

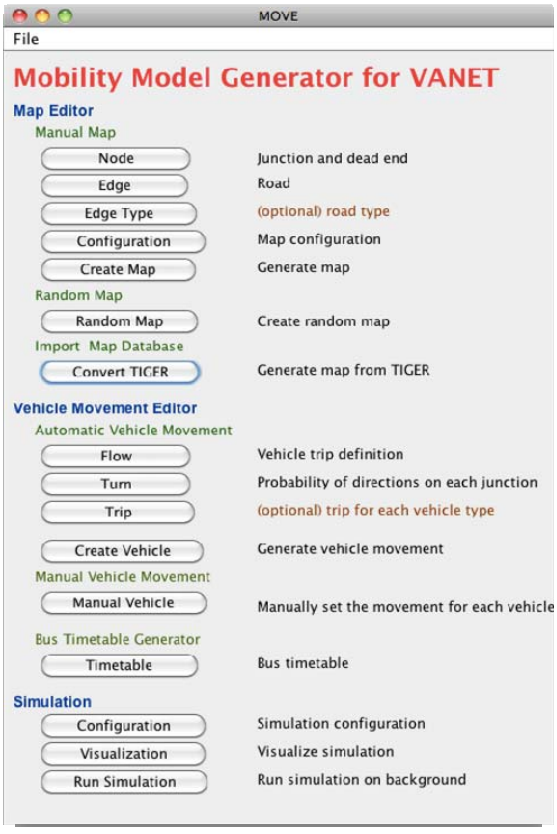

Figura 3 - Pantalla de simulación de movilidad de MOVE

Como se puede apreciar en la Figura 3, MOVE ofrece gran cantidad de opciones para la simulación de movilidad pero este gran número de opciones que pueden ser de gran utilidad para un usuario experimentado con la aplicación se convierte en un problema para los usuarios que no están familiarizados con MOVE. Opciones como la creación de mapas aleatorios

automáticamente o importar mapas de TIGER hacen el uso más simple pero a la hora de definir la movilidad en la simulación no existe ninguna opción que no pase por tener una movilidad ya definida o por crear una de forma totalmente manual, echándose de menos una opción que genere rutas de forma aleatoria de acuerdo a un parámetro o un asistente para la generación de rutas de forma manual más intuitivo y que no requiera conocer los identificadores de los nodos que intervienen en la simulación. Esto convierte a MOVE en una buena aplicación con muchas y muy interesantes opciones pero estas mismas opciones convierten a la aplicación en casi inaccesible para usuarios no experimentados.

Los lectores interesados pueden consultar [11, 12, 13] para consultar más detalles sobre las distintas opciones de simulación disponibles.

2.2 Prácticas de simulación actuales

En esta sección se muestran los resultados del estudio realizado sobre las prácticas de simulación más comunes hoy por hoy, así como los valores más utilizados para los distintos parámetros que conforman los experimentos de simulación.

La elección de la configuración de simulación tiene un impacto directo en sus resultados y, por tanto, en la validación de distintas propuestas. De esta forma un protocolo de intercambio de datos pesado y con un gran consumo de ancho de banda puede obtener unos resultados positivos si se valida en un escenario rural con pocos vehículos pero sin embargo resultar inviable en un escenario urbano con una gran densidad de vehículos.

Se ha llevado a cabo un estudio de más de 30 publicaciones realizadas entre los años 2.008 y 2.011 para evaluar las prácticas de simulación que se vienen realizando en la comunidad investigadora en los últimos años. La Tabla 1 muestra los distintos parámetros medidos en las 32 publicaciones analizadas como son los simuladores de red y movilidad utilizados, el tiempo total de simulación, el número de vehículos, el tipo de escenario y el tamaño del escenario; ordenados según el simulador de movilidad y de red utilizados en cada publicación.

	Tipo de herramienta de simulación		**Parámetros de simulación**			
	Movilidad	Red	Tiempo (seg)	Nº nodos	Escenarios	Tamaño
[14]	VanetMobiSim	Qualnet	250	100-350	Carretera	2,5x2,5km
[15]	VanetMobiSim	No especificado	200	12, 144	Aleatorio	1x1km
[16]	VanetMobiSim	No especificado	No especificado	50-250	Autopista	4km
[17]	TraNS (SUMO)	NS-2	100	576	Urbano	13,4x12,3km
[18]	TraNS (SUMO)	NS-2	100	576	Urbano	13,4x12,3km
[19]	TraNS (SUMO)	NS-2	200	916	Urbano	No especificado
[20]	TraNS (SUMO)	NS-2	78	20-80	Urbano	No especificado
[21]	SUMO	NS-2	300	150,250,350	Urbano	1,5x1,5km
[22]	STRAW (Urbano), Flee-Net (Autopista)	Jist/SWANS	60	10 urbano, 100 autopista	Urbano, Autopista	3x3km, 12km
[23]	Ninguno (Mov. Aleatorio)	NS-2	600	50x9	No especificado	5x5km
[24]	Ninguno (Mov. Aleatorio)	MATLAB	600	35	Aleatorio	1x1km
[25]	Ninguno (Mov. Aleatorio)	Qualnet	900	2-100	Aleatorio	3x3km
[26]	R	R	No especificado	25, 100	No especificado	No especificado
[27]	Trazas pre-generadas	NS-2	No especificado	Variable	Urbano	2,4x2,4km
[28]	No especificado	Qualnet	60	15-25	Carretera	1x1km
[29]	No especificado	NS-2	No especificado	50	Urbano	5x5km
[30]	No especificado	NS-2	600	75	Autopista	No especificado
[31]	No especificado	NS-2	No especificado	No especificado	Autopista	No especificado
[32]	No especificado	NS-2	Variable	Variable	Carretera	No especificado
[33]	No especificado	No especificado	No especificado	No especificado	No especificado	No especificado
[34]	No especificado	No especificado	No especificado	Variable	Carretera	30km
[35]	No especificado	MATLAB	1200	Variable	Autopista	5000 miles
[36]	No especificado	MATLAB & NS-2	90	50 por segundo	Urbano	7,4x7,4km
[37]	NGSim	NS-2	15	18 cada 100m	Autopista	640m
[38]	NCTUns	NCTUns	80-100	Variable	Carretera	600x400m
[39]	A medida	NS-2	700	100	Urbano	1,2x1,2km
[40]	A medida	No especificado	No especificado	No especificado	No especificado	No especificado
[41]	A medida	No especificado	No especificado	Variable	Autopista	No especificado
[42]	Trazas pre-generadas	NS-2	No especificado	Variable	Urbano	2,4x2,4km
[43]	Trazas pre-generadas	NS-2	200	150	Urbano	2,4x2,4km
[44]	Trazas pre-generadas	NS-2	No especificado	50-100	Carretera	3x3km
[45]	Trazas pre-generadas	NS-2	No especificado	Variable	Urbano	6,7x2,7km

Tabla 1 - Herramientas y parámetros de simulación en publicaciones recientes

Tiempos de simulación. Existe una importante diferencia en los tiempos de simulación utilizados incluso en simulaciones que utilizan el mismo tipo de mapa. Por ejemplo en dos publicaciones que utilizan un mapa de autopista se utilizan 15 segundos [37] y 600 [30]. Los valores más comunes, utilizados en 11 publicaciones, están entre 100 y 600 segundos. Adicionalmente se observa que cerca de un tercio de las publicaciones no especifica este parámetro.

Número de nodos. Los valores más utilizados para este parámetro se encuentran entre 50 y 250 nodos, presente en más de un tercio de las publicaciones analizadas. Este parámetro es también más documentado con tan solo 3 publicaciones que no lo informan.

Tipo y tamaño de escenario. Existen tres tipos básicos de escenarios: urbano, autopista y carreteras secundarias. Entre los más escogidos para las simulaciones son los escenarios urbanos, usados en 12 publicaciones, seguidos por los escenarios de autopistas, usados en 7 publicaciones. De nuevo nos encontramos con un número significativo de publicaciones, 7, que no indican el tipo de mapa utilizado para la simulación o no utilizan ningún tipo de mapa. El tamaño de los escenarios seleccionados toma valores entre 600x400 metros hasta 13,4x12,3 kilómetros, pero los valores más utilizados están entre 1x1 y 5x5 kilómetros.

Estos resultados muestran un conjunto muy heterogéneo características en las simulaciones realizadas para validad las distintas propuestas realizadas.

Capítulo 3

Análisis

3.1 Perspectiva general del sistema

En esta sección se aborda el sistema que se desarrollará para cubrir el objetivo primero definido en el Capítulo 1 de este documento: desarrollar una aplicación que simplifique la generación de experimentos de simulación de VANET.

El sistema a desarrollar deberá facilitar la creación de experimentos de simulación que permitan validar distintas propuestas realizadas sobre VANET. Esta generación se deberá ser simple y directa a través de una única aplicación de fácil utilización.

Por tanto, el principal objetivo definido para el sistema será que ofrezca la funcionalidad adecuada para la generación, edición y también ejecución de experimentos de simulación en un número reducido de pasos, que no requieran amplios conocimientos de las herramientas utilizadas.

Para abordar este objetivo, el sistema deberá ofrecer la posibilidad de editar un conjunto determinado de parámetros tanto de las simulaciones de movilidad como de las simulaciones de red. Las simulaciones resultantes de la edición de estos parámetros se deberán poder guardar para su futuro uso.

Para hacer el sistema accesible a toda la comunidad investigadora deberá ser desarrollado sobre una tecnología que facilite su portabilidad entre distintos sistemas.

A fin de acotar el alcance del sistema en lo que se refiere a su integración con distintos simuladores de red y de movilidad se ha decidido definir un conjunto de simuladores con los que se integrará el sistema en su primera versión, teniendo en cuenta las distintas opciones analizadas en el Capítulo 2: su uso actual además de otros factores.

Considerando el análisis de las soluciones de simulación realizado en el Capítulo 2 se ha decidido utilizar como simulador de red el simulador NS-2 ya que además de tratarse de una herramienta de código abierto y libre distribución, cuenta con una amplia aceptación en la comunidad investigadora.

Se han elegido dos herramientas como origen de las trazas movilidad: el simulador SUMO que permite la ejecución de simulaciones sobre todo tipo de topologías y con distintas configuraciones de movilidad que, a partir de estas simulaciones, genera las trazas de movilidad que serán utilizadas para crear las simulaciones de red; la otra herramienta seleccionada es CityMob que permite la generación de trazas de movilidad en un entorno urbano pudiendo configurar diversos aspectos del escenario desde el número de vehículos hasta puntos de interés o tiempos de parada en semáforos.

Debido a la importancia de las trazas de movilidad utilizadas para la generación de simulación de red [46] el sistema deberá ser extensible para incluir nuevas fuentes de trazas de movilidad y también nuevos simuladores de red. El sistema también deberá permitir su extensión a través de la inclusión de nuevas funcionalidades.

3.2 Arquitectura del sistema

Como hemos visto anteriormente el sistema deberá ser extensible tanto con nuevas funcionalidades como con nuevos simuladores de movilidad y de red. Es por esto por lo que se decide emplear un diseño modular en donde la funcionalidad de las distintas etapas de la creación de un experimento de simulación completo esté claramente separada entre sí. Además de agrupar las distintas etapas de la simulación deberán agruparse también las funcionalidades de soporte, como por ejemplo el acceso a archivos, y las interfaces de la aplicación.

La granularidad con la que se defina la arquitectura es muy importante ya que una granularidad incorrecta a la hora de separar las funcionalidades en distintos componentes puede no aportar ningún beneficio y dificultar el desarrollo y funcionamiento del sistema.

La Figura 4 muestra el diagrama de componentes de la arquitectura preliminar del sistema. En este diseño se observa que la simulación de red y la simulación de movilidad se encuentran separadas en dos componentes independientes, de esta forma se pretende hacer el sistema más extensible, pudiendo así extenderse con nuevos componentes que realicen simulaciones de movilidad o de red.

Además de los componentes que realizan las tareas de simulación se han definido otros dos componentes: el encargado de mostrar la interfaz de usuario adecuada en cada momento de la simulación y el que contiene las utilidades comunes utilizadas por otros componentes del sistema, principalmente operaciones de manipulación de ficheros y ejecución de comandos.

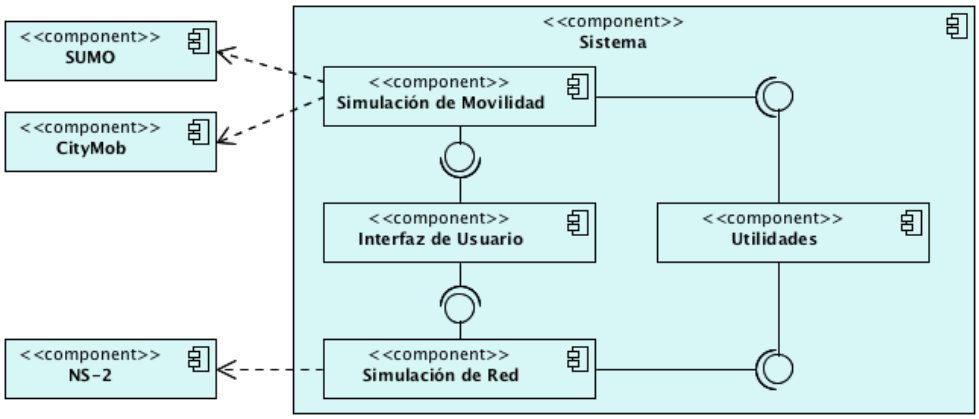

Figura 4 - Arquitectura preliminar del sistema

Junto con los componentes internos del sistema, la Figura 4 también muestra los componentes externos al mismo correspondientes a los simuladores de red y movilidad de los que hará uso el sistema para realizar la ejecución y creación de simulaciones.

3.3 Estudio tecnológico

En esta sección se detalla el estudio de las posibles tecnologías que se pueden aplicar para el desarrollo de los distintos componentes que forman la aplicación. En las siguientes subsecciones se analizarán las tecnologías impuestas junto con las tecnologías aplicables a los distintos componentes definidos en la arquitectura preliminar.

3.3.1 Tecnologías impuestas

La única imposición detectada para el sistema que se desarrollará en este proyecto fin de carrera viene dada por el objetivo del sistema de la generación de experimentos de simulación para NS-2, y que también deberá ofrecer la posibilidad de ejecutarlos. Es por esto por lo que deberá relacionarse con el simulador de red NS-2.

El simulador de red NS-2 está desarrollado con el lenguaje de programación C++ [47]. A pesar de tratarse de un lenguaje que puede ser ejecutado en distintos entornos, NS-2 sólo permite ser ejecutado en entornos Linux no estando soportado su uso en Windows de forma directa, si bien puede ser utilizado a través de herramientas como Cygwin [48]. La relación entre el sistema que se desarrollará y NS-2 se realizará a mediante invocaciones de la línea de comandos.

3.3.2 Tecnologías aplicables al componente *Utilidades*

En este apartado se muestra el estudio de las tecnologías que se tomarán en consideración dentro de las aplicables para el desarrollo del componente *Utilidades;* en él también se describirán las características principales de las distintas tecnologías consideradas.

El componente *Utilidades* se ofrece la funcionalidad de manipulación de ficheros así como de ejecución de comandos a través de la línea de comandos. Al no requerir una interfaz gráfica, mostrar ningún dato y no requerir el uso de librerías poco comunes o específicas para realizar sus operaciones, el abanico de lenguajes para su programación es muy extenso. De todos los lenguajes aplicables se ha hecho una preselección de tres de los más populares y utilizados hoy en día: Java [49], C++ y C# [50].

La primera opción estudiada es el lenguaje Java, un lenguaje de programación orientado a objetos que soporta su ejecución en múltiples entornos al ser su código compilado para ser interpretado por una máquina virtual de Java (JVM). Además de las características propias del lenguaje, existe una ingente cantidad de documentación y manuales disponibles sobre este lenguaje lo que facilita el desarrollo con esta herramienta.

C# es otra de las opciones a considerar. Se trata de un lenguaje orientado a objetos, con una sintaxis muy similar a la de C++ y Java. Su desarrollo está orientado a la plataforma .NET de Microsoft [51] por lo que su uso está orientado a entornos Windows, sin embargo existen opciones para ejecutar aplicaciones desarrolladas con C# en entornos Linux, como Mono [52].

C++ es la tercera opción analizada, se trata de un lenguaje “híbrido” que soporta la orientación a objetos y tiene una sintaxis similar a la de Java y C#. Es un lenguaje compilado por lo

que su código requiere ser compilado en los distintos entornos en los que vaya a ser utilizado, pero a cambio ofrece un mayor rendimiento en su ejecución en los distintos sistemas..

3.3.3 Tecnologías aplicables al componente *Simulación de Movilidad*

En este apartado se muestra el estudio de las tecnologías que se tomarán en consideración dentro de las aplicables para el desarrollo del componente *Simulación de Movilidad*; en él también se describirán las características principales de las distintas tecnologías consideradas.

La principal funcionalidad de este componente es generar las trazas de movilidad que serán usadas para crear los experimentos de simulación. Para llevar a cabo esta función se hará uso de herramientas ya existentes. Las herramientas aplicables para esto se encuentran detalladas en el Capítulo 2, si bien a continuación se analizarán las características del simulador de red más utilizado en el conjunto de publicaciones: SUMO. Además, se incluye entre las herramientas aplicables CityMob que no es un simulador de movilidad si no un generador de trazas de movilidad, que es el resultado final buscado por las simulaciones de movilidad.

SUMO es un simulador desarrollado en C++ utilizando para ello únicamente librerías estándar y portables. Parte de la funcionalidad que ofrece SUMO se lleva a cabo con scripts de Python [53] por tanto el entorno en el que se ejecuten deberá contar con un intérprete de Python. Adicionalmente cuenta con versiones compiladas para Windows y Linux. La interacción con SUMO se realizará invocando sus distintas funcionalidades a través de la línea de comandos.

CityMob es, como se ha indicado previamente, un generador de trazas de movilidad desarrollado en Java lo que facilita su portabilidad a todos los entornos compatibles con esta tecnología. La relación entre el sistema que se desarrollará y CityMob se realizará ejecutando métodos expuestos por el componente o instanciando directamente distintas clases ofrecidas por el componente. Se deberá modificar el código de CityMob para adaptarlo a su uso en el sistema y permitir la generación de varias trazas de movilidad de forma simultánea para lograr así crear experimentos de simulación con varias simulaciones de red.

Respecto a los lenguajes de programación aplicables a este componente se observa que las necesidades que deben cubrir son similares a las indicadas para el componente *Utilidades*, por lo que se ha hecho de nuevo una preselección de tres lenguajes que serán considerados: Java, C# y C++. La sección 3.3.2 profundiza en el estudio de estas tres alternativas.

3.3.4 Tecnologías aplicables al componente *Simulación de Red*

En este apartado se muestra el estudio de las tecnologías que se tomarán en consideración dentro de las aplicables para el desarrollo del componente *Simulación de Red*; en él también se describirán las características principales de las distintas tecnologías consideradas.

Las principales funcionalidades de este componente serán la creación y la ejecución de experimentos de simulación, es decir crear y ejecutar las distintas simulaciones de red que pueden formar un experimento.

Para la generación de simulaciones de red se han considerado dos herramientas sobre las que se puede apoyar el componente: MOVE y TraNS. Ambas herramientas, analizadas con detalle en el Capítulo 2, cuentan con la opción de crear simulaciones de red para NS-2 a par-

tir de trazas de movilidad de SUMO, esta funcionalidad se realiza tomando las trazas generadas tras ejecutar una simulación de SUMO y recorriéndolas analizando los movimientos y posiciones de los distintos nodos simulados, convirtiéndolos en movimientos y posiciones en el formato aceptado por el simulador de red NS-2. Ambas aplicaciones están implementadas en Java y tienen un conjunto de clases bien definido encargado de tratar las trazas resultantes de una simulación de movilidad con SUMO y, junto con la configuración especificada por el usuario, generar simulaciones de red. Para su uso en el sistema que se desarrollará se deberán realizar modificaciones a las opciones para la generación de simulaciones de red que el sistema ofrecerá y que no están ofrecidas actualmente por MOVE o TraNS, como por ejemplo el uso de RSU.

Respecto a los lenguajes de programación aplicables a este componente se observa que las necesidades que deben cubrir son similares a las indicadas para el componente *Utilidades*, por lo que se ha hecho de nuevo una preselección de tres lenguajes que serán considerados: Java, C# y C++. La sección 3.3.2 profundiza en el estudio de estas tres alternativas.

3.3.5 Tecnologías aplicables al componente *Interfaz de Usuario*

En este apartado se muestra el estudio de las tecnologías que se tomarán en consideración dentro de las aplicables para el desarrollo del componente *Interfaz de Usuario*; en él también se describirán las características principales de las distintas tecnologías consideradas.

Las necesidades que se deben cubrir con este componente son distintas de las vistas en los anteriores puesto que es el único responsable de mostrar las distintas pantallas que permitirán la interacción del usuario con el sistema. Por ello, la selección de los lenguajes aplicables a este componente deberá tener en cuenta las funcionalidades ofrecidas por cada lenguaje en el ámbito de interfaces gráficas de usuario.

De nuevo el abanico de lenguajes disponibles que soporten adecuadamente el uso de interfaces gráficas de usuario es muy amplio, por lo tanto se van a hacer una preselección de los lenguajes a considerar para el desarrollo de este componente: C#, Java, Ruby [54] y C++.

C# es un lenguaje orientado a objetos que ofrece a través de su librería *Windows Forms* [55] un amplia gama de posibilidades para el desarrollo de interfaces gráficas de usuario y, a través del entorno de desarrollo Visual Studio [56], un editor gráfico de estas interfaces lo que simplifica la creación de las interfaces gráficas permite obtener unos resultados homogéneos entre las distintas interfaces.

Java, también orientado a objetos, cuenta con dos librearías distintas para el desarrollo de interfaces gráficas de usuario: *Swing* [57] y *AWT* [58]. Al igual que con C# las interfaces se pueden desarrollar con la ayuda de un editor gráfico, como el que ofrece el entorno de desarrollo NetBeans [59].

Ruby es un lenguaje interpretado orientado a objetos que ofrece, mediante el uso distintas librerías, la posibilidad de desarrollar interfaces de usuario, de entre estas librerías para interfaces gráficas de usuario destacan *Shoes* [60] y *GTK* [61]. Siendo un lenguaje interpretado soportado en distintos entornos, el código de Ruby es portable entre estos entornos.

Al igual que otras tecnologías, C++ ofrece la posibilidad de desarrollar interfaces de usuario mediante el uso de librerías externas entre las que se han seleccionado *Qt* [62] y *GTK*.

3.4 Selección de tecnologías no impuestas

En esta sección se mostrarán las tecnologías no impuestas seleccionadas para el desarrollo del sistema junto con el razonamiento que ha llevado a esas decisiones.

Del conjunto de tecnologías aplicables a los distintos componentes que se han visto en el apartado 3.3 se ha tomado la decisión de escoger una única tecnología para el desarrollo de toda la aplicación de forma que se simplifique así el desarrollo y la integración de los componentes que compondrán el sistema.

De todo el conjunto de herramientas para la generación de trazas de movilidad, principalmente simuladores, se han escogido dos: SUMO y CityMob. SUMO ha sido elegido por ser una de las opciones más populares para la simulación de movilidad en distintos ámbitos y ofrecer una amplia gama de funcionalidades como es la creación de mapas automáticamente o la creación de rutas aleatorias. CityMob ha sido escogido por permitir la generación de trazas sobre una topología común en los experimentos de simulación como es la de rejilla, estas trazas cuentan con un amplio abanico de opciones de configuración que hacen más atractivo su uso. Para permitir la generación de varias simulaciones de red por experimento se debe modificar el código de CityMob.

Para la generación de simulaciones de red se ha escogido MOVE, del que se utilizará solamente la funcionalidad relacionada con la generación de simulaciones para NS-2. Se ha escogido MOVE frente a TraNS porque el desarrollo de esta última está suspendido y no soporta nuevas versiones de SUMO. La funcionalidad escogida para convertir las trazas de movilidad de SUMO y generar simulaciones de red deberá de ser modificada en el código fuente para adaptarla al sistema que se va a desarrollar.

Respecto a los lenguajes de programación, dentro de las tecnologías aplicables que se han identificado en los apartados 3.3.2, 3.3.3 y 3.3.4 se ha descartado en primer lugar C# ya que, si bien cumple ampliamente los requisitos esperados por el lenguaje de programación a utilizar para el desarrollo, no se trata de un lenguaje portable y su uso en el entorno impuesto por el uso de NS-2, Linux, es sensiblemente más complejo y problemático que las otras opciones identificadas.

Dentro de las opciones restantes se descarta Ruby para el desarrollo del componente *Interfaz de Usuario* debido a la decisión de utilizar un único lenguaje de programación para el desarrollo de todo el sistema y a que no se haya tomado en consideración en el desarrollo de los demás componentes debido a que cuenta con un rendimiento sensiblemente inferior a las demás opciones consideradas y además el equipo desarrollador no está familiarizado con la sintaxis del lenguaje, a diferencia de las otras opciones mencionadas.

De los dos lenguajes de programación restantes entre los que elegir se ha elegido Java frente a C++. Esta decisión ha estado motivada por la mayor familiaridad del equipo desarrollador con el lenguaje, lo que facilitará la implementación y disminuirá el riesgo asociado al desconocimiento del lenguaje. Adicionalmente Java cuenta con una amplia documentación tanto propia como externa. En lo que se refiere a las interfaces gráficas de usuario Java sale perdiendo frente a C++ ya que las interfaces multiplataforma, como es la ofrecida por Java con su librearía *Swing*, tienden a ofrecer resultados muy heterogéneos dependiendo de la plataforma donde se ejecuten, pero a cambio mantienen la portabilidad del sistema. Respecto a las funcionalidades requeridas por el resto de componentes de la aplicación, se ha visto que ambos lenguajes los cubren, si bien Java ha destacado en la sencillez a la hora de implementar y depurar las aplicaciones que se desarrollan en el. Todas estas razones, junto con el hecho de

que las herramientas cuyo código se ha modificar (MOVE y CityMob) están desarrolladas en Java, han afianzado a Java con el lenguaje de programación a utilizar para el desarrollo del sistema.

3.5 Arquitectura definitiva de alto nivel

Partiendo del diseño inicial presentado en la sección 3.2 y teniendo en consideración las tecnologías impuestas y las decisiones tomadas respecto a las tecnologías a utilizar se ha modificado el diagrama de componentes de la Figura 4 para reflejar la arquitectura definitiva del sistema.

La Figura 5 muestra este diagrama de componentes con la arquitectura definitiva. En él los distintos componentes y el propio sistema tienen ya los nombres que deberán tener en la implementación, dichos nombres están en inglés debido a que de esta forma podrá ser utilizada, comprendida y ampliada por un abanico mucho mayor de usuarios.

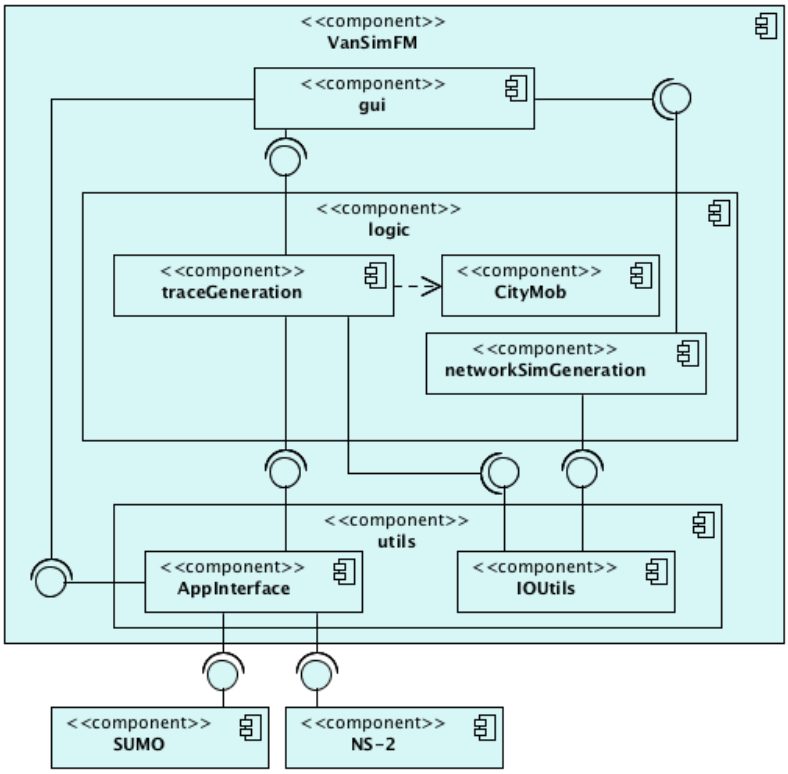

Figura 5 - Diagrama de componentes definitivo

Los principales cambios realizados con respecto al diagrama de componentes de la arquitectura preliminar son el cambio de enfoque y nombre del componente *Simulación de Movilidad*, que pasa a llamarse *traceGeneration* con su funcionalidad orientada únicamente a la generación de trazas apoyándose en el componente externo *SUMO* que ejecutará la simulación de movilidad, a través del componente *AppInterface*, o bien utilizando generadores de trazas como CityMob. El mismo cambio de enfoque se ha llevado a cabo con el componente *Simulación de Red*, que ahora pasa a ser *networkSimGeneration* y su función se centrará en la generación de las simulaciones de red, sacando la ejecución de la simulación al componente externo NS-2 a través del componente *AppInterface* al igual que se hará con SUMO. Por otra parte, el componente *Interfaz de Usuario* se ha renombrado a *gui*.

Dado que se ha detectado la necesidad de un componente encargado de ejecutar las distintas herramientas externas a través de línea de comandos se ha descompuesto el componente *Utilidades* en dos: uno con la funcionalidad indicada en el diseño preliminar, la manipulación de ficheros, y otro con la funcionalidad indicada para la ejecución de componentes externos a través de línea de comandos. Estas funcionalidades se ofrecerán al componente *gui* para que este pueda iniciar la ejecución de simulaciones de red y al componente *traceGeneration* para que pueda iniciar la ejecución de simulaciones de movilidad para la generación de trazas.

Otras interacciones entre componentes son las que se realizan a través de los métodos ofrecidos por el componente *IOUtils*, llamado *Utilidades* en el diseño preliminar, para la manipulación de ficheros; estos métodos serán consumidos por los componentes *networkSimGeneration* y *traceGeneration* durante la generación de simulaciones de red y trazas de movilidad respectivamente. Estos dos componentes a su vez ofrecerán métodos para configurar e iniciar la generación de simulaciones y trazas de movilidad que serán consumidos por la interfaz *gui*.

3.6 Casos de uso

En esta sección se muestra el diagrama de casos de uso de la aplicación junto con la definición textual de cada uno de los casos de uso presentes en el diagrama. El estudio de los casos de uso descritos facilitará la extracción de requisitos en fases posteriores del proyecto.

3.6.1 Diagrama de casos de uso

La Figura 6 muestra el diagrama que representa los tres casos de uso identificados para el proyecto. En dicho diagrama se pueden apreciar las tres funcionalidades principales ofrecidas por la aplicación: creación de trazas de movilidad a partir de una determinada configuración que podrá ser creada en el acto o estar guardada previamente, configuración de la simulación de red con datos relativos al medio de transmisión, a la simulación de movilidad y al uso o no de RSU en la simulación de red; y por último ejecución de las simulaciones de red que componen un experimento de simulación. Como se puede apreciar, sólo se ha definido un único actor con el que interactuará el sistema: el usuario de la aplicación.

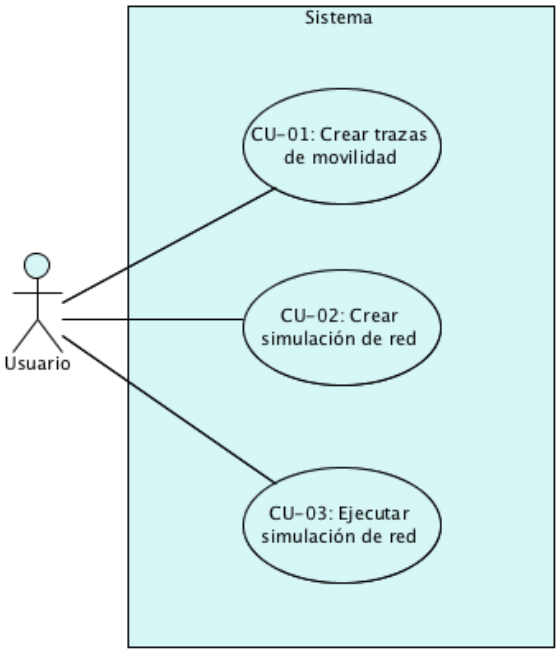

Figura 6 - Diagrama de casos de uso

3.6.2 Definición textual de los casos de uso

En este apartado se muestra información detallada de los distintos casos de uso identificados en el diagrama de casos de uso del apartado anterior. Para mostrar esta información se seguirá el formato definido en la plantilla 1 del Anexo 3.

La Tabla 2 muestra el primer caso de uso identificado que consiste en la generación de trazas de movilidad con una de las fuentes de trazas definidas en el sistema a partir de una configuración especificada por el usuario. El usuario podrá elegir la aplicación con la que generar las trazas y utilizar o no un escenario ya existente para lo que deberá definir la nueva configuración a utilizar para la generación de trazas de movilidad.

Identificador: CU-01	
Nombre:	Crear trazas de movilidad
Autor:	José Munera López
Descripción Generar trazas de movilidad de un experimento a partir de la configuración definida por el usuario.	
Actores: Usuario de la aplicación.	
Precondiciones: • La aplicación está arrancada.	
Poscondiciones: • Generación de trazas de movilidad de un experimento a partir de un escenario definido por el usuario.	
Flujo Normal: 1. Seleccionar el programa generador de trazas de movilidad. 2. Crear el escenario sobre el que ejecutar la simulación. 3. Generar trazas de movilidad ejecutando la simulación. 4. Finalizar	
Flujo Alternativo: • Se ha seleccionado CityMob como aplicación para generar las trazas 2a. Especificar parámetros del mapa. 3a. Especificar parámetros de la movilidad. 4a. Ir al paso 3. • Se ha seleccionado SUMO como aplicación para generar las trazas 2a. Especificar mapa existente, a importar o parámetros para su creación. 3a. Especificar movilidad existente o parámetros para la creación de movilidad aleatoria. 4a. Ir al paso 2. • No se va a crear un escenario (Sólo con SUMO) 2a. Seleccionar un escenario existente. 3a. Ir al paso 3.	

Tabla 2 - Definición textual del caso de uso CU-01

La Tabla 3 muestra el segundo caso de uso definido consistente en la configuración y creación de la parte correspondiente a la simulación de red de un experimento de simulación. Para la configuración de la simulación de red el usuario deberá especificar valores para los distintos datos que desee modificar. En el caso de que se desee incluir en la simulación una o más RSU el usuario deberá marcar dicha opción y añadir las posiciones de dichas RSU así como la aplicación con la que participarán en la simulación.

Identificador: CU-02	
Nombre:	Crear simulación de red
Autor:	José Munera López
Descripción Configurar y crear la simulación de red de un experimento de simulación.	
Actores: Usuario de la aplicación.	
Precondiciones: • La aplicación está arrancada. • Se han generado trazas de movilidad.	
Poscondiciones: • Generación de simulación de red de un experimento.	
Flujo Normal: 1. Configurar los distintos aspectos de la simulación de red del experimento. 2. Definir la aplicación a simular en los vehículos y el porcentaje de vehículos que participarán en la simulación. 3. Definir las opciones de creación de trazas resultantes de la simulación de red. 4. Finalizar	
Flujo Alternativo: • El experimento de simulación incluye RSU. 3a. Marcar la opción “utilizar RSU”. 4a. Definir la aplicación a simular en las RSU. 5a. Añadir las posiciones de las RSU que se desean incluir en la simulación de red. 6a. Ir al paso 4.	

Tabla 3 - Definición textual del caso de uso CU-02

Por último, la Tabla 4 muestra el tercer caso de uso identificado, consistente en la ejecución de un experimento de simulación creado previamente. Para la ejecución del experimento de simulación el usuario podrá especificar la simulación de red en concreto que desea simular o si desea simular todas las correspondientes al experimento. Adicionalmente el usuario podrá indicar si desea visualizar las trazas NAM generadas durante la simulación o ejecutar scripts AWK [84] para el procesado de datos generados por la simulación. En caso de haber seleccionado alguna de estas opciones se llevará a cabo una vez se hayan ejecutado todos los pasos previos y, por tanto, se haya ejecutado la simulación de red.

Identificador: CU-03	
Nombre:	Ejecutar simulación.
Autor:	José Munera López
Descripción Ejecutar un experimento de simulación existente.	
Actores: Usuario de la aplicación.	
Precondiciones: • La aplicación está iniciada. • Se ha creado el experimento de simulación.	
Poscondiciones: • Se ha ejecutado el experimento de simulación. • Se ha generado la información requerida sobre la ejecución del experimento de simulación.	
Flujo Normal: 1. Seleccionar un fichero *tcl* del experimento de simulación. 2. Indicar si se quieren ejecutar todas las simulaciones de red existentes en el directorio del fichero seleccionado 3. Indicar si se desean ejecutar los scripts AWK existentes en la carpeta del experimento de simulación. 4. Indicar si se desean visualizar las trazas NAM generadas en la simulación. 5. Ejecutar simulación.	
Flujo Alternativo: • Se ha marcado la opción para visualizar las trazas NAM. 6a. Se muestra la pantalla del visualizador de trazas NAM. • Se ha marcado la opción para ejecutar los scripts AWK. 6b. Se ejecutan los scripts sobre los archivos resultantes de la simulación de red, generando a su vez archivos con la información procesada..	

Tabla 4 - Definición textual del caso de uso CU-03

3.7 Requisitos de software

En esta sección se presentan los requisitos de software identificados que deberán ser cubiertos para cumplir con los objetivos especificados para este proyecto. Las distintas subsecciones que componen esta sección detallan los requisitos de los distintos tipos definidos por la metodología de la ESA [63]. Se muestran únicamente los tipos de requisitos utilizados en esta definición.

3.7.1 Requisitos funcionales

La Tabla 5 agrupa la información de los distintos requisitos funcionales de acuerdo al formato definido en la plantilla 2 del Anexo 3.

Requisitos de Software				
Tipo: Funcional				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RF-01	Crear experimentos.	Crear experimentos de simulación para el simulador de red NS-2.	Alta	Alta
RF-02	Ejecutar experimentos.	Ejecutar experimentos de simulación para el simulador de red NS-2.	Alta	Alta
RF-03	Crear trazas de movilidad con SUMO.	Crear una o más trazas de movilidad a partir de la simulación de escenarios de movilidad de SUMO existentes.	Alta	Alta
RF-04	Crear trazas de movilidad con CityMob.	Crear una o más trazas de movilidad a partir de patrones de movilidad creados con CityMob.	Alta	Alta
RF-05	Visualizar simulación de SUMO.	Visualización de las simulaciones de movilidad ejecutadas con SUMO.	Alta	Baja
RF-06	Crear de escenarios de movilidad de SUMO.	Crear escenarios de movilidad de SUMO completos.	Alta	Alta
RF-07	Seleccionar mapa de SUMO.	Seleccionar un mapa de SUMO ya existente.	Alta	Alta
RF-08	Importar mapa de OSM en SUMO.	Importar un mapa en formato OSM para su uso con SUMO.	Media	Alta
RF-09	Crear mapa de SUMO aleatorio.	Crear automáticamente un mapa de SUMO con forma aleatoria.	Media	Alta
RF-10	Crear mapa con forma de rejilla automáticamente.	Crear automáticamente un mapa con forma de rejilla..	Media	Alta
RF-11	Crear mapa con forma de telaraña automáticamente.	Crear automáticamente un mapa con forma de telaraña.	Media	Alta

Requisitos de Software				
Tipo: Funcional				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RF-12	Seleccionar movilidad de SUMO ya existente.	Seleccionar un fichero de movilidad de SUMO ya existente.	Alta	Alta
RF-13	Crear movilidad aleatoria de SUMO.	Crear movilidad aleatoria de SUMO indicando el número de simulaciones y de vehículos.	Media	Alta
RF-14	Introducir RSU en la simulación.	Introducir RSU en la simulación de red.	Media	Alta
RF-15	Introducir código en las simulaciones de red.	Introducir bloques de código dentro de las simulaciones de red de un experimento como parte de la configuración de red..	Media	Baja
RF-16	Ejecutar scripts AWK.	Ejecutar scripts AWK sobre los datos generados por la simulación de red.	Media	Baja
RF-17	Visualizar trazas Nam.	Visualizar trazas para Nam generadas por la simulación de red.	Alta	Media
RF-18	Mostar información de la aplicación	Mostrar información sobre la aplicación, la versión y sus autores.	Alta	Media

Tabla 5 - Requisitos funcionales

3.7.2 Requisitos no funcionales

La Tabla 6 agrupa la información de los distintos requisitos no funcionales de acuerdo al formato definido en la plantilla 2 del Anexo 3. Como se ha indicado anteriormente la tabla no menciona los tipos de requisitos no utilizados

Requisitos de Software				
Tipo: Interfaz				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RNF-01	Comunicación con SUMO.	Las funcionalidades de SUMO se ejecutan mediante comandos.	Media	Media
RNF-02	Comunicación con NS-2.	Las funcionalidades de NS-2 se ejecutan mediante comandos	Media	Media
Tipo: Operacional				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RNF-03	Idioma de la aplicación.	La aplicación muestra toda la información en inglés.	Media	Alta
RNF-04	Validar datos introducidos.	Validar los datos introducidos por el usuario.	Alta	Alta
RNF-05	Mensajes de error.	Mostrar al usuario mensajes de error en caso de producirse excepciones durante la ejecución.	Alta	Alta
RNF-06	Pantallas comunes.	La pantalla de generación de experimentos de simulación a partir de trazas de movilidad para CityMob tiene los mismos campos y estructura que la usada con SUMO.	Media	Media
RNF-07	Simulación de red.	Una simulación de red está formada por la configuración del intercambio de datos entre los nodos que intervienen junto a las trazas de movimientos que estos realizan, obtenidas de una simulación de movilidad.	Alta	Alta
RNF-08	Experimento de simulación.	Un experimento de simulación está compuesto por una o varias simulaciones de red.	Media	Alta

Requisitos de Software				
Tipo: Interfaz				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RNF-09	Escenario de movilidad de SUMO.	Un escenario de movilidad de SUMO está compuesto por un mapa y una definición implícita o explícita de los movimientos realizados por los vehículos sobre este mapa.	Media	Alta

Tabla 6 - Requisitos no funcionales

3.7.3 Requisitos inversos

La Tabla 7 agrupa la información de los distintos requisitos inversos de acuerdo al formato definido en la plantilla 2 del Anexo 3.

Requisitos de Software				
Tipo: Inverso				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RI-01	Inclusión manual de RSU.	No se añaden RSU de forma automática a las simulaciones de red.	Media	-
RI-02	Origen de trazas de movilidad.	SUMO y CityMob son el único origen de trazas de movilidad soportado por el sistema.	Media	-
RI-03	Simulador de red.	NS-2 es el único simulador de red que soporta el sistema.	Alta	-
RI-04	Formatos de mapas para importar.	OSM es el único formato soportado por el sistema para la importación de mapas.	Media	-
RI-05	No se modifican experimentos de simulación existentes.	No se modifican experimentos de simulación ya existentes.	Media	-
RI-06	Aplicaciones a simular con NS-2.	Se simulan como máximo dos aplicaciones en cada experimento: la que incluyen los vehículos y la que incluyen las RSU.	Alta	
RI-07	No se valida el código de usuario.	No se valida el código que introduce el usuario durante la configuración de la simulación de red.	Alta	

Tabla 7 - Requisitos inversos

3.8 Diseño del plan de pruebas de aceptación

Una vez se ha definido las características de la aplicación que se desarrollará a lo largo de esta proyecto fin de carrera se puede proceder a realizar el plan de pruebas de aceptación de la misma.

En esta sección se detalla el plan de pruebas de aceptación de la aplicación en donde se muestran las distintas pruebas que se han de superar y los resultados esperados de las mismas para que se superen las pruebas de aceptación.

La Tabla 8 detalla las pruebas de aceptación definidas de acuerdo al formato definido en la plantilla 3 del Anexo 3.

Pruebas de aceptación			
Id	**Elementos probados**	**Entrada**	**Salida**
PA-01	RF-01, RF-03	Crear un experimento de simulación utilizando trazas de movilidad de SUMO. El experimento deberá estar compuesto por varias simulaciones de red. Sus características serán las siguientes: mapa de entorno urbano de 4x3km, tiempo de simulación 100 segundos y 50 nodos.	Experimento de simulación compuesto por varias simulaciones de red en donde se reflejen correctamente las opciones indicadas por el usuario.
PA-02	RF-01, RF-04	Crear un experimento de simulación utilizando trazas de movilidad de CityMob. El experimento deberá estar compuesto por varias simulaciones de red. Sus características serán las siguientes: mapa de entorno urbano con planta de rejilla de 5x5km, tiempo de simulación 100 segundos y 50 nodos.	Experimento de simulación compuesto por varias simulaciones de red en donde se reflejen correctamente las opciones indicadas por el usuario.
PA-03	RF-01, RF-03	Crear un experimento de simulación utilizando trazas de movilidad de SUMO. El experimento deberá estar compuesto por una simulación de red. Sus características serán las siguientes: mapa de entorno urbano de 4x3km, tiempo de simulación 100 segundos y 50 nodos.	Experimento de simulación compuesto por una única simulación de red en donde se reflejen correctamente las opciones indicadas por el usuario.
PA-04	RF-01, RF-04	Crear un experimento de simulación utilizando trazas de movilidad de CityMob. El experimento deberá estar compuesto por una simulación de red. Sus características serán las siguientes: mapa de entorno urbano con planta de rejilla de	Experimento de simulación compuesto por una única simulación de red en donde se reflejen correctamente las opciones indicadas por el usuario.

Pruebas de aceptación			
Id	**Elementos probados**	**Entrada**	**Salida**
		5x5km, tiempo de simulación 100 segundos y 50 nodos.	
PA-05	RF-02	Ejecutar un experimento de simulación compuesto por una única simulación de red. La simulación del experimento utilizará la movilidad proveniente de las trazas del experimento creado en PA-04.	Se ejecutan correctamente la simulación de red de acuerdo a las especificaciones indicadas en el fichero de simulación, generándose los ficheros resultantes esperados.
PA-06	RF-02	Ejecutar un experimento de simulación compuesto por varias simulaciones de red. La configuración del experimento será la especificada por defecto por el sistema y utilizará la movilidad proveniente de las trazas del escenario creado en PA-02.	Se ejecutan correctamente las simulaciones de red de acuerdo a las especificaciones indicadas en los ficheros de simulación, generándose los ficheros resultantes esperados.
PA-07	RF-06, RF-09, RF-10, RF-11, RF-13	Crear tres escenarios de movilidad con SUMO utilizando generando automáticamente un mapa totalmente aleatorio, uno de tipo rejilla y otro de tipo telaraña. La movilidad para todos estos mapas será aleatoria con 50 coches en un tiempo de simulación de 100 segundos.	Se crea un escenario de movilidad para cada tipo de mapa utilizado de acuerdo a la estructura de directorios definida para la aplicación.
PA-08	RF-06, RF-08, RF-13	Crear un escenario de movilidad con SUMO utilizando un mapa con formato OSM importado de SUMO y movilidades aleatorias. El número de vehículos será 250 y el tiempo de simulación 600 segundos.	Se crea un escenario de movilidad para el mapa y la movilidad definidas de acuerdo a la estructura de directorios definida para la aplicación.
PA-09	RF-06, RF-07, RF-13	Crear un escenario de movilidad con SUMO utilizando un mapa creado previamente y movilidades aleatorias. El número de vehículos será 250 y el tiempo de simulación 600 segundos.	Se crea un escenario de movilidad para el mapa y la movilidad definidas de acuerdo a la estructura de directorios definida para la aplicación.

Pruebas de aceptación			
Id	**Elementos probados**	**Entrada**	**Salida**
PA-10	RF-12	Crear un escenario de movilidad con SUMO utilizando un mapa y movilidades creadas previamente.	Se crea un escenario de movilidad para el mapa y la movilidad definidas.
PA-10	RF-03	Crear trazas de movilidad con SUMO a partir de un escenario previamente creado. El número de vehículos será 50 y el tiempo de simulación 100 segundos.	Se ejecuta la simulación de movilidad de un escenario previamente definido y se generan las trazas correspondientes a la configuración del escenario.
PA-11	RF-05	Crear trazas de movilidad con SUMO a partir de un escenario previamente creado y visualizar la simulación de movilidad. El número de vehículos será de 50 y el tiempo de simulación 100 segundos.	Se ejecuta la simulación de movilidad de un escenario previamente definido y se generan las trazas correspondientes a la configuración del escenario. Adicionalmente se visualiza la simulación de movilidad en una nueva ventana.
PA-13	RF-04	Crear trazas de movilidad con CityMob. La configuración para la generación de dichas trazas será la siguiente: número de nodos 50 y 250, tiempo de simulación 100 y 600 segundos, tamaño del mapa 5x4 Km.	Se crean trazas de movilidad a partir de la configuración indicada por el usuario.
PA-14	RF-01, RF-14	Crear un experimento de simulación especificando aplicación y posiciones para RSU. Se utilizará como punto de partida la simulación de movilidad de PA-10. Se definirán 4 RSU junto con sus posiciones.	Se crea un experimento de simulación que contiene en los ficheros de simulación de red las posiciones y definiciones del comportamiento de las RSU indicadas por el usuario.
PA15	RF-01, RF-15	Crear un experimento de simulación especificando bloques de código a añadir en los ficheros de simulación. Se utilizará como punto de partida la simulación de movilidad de PA-10 añadiendo bloques de código fácilmente identificable en el fichero tcl final.	Se crea un experimento de simulación que contiene en los ficheros de simulación de red los bloques de código añadidos por el usuario a través del sistema.

Pruebas de aceptación			
Id	**Elementos probados**	**Entrada**	**Salida**
PA-16	RF-02, RF-17	Ejecutar un experimento de simulación visualizando posteriormente las trazas NAM.	Se ejecuta un experimento de simulación y visualización automática de las trazas NAM de todas las simulaciones de red ejecutadas.
PA-17	RF-02, RF-16	Ejecutar un experimento de simulación procesando posteriormente los ficheros que genere con scripts AWK.	Ejecución de un experimento de simulación y posterior ejecución de los scripts AWK contenidos en la misma carpeta que procesan los datos generados por la simulación de red.
PA-18	RF-18	Mostar la pantalla con información sobre la aplicación y sus autores.	Se muestra una ventana de la aplicación con la información más destacada relativa a la aplicación y sus autores.

Tabla 8 - Pruebas de aceptación

Capítulo 4

Diseño detallado

4.1 Diseño de Software

En esta sección se mostrará una descripción detallada de los distintos componentes identificados en el capítulo de análisis. En el caso de que durante esta fase se detecte la necesidad de descomponer un componente en varios subcomponentes se detallarán estos nuevos subcomponentes en la misma subsección por razones de claridad.

Por razones de simplicidad, se han obviado los métodos y variables que tuvieran poco interés para la implementación, como por ejemplo los métodos de obtención y asignación de variables.

La Figura 7 muestra la arquitectura definitiva del sistema señalando la subsección en la que se tratará cada uno de los componentes identificados, a modo de índice gráfico. No se detallarán los componentes que no han sido desarrollados en este proyecto, es decir NS-2, SUMO y CityMob.

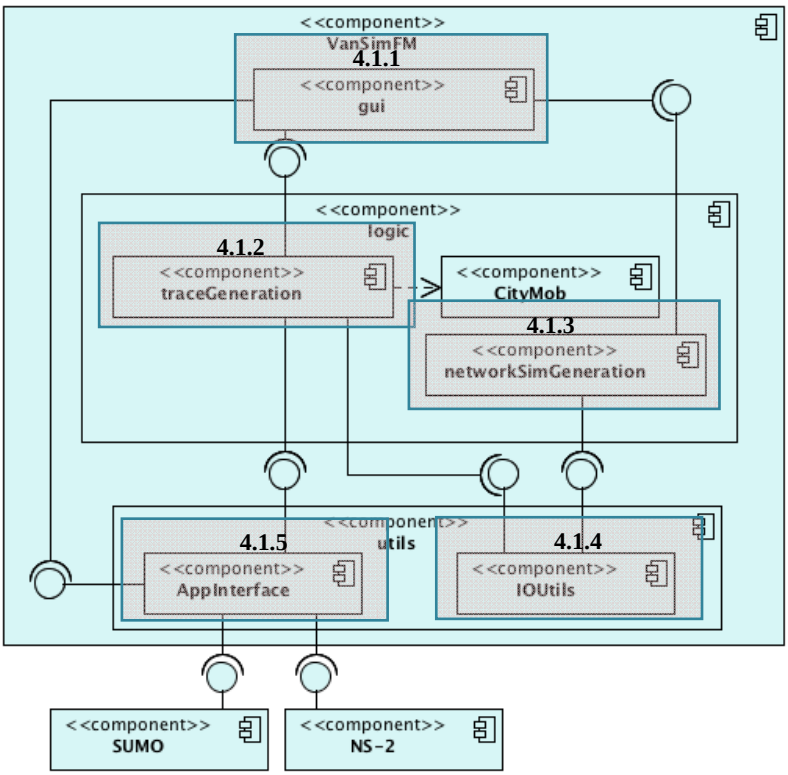

Figura 7 - Diagrama de componente definitivo

4.1.1 Componente *gui*

La funcionalidad de este componente se centra en mostrar las distintas ventanas de la interfaz de usuario. Se ha refinado este componente dividiéndolo en tres subcomponentes nuevos: *MainAppGui*, *traceGenerationGUI* y *tclGenerationGUI*. La Figura 8 muestra esta descomposición.

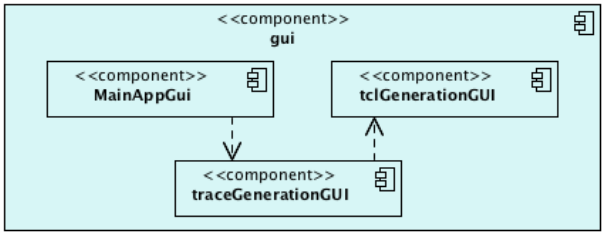

Figura 8 - Descomposición del componente *gui*

4.1.1.1 Componente *MainAppGui*

Este componente mostrará la interfaz principal de la aplicación así como las ventanas de configuración y de información sobre la aplicación. En estas ventanas se mostrará al usuario la pantalla inicial de la aplicación en donde el usuario podrá decidir la acción que quiere realizar: generar experimentos de simulación utilizando una de las fuentes de trazas de movilidad incluidas, ejecutar experimentos de simulación, configurar el directorio donde se almacenarán y recuperarán las simulaciones y por último visualizar información sobre la aplicación y sus autores.

La Figura 9 muestra el diagrama de clases que forman el componente.

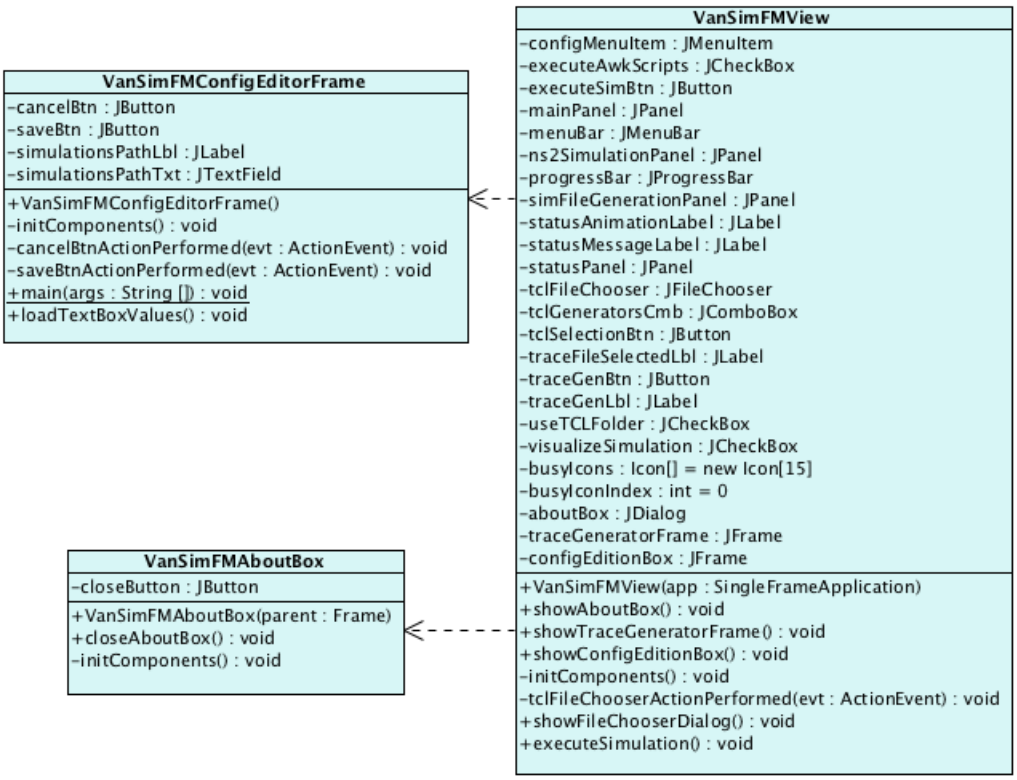

Figura 9 - Diagrama de clases del componente *mainAppGUI*

Como se puede apreciar este componente está compuesto por 3 clases cada una encargada de mostrar una de las tres ventanas principales de la aplicación: ventana principal de la aplicación, *VanSimFMView*; ventana de configuración, *VanSimFMConfigEditorFrame*; y ventana con información general sobre la aplicación, *VanSimFMAboutBox*.

La interacción del componente con los demás componentes del sistema se realiza consumiendo el método e*xecuteCmd*, ofrecido por el componente *AppInterface* que ejecuta en línea de comandos un comando recibido como parámetro.

4.1.1.2 Componente *traceGenerationGUI*

Este componente muestra las distintas interfaces correspondientes a la generación de trazas de movilidad. Cada una de las clases que la forma está relacionada con un simulador específico. En esta interfaz se ofrecerán al usuario las opciones específicas de cada generador de trazas de movilidad permitiéndole configurar esta generación.

La Figura 10 muestra el diagrama de clases que forman el componente.

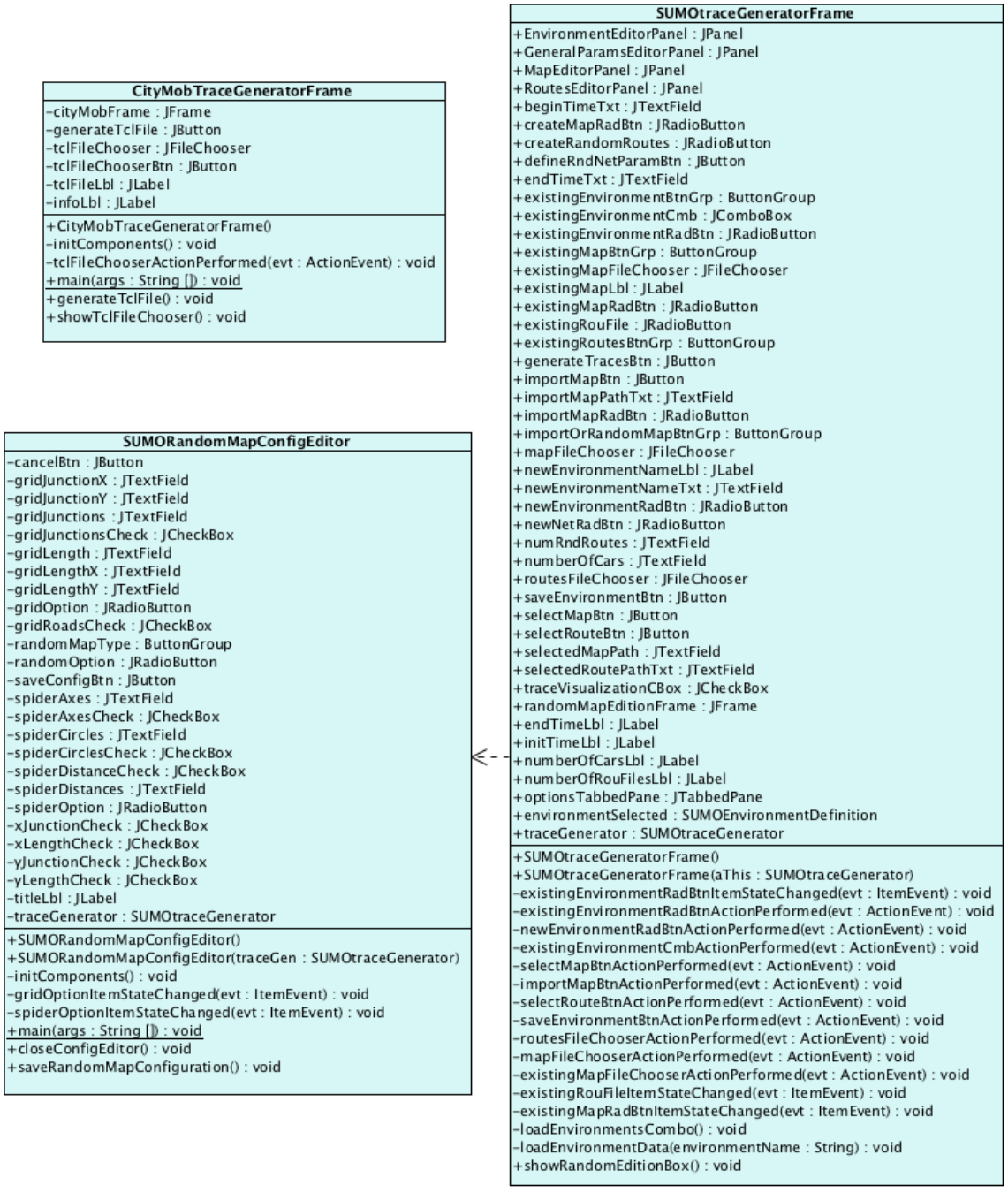

Figura 10 - Diagrama de clases del componente *traceGenerationGUI*

De nuevo se puede apreciar que el componente únicamente cuenta con las clases responsables de mostrar las ventanas de generación de trazas de movilidad: *SUMOtraceGeneratorFrame* y *CityMobTraceGeneratorFrame*. Además de estas dos clases está la clase *SUMORandomMapConfigEditor* que es utilizada por *SUMOtraceGeneratorFrame* para mostrar la ventana de generación automática de mapas con SUMO.

La interacción del componente con los demás componentes del sistema se realiza consumiendo los siguientes métodos:

- ***saveEnvironment*:** Método ofrecido por el componente *traceGeneration* que guarda un escenario de simulación configurado por el usuario.
- ***loadEnvironmentsFromFolder*:** Método ofrecido por el componente *traceGeneration* que carga los escenarios de movilidad almacenados en la carpeta definida a tal efecto.
- ***loadEnvironmentDefinition*:** Método ofrecido por el componente *traceGeneration* que recupera los datos de un escenario de simulación almacenado en un directorio que recibirá como parámetro.
- ***executeSimulation:*** Método ofrecido por el componente *traceGeneration* que ejecuta la simulación de un escenario seleccionado.

4.1.1.3 Componente *tclGeneratorGUI*

Este componente muestra la interfaz para la generación de experimentos de simulación para NS-2. Las clases que lo forman están relacionadas con cada una de las fuentes de trazas de movilidad definidas en el sistema. En esta ventana el usuario podrá configurar los distintos aspectos de la simulación de red además de definir las aplicaciones que se simularán, su ratio de penetración, si se utilizarán RSU, definir código a incluir directamente en los ficheros de simulación de red, definir las opciones para la generación de trazas resultantes de la simulación, etc.

La Figura 11 muestra el diagrama de clases del componente.

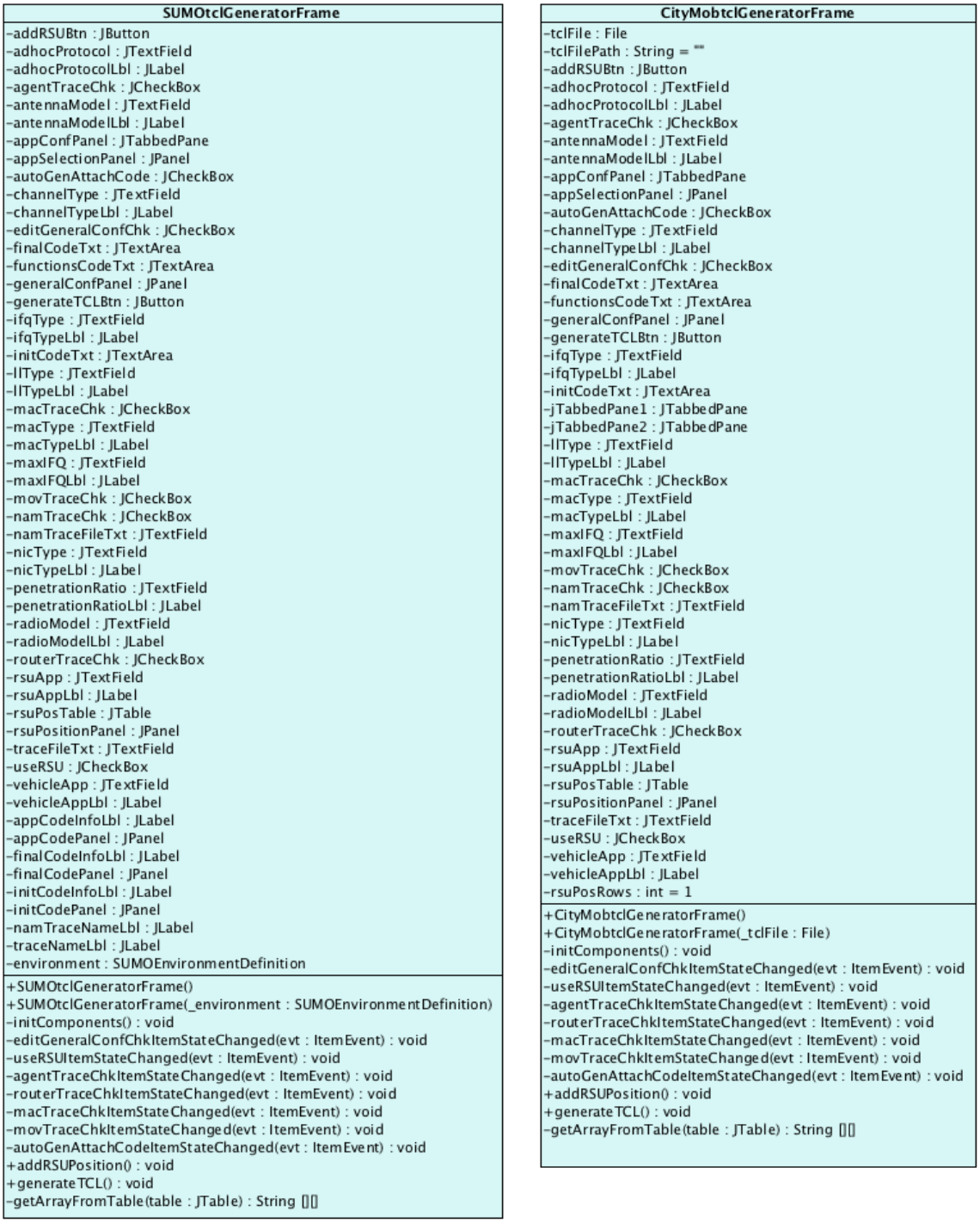

Figura 11 - Diagrama de clases del componente *tclGenerationGUI*

Como se ha mencionado previamente el componente está formado por una clase para la generación de experimentos a partir de la movilidad generada por distintas fuentes: SUMO, *SUMOtclGeneratorFrame*, y CityMob, *CityMobtclGeneratorFrame*. La interacción de este componente con los demás componentes del sistema se realiza consumiendo el método *write* del componente *netwoSimGeneration* que creará y escribirá en un fichero las simulaciones de red para NS-2 con las opciones indicadas por el usuario.

4.1.2 Componente traceGeneration

Este componente es el encargado de ofrecer la funcionalidad específica para la generación de trazas de movilidad utilizando las distintas fuentes definidas. Las clases que lo forman ejecutan las operaciones de generación de trazas que se hayan definido en las ventanas mostradas por el componente *traceGenerationGUI*.

La Figura 12 muestra el diagrama de clases del componente.

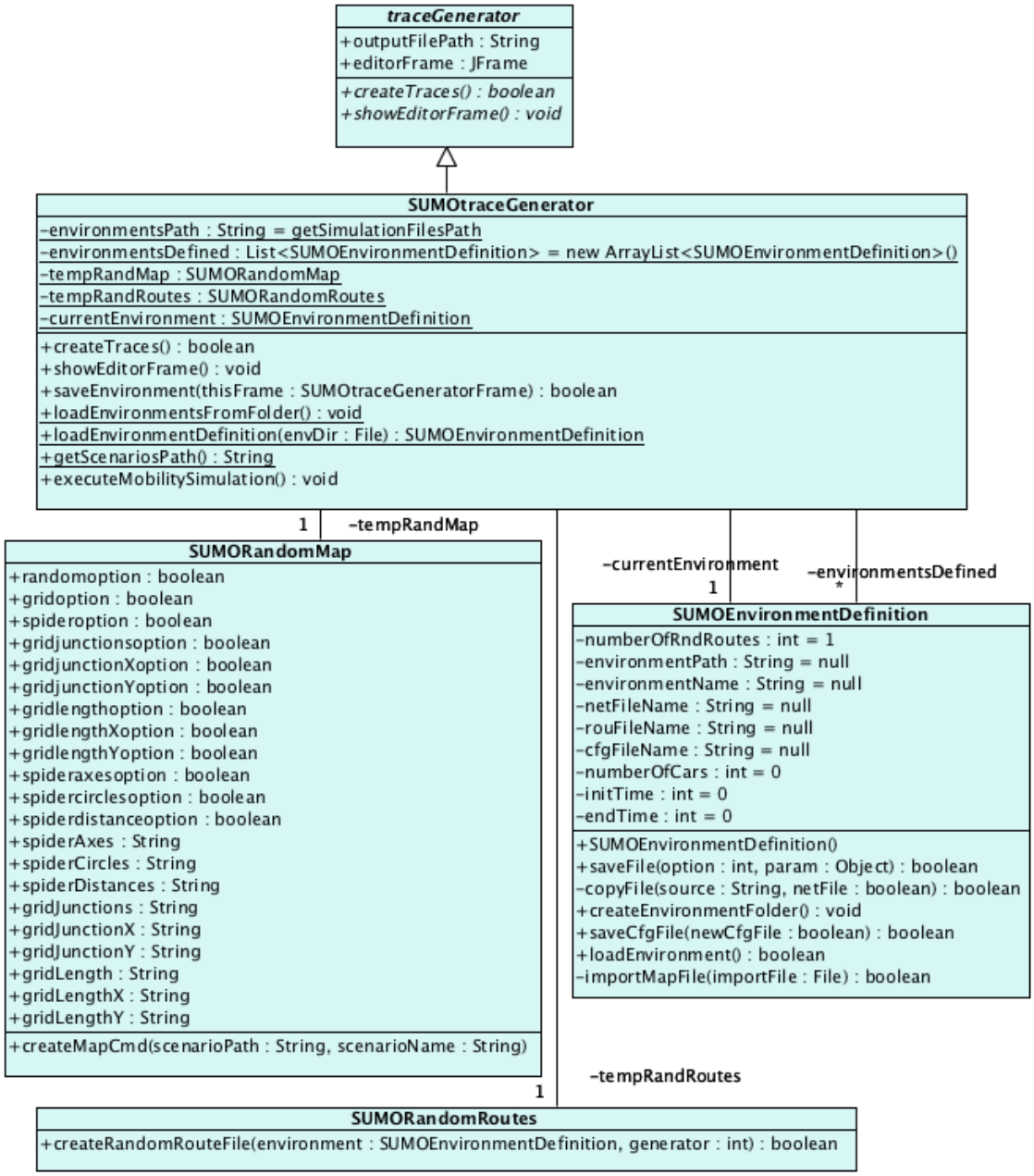

Figura 12 - Diagrama de clases del componente *traceGeneration*

El diagrama de la Figura 12 muestra cómo se ha intentado incrementar la funcionalidad del sistema. Se puede ver que en lo alto de esta jerarquía de clases hay una clase abstracta *TraceGenerator*, esta clase se corresponde con la "Estrategia" del patrón Strategy [64], siendo *SUMOTraceGenerator* la "Estrategia concreta", al hacerse uso de este patrón

solo para una estrategia concreta no se llega apreciar su utilidad, pero para cuando se quiera añadir una nueva fuente de trazas de movilidad este patrón facilitará las cosas.

También se puede ver en el diagrama, a pesar de que la aplicación se apoya en dos programas distintos para generar trazas de movilidad sólo aparecen clases relacionadas con SUMO, esto se debe a que la funcionalidad de generación de trazas se hace invocando directamente a CityMob y no ha habido necesidad de generar clases adicionales a este componente para integrar CityMob con la aplicación.

La versión de CityMob utilizada es una versión modificada de la ofrecida por sus desarrolladores. Los cambios se centran en ofrecer la posibilidad de generar varias trazas de movilidad a partir de una configuración predefinida de forma que los experimentos generados con CityMob también puedan contar con más de una simulación de red, cada una con una movilidad distinta, aunque a partir de la misma configuración.

Respecto a las clases de generación de trazas de SUMO lo primero que cabe destacar es que el diseño está pensado para poder aceptar nuevas fuentes de trazas de movilidad, siempre que su punto de unión con la aplicación sea a través de la clase abstracta *traceGenerator* de la que deberán descender.

La clase *SUMOtraceGenerator* extiende la funcionalidad de la clase abstracta *traceGenerator* y está relacionada con las distintas clases definidas para la generación de trazas con SUMO: *SUMORandomRoutes*, *SUMORandomMap* y *SUMOEnvironmentDefinition.*

El componente ofrece los siguientes métodos para que sean consumidos por los demás componentes de la aplicación:

- ***saveEnvironment*:** Método que guarda un escenario de simulación configurado por el usuario.
- ***loadEnvironmentsFromFolder*:** Método que carga los escenarios de movilidad almacenados en la carpeta definida a tal efecto.
- ***loadEnvironmentDefinition*:** Método que recupera los datos de un escenario de simulación almacenado en un directorio que recibirá como parámetro.
- ***executeSimulation:*** Método que ejecuta la simulación de un escenario seleccionado.

A su vez el componente consume los siguiente métodos ofrecidos por distintos componentes del sistema:

- ***getDirList*:** Método ofrecido por el componente *IOUtils* que obtiene el listado de directorios contenidos en un directorio especificado.
- ***getSUMODirList*:** Método ofrecido por el componente *IOUtils* que obtiene el listado de directorios contenidos en un directorio especificado y que contienen un escenario de simulación de SUMO.
- ***getStrCanonicalPath*:** Método ofrecido por el componente IOUtils que obtiene la ruta canónica de una ruta que recibe como parámetro ya sea absoluta o relativa.
- ***loadXml*:** Método ofrecido por el componente XmlUtils que obtiene el contenido de un fichero XML cuya ruta recibe como parámetro y lo carga en memoria.
- ***saveXml*:** Método ofrecido por el componente XmlUtils que guarda el contenido de un fichero XML almacenado en memoria en un fichero cuya ruta absoluta recibe como parámetro.

- **e*xecuteCmd***: Método ofrecido por el componente *AppInterface* que ejecuta en línea de comandos un comando recibido como parámetro.

En caso de que se desee modificar la aplicación para aceptar nuevas fuentes de movilidad en la generación de experimentos las nuevas clases con la funcionalidad principal deberán de añadirse a este componente, partiendo de la clase abstracta *traceGenerator* como se ha indicado antes.

4.1.3 Componente networkSimGeneration

La funcionalidad de este componente consiste en la generación de trazas de movilidad a partir de distintas fuentes. Las clases que lo forman se encargarán de la generación de los experimentos para NS-2.

El código de estas clases ha sido obtenido del código fuente de MOVE, la clase principal *tclGenerator* ha sido intensamente modificada para adaptarla a su uso con el sistema y para incluir opciones en la generación de simulaciones de red que no proporciona MOVE como por ejemplo la simulación de aplicaciones específicas asignadas a los vehículos, la definición de un rango de penetración de dichas aplicaciones de entre todo el conjunto de vehículos simulados, la inclusión de RSU y la definición de aplicaciones específicas para ellas, entre otras. A lo largo de esta subsección se darán mas detalles de estas modificaciones realizadas al código.

La Figura 13 muestra el diagrama de clases que forman el componente, debido a su gran tamaño se han obviado los métodos y variables carentes de interés, como los métodos de asignación y recuperación de valores (*get/set*).

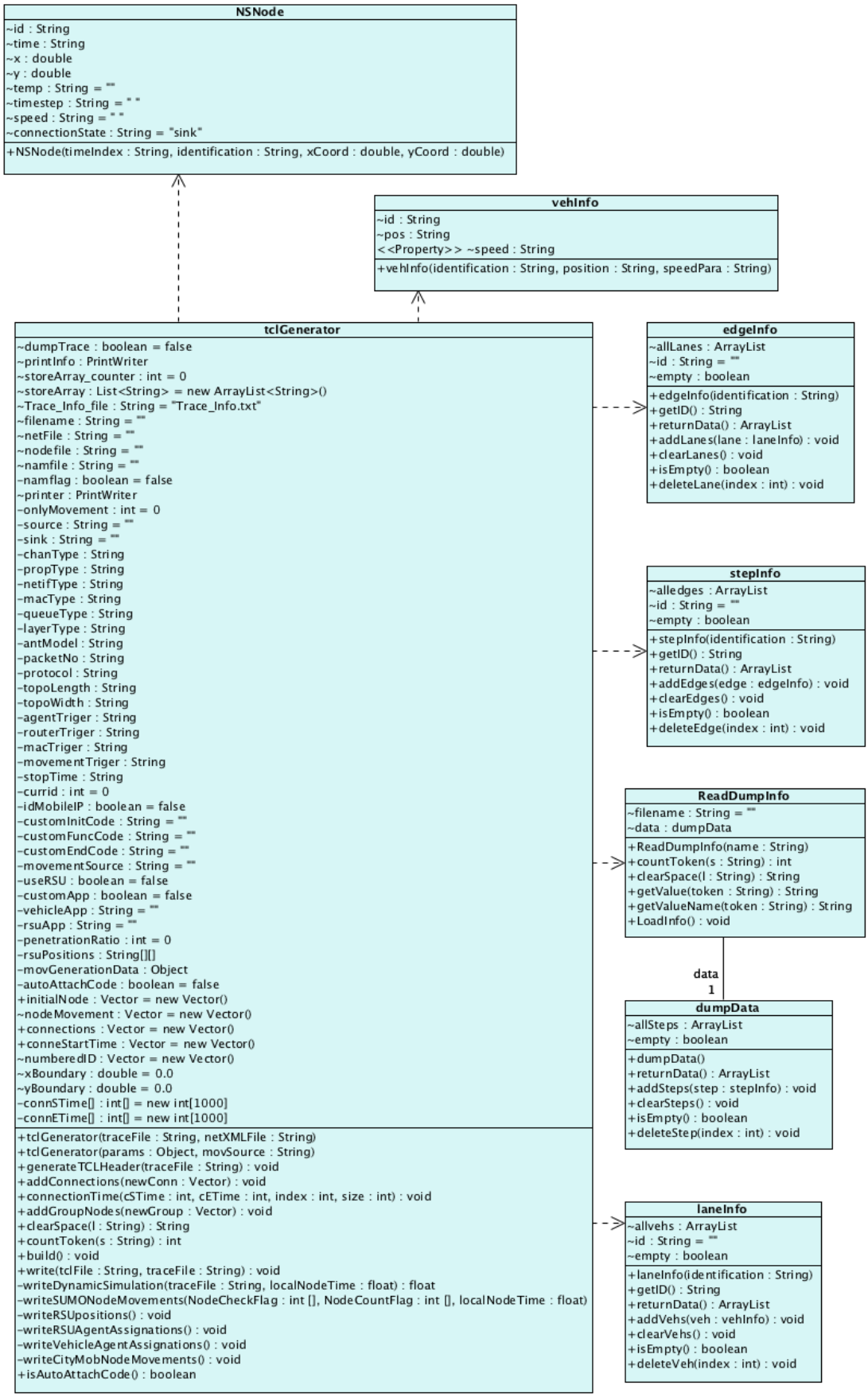

Figura 13 - Diagrama de clases del componente *networkSimGeneration*

En este componente no hay diferenciación entre las clases utilizadas para la generación de experimentos con distintas fuentes de trazas de movilidad. Como se ha explicado anteriormente, la generación de experimentos la lleva a cabo la clase *tclGenerator*, se trata de una clase hecha con código del simulador MOVE que ha sido ampliamente modificada para hacer más rica la generación de experimentos, permitiendo así utilizar distintas fuentes de generación de movilidad, a diferencia de MOVE que sólo acepta trazas generadas con SUMO. También se ha añadido la posibilidad de incluir RSU con sus ubicaciones y uso de una aplicación distinta, la posibilidad de que el usuario añada código propio sin necesidad de editar el fichero resultante una vez generado y la posibilidad de definir un ratio de penetración de la aplicación a simular en un determinado porcentaje de coches. Además, se han separado distintas funcionalidades para permitir una mejor adaptación a nuevas fuentes de trazas de movilidad y nuevas opciones para incluir en la generación de experimentos para NS-2.

El único método destacable que ofrece el componente para ser consumido por otros componentes de la aplicación es, a su vez, uno de los más importantes de todo el sistema: *write* es un método que crea un fichero de simulación de red de NS-2 a partir de las trazas de movilidad indicadas con la configuración que se le especifica.

Para llevar a cabo distintas funciones de lectura y escritura sobre archivos el componente hará uso de los siguientes métodos:

- ***getDirList*:** Método ofrecido por el componente *IOUtils* que obtiene el listado de directorios contenidos en un directorio especificado.
- ***writeFile*:** Método ofrecido por el componente *IOUtils* que escribe una cadena de caracteres que recibe como parámetro al final de un fichero especificado.
- ***readFile*:** Método ofrecido por el componente *IOUtils* que lee una cadena de caracteres de un fichero especificado por parámetro.

En caso de que se desee modificar la aplicación para generar experimentos para un simulador de red además de NS-2 las clases para esta generación deberán de ubicarse en este componente.

4.1.4 Componente *IOUtils*

La funcionalidad de este componente consiste en ofrecer a los demás componentes de la aplicación funcionalidades comunes como la manipulación de ficheros, las búsquedas en directorios con distintos criterios o, como se ha visto al realizar el diseño con más detalle, la carga y guardado de ficheros en formato XML, formato ampliamente utilizado por algunas de las herramientas usadas por el sistema. Es por esto por lo que se ha dividido este componente en dos: *IOUtils* y *XmlUtils*. El primero se encargará de la funcionalidad originalmente planificada para este componente, sin embargo el segundo ofrece operaciones mara trabajar con ficheros en formato XML.

La Figura 14 muestra el componente *utils* incluyendo el nuevo componente creado.

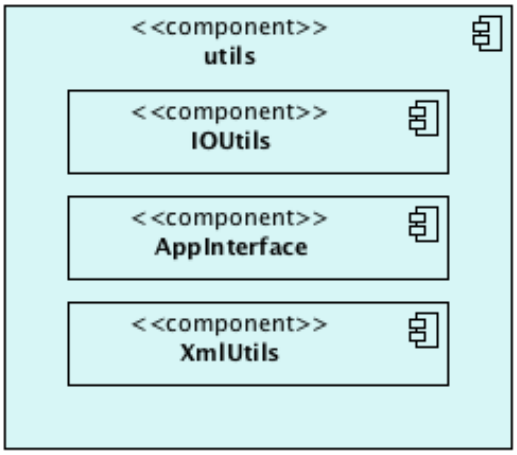

Figura 14 - Subcomponentes del componente *utils*

4.1.4.1 Componente *IOUtils*

Este componente ofrece distintas funcionalidades relacionadas con la manipulación de ficheros y directorios. Debido a que Java nativo dispone de diversas clases que realizan distintas labores de manipulación se han agrupado en esta clase algunas funciones de más alto nivel como son la búsqueda por directorios y patrones, la búsqueda de directorios con escenarios de simulación de SUMO, entre otras.

La Figura 15 muestra la clase que forma el componente.

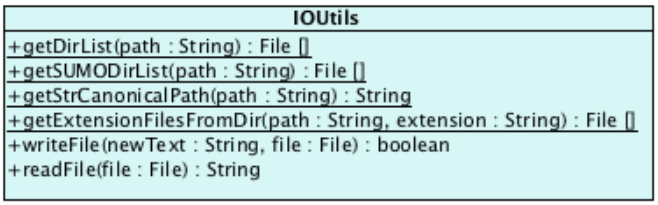

Figura 15 - Diagrama de clases del componente IOUtils

4.1.4.2 Componente *XmlUtils*

La funcionalidad de este componente consiste en la ejecución operaciones de lectura y guardado de ficheros XML. Puesto que Java permite la manipulación a bajo nivel de ficheros XML a través de la librería JDOM [65], las operaciones que ofrece esta clase se resumen a la de cargar en memoria un XML o guardar en un fichero un XML residente en memoria. A medida que se amplíe el sistema con nuevas funcionalidades se irán incorporando a esta clase operaciones comunes en los distintos componentes y lo suficientemente complejas como para refactorizarlas en esta clase..

La Figura 16 muestra el diagrama de clases que forman el componente.

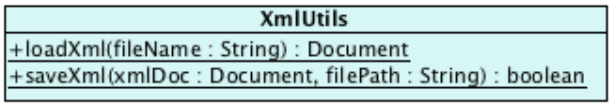

Figura 16 - Diagrama de clases del componente XmlUtils

Los métodos ofrecidos por este componente para que sean consumidos por los distintos componentes del sistema son:

- ***loadXml*:** Método que obtiene el contenido de un fichero XML cuya ruta recibe como parámetro y lo carga en memoria.
- ***saveXml*:** Método que guarda el contenido de un fichero XML almacenado en memoria en un fichero cuya ruta absoluta recibe como parámetro.

4.1.5 Componente AppInterface

Este componente tiene el objetivo de invocar a componentes externos al sistema. En la primera versión del sistema la única función que hará será hacer llamadas por línea de comando, pero si el sistema evoluciona incluyendo nuevas herramientas de simulación este componente se deberá adaptar para incluir las nuevas formas de llamar a dichas herramientas.

La Figura 17 muestra el diagrama de clases que forman el componente.

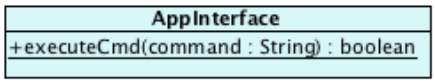

Figura 17 - Diagrama de clases del componente *AppInterface*

La única clase que forma el componente, *AppInterface*, actualmente sólo tiene implementado un método que ejecuta un comando que recibe como parámetro, esto se debe a que la interacción con los componentes externos a los que llamada actualmente la aplicación se puede realizar de esta forma. En caso de añadir nuevos componentes externos a la aplicación que requieran otro tipo de interacción se añadirán a esta clase los métodos necesarios para implementar el nuevo tipo de interacción.

- ***executeCmd***: Método que ejecuta en línea de comandos un comando recibido como parámetro.

4.2 Diagramas de secuencia

Una vez se ha realizado el diseño del software, se pasa a reflejar las distintas interacciones entre el usuario y el sistema, y también entre las clases del propio sistema. Este apartado muestra, a través de diagramas de secuencia, la interacción entre los distintos objetos del sistema en la ejecución de los distintos casos de uso definidos en el apartado 3.6. No se van a representar los diagramas de secuencia de los flujos alternativos de los distintos casos de uso si no se considera necesario mostrándose siempre que sea posible el flujo de ejecución normal.

Se han simplificado todos los diagramas eliminando interacciones con clases del lenguaje como son Integer o Double, así como las invocaciones a métodos que no aportan información ni son de interés, como los métodos get y set. De esta forma se ha reducido considerablemente el tamaño de algunos diagramas y con esto la claridad de los mismos pudiéndose visualizar las interacciones más importantes de cada caso de uso.

4.2.1 Crear trazas de movilidad (CU-01)

La interacción entre los objetos del sistema en este caso de uso dependerá del origen para las trazas que se esté utilizando puesto que SUMO y CityMob operan de distinta forma.

Dentro de los diagramas de secuencia pertenecientes al caso de uso CU-01 se han elegido mostrar los diagramas de secuencia correspondientes a la creación de trazas con CityMob y SUMO.

La Figura 18 muestra el diagrama de secuencia correspondiente a la creación de trazas de movilidad utilizando CityMob.

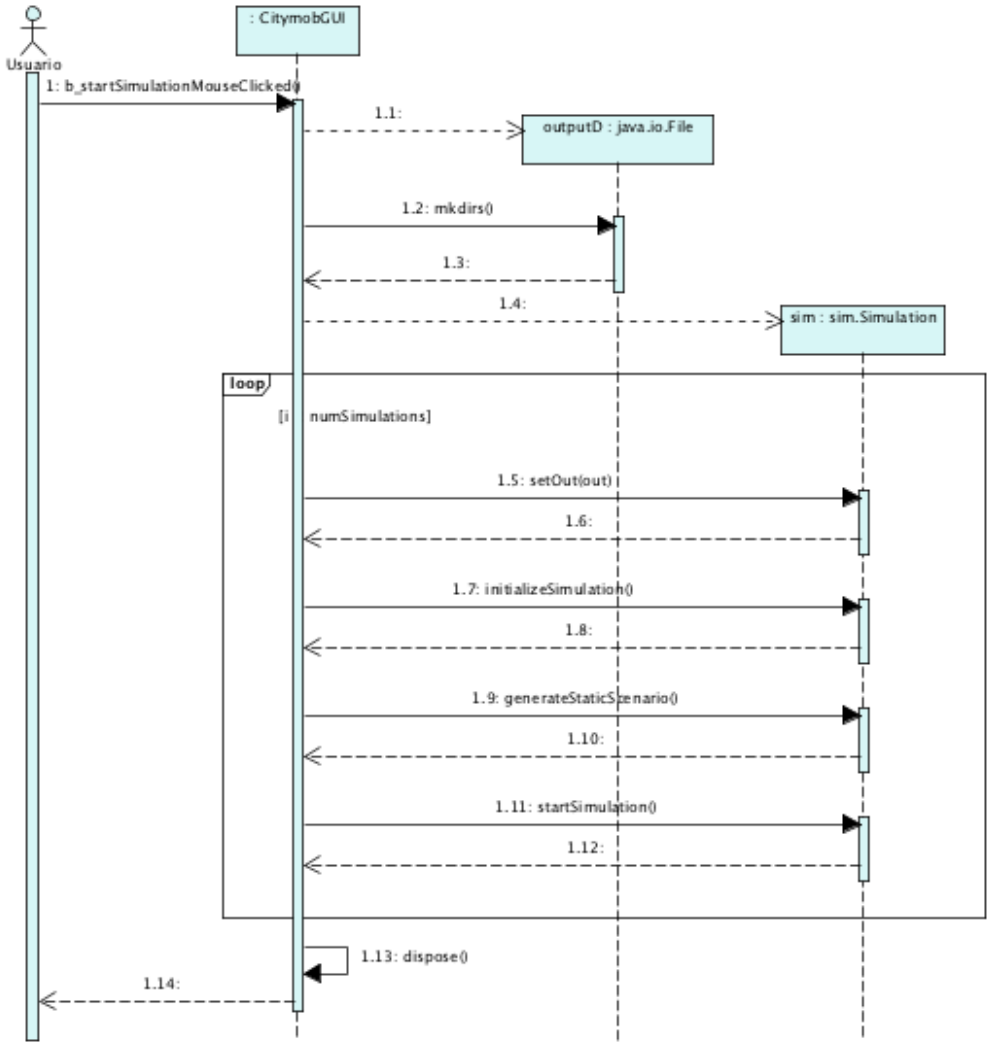

Figura 18 – Diagrama de secuencia de generación de trazas de movilidad con CityMob (CU-01)

La generación de trazas de movilidad con CityMob la realiza la propia aplicación CityMob aunque está incluida en el sistema y su código ha sido modificado para que se adaptara a las necesidades del sistema.

Una vez el usuario pulsa el botón de inicio de generación de trazas de movilidad CityMob comprueba si existe el directorio en el que se van a crear los fichero y de no existir crea un nuevo directorio donde se generarán los ficheros. Tras esto se crea el objeto de Simulación con el que se crean e inicializan los distintos parámetros de la generación de trazas y por último se ejecuta la generación de trazas con el método *startSimulation*. Este proceso de configuración y generación de trazas se repite en un bucle que hará que se generen tantas trazas de movilidad como el usuario haya especificado.

La Figura 19 muestra el diagrama de secuencia correspondiente a la creación de trazas de movilidad utilizando SUMO.

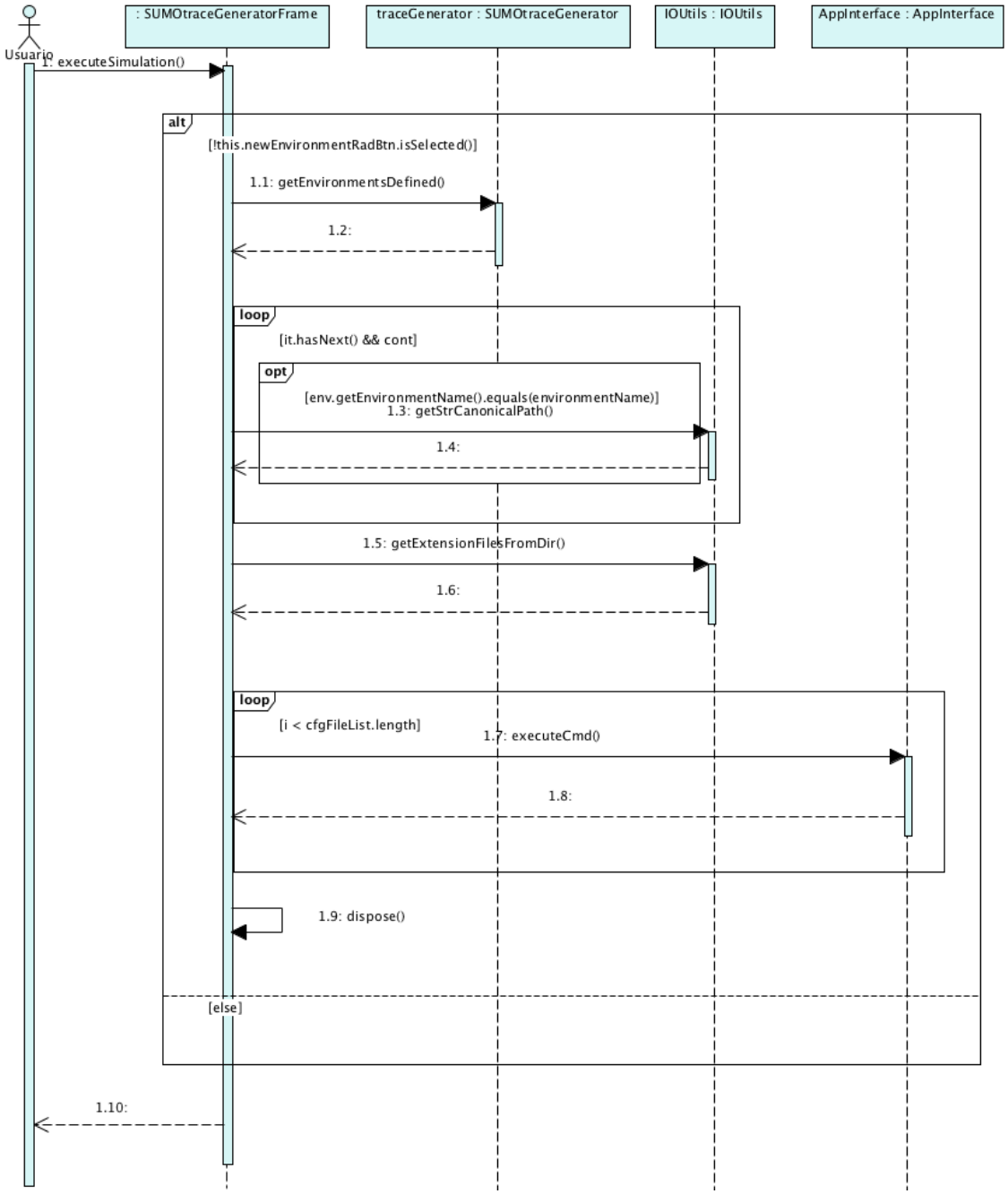

Figura 19 - Diagrama de secuencia de generación de trazas de movilidad con SUMO (CU-01)

El proceso de generación de trazas de movilidad comienza cuando el usuario indica que desea iniciar la generación. La aplicación recupera el escenario de simulación seleccionado iterando por todos los existentes, buscando el seleccionado. Si lo encuentra busca la ruta canónica al directorio del escenario y se recupera un listado con todos los ficheros de ese directorio que corresponden a escenarios de movilidad de SUMO, para esto se invoca el método *getExtensionsFilesFromDir* de la clase *IOUtils*. Una vez se tiene el listado de ficheros correspondientes a escenarios movilidad se recorre ejecutando SUMO en cada iteración con cada uno de los ficheros por medio del método *executeCommand* de la clase *AppInterface.*

4.2.2 Crear simulación de red (CU-02)

Al igual que pasa con el caso de uso CU-01, para la creación de simulaciones de red hay que diferenciar entre las interacciones llevadas a cabo si se toma como origen de las trazas de movilidad de una u otra aplicación. Pero en este caso la interacción para ambos casos es similar puesto que ambos generan simulaciones de NS-2 por lo que el proceso es común a este nivel en ambos casos. Es por esto por lo que sólo se incluirá el diagrama de secuencia para la situación en la que se generan experimentos de simulación utilizando trazas de movilidad de SUMO.

La Figura 20 muestra el diagrama de secuencia correspondiente a la creación de simulaciones de red a partir de trazas de movilidad generadas por SUMO.

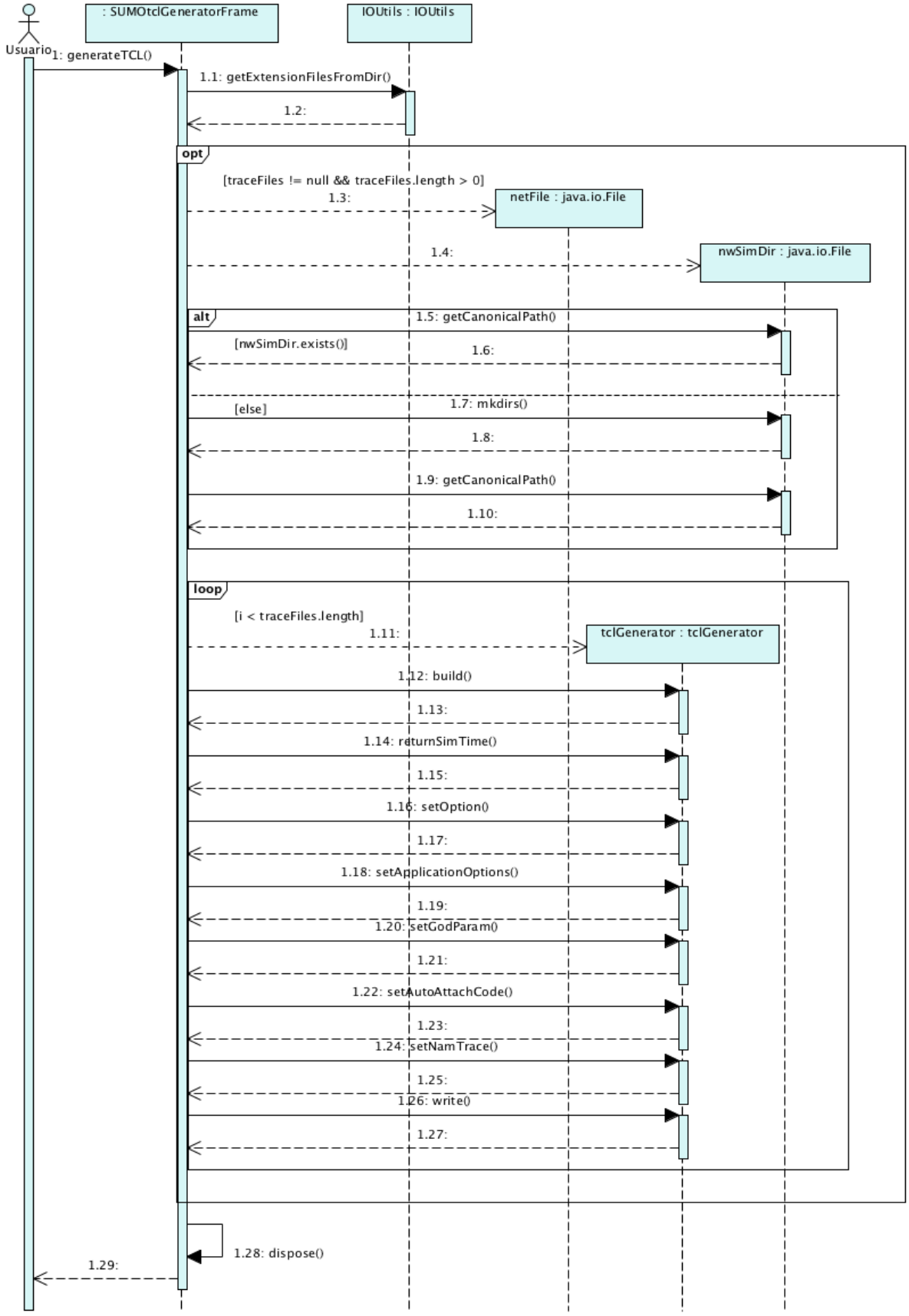

Figura 20 - Diagrama de secuencia de creación de simulaciones de red con trazas de SUMO (CU-02)

El proceso de creación de simulaciones de red con trazas de movilidad de tanto de SUMO como de CityMob empieza cuando el usuario termina de configurar la simulación en la interfaz correspondiente y pulsa el botón que da comiendo a la generación de los ficheros *tcl*. En primer lugar se buscan, en las carpetas adecuadas, todos los ficheros con el formato de las trazas dependiendo de su origen, utilizando el método *getExtensionFilesFromDir* del componente *utils*. En caso de existir trazas válidas para la generación de simulaciones de red se crea, en caso de no existir, el directorio donde se almacenarán las simulaciones de red y sus resultados si se ejecutan. Tras esto se entra en un bucle que recorre todos los ficheros de trazas localizados en los primeros pasos y para cada uno de ellos se configura una generación de una simulación de red y se ejecuta dicha generación invocando al método *write* del componente *networkSimGeneration*, dando como resultado un fichero *tcl* con la simulación de red. En caso de estar ante un experimento con varias simulaciones de red el bucle continuará repitiendo el mismo paso generando tantas simulaciones de red como trazas de movilidad hay en el directorio seleccionado.

4.2.3 Ejecutar simulación (CU-03)

A diferencia de lo visto para el caso de uso CU-01, el diagrama de secuencia del caso de uso CU-03 es independiente del origen de las trazas de movilidad que forman la simulación puesto que únicamente refleja el flujo seguido durante la ejecución de la simulación, en donde no debería de haber diferencias entre las simulaciones creadas con distintas herramientas.

La Figura 21 muestra el diagrama de secuencia correspondiente a la ejecución de un experimento de simulación.

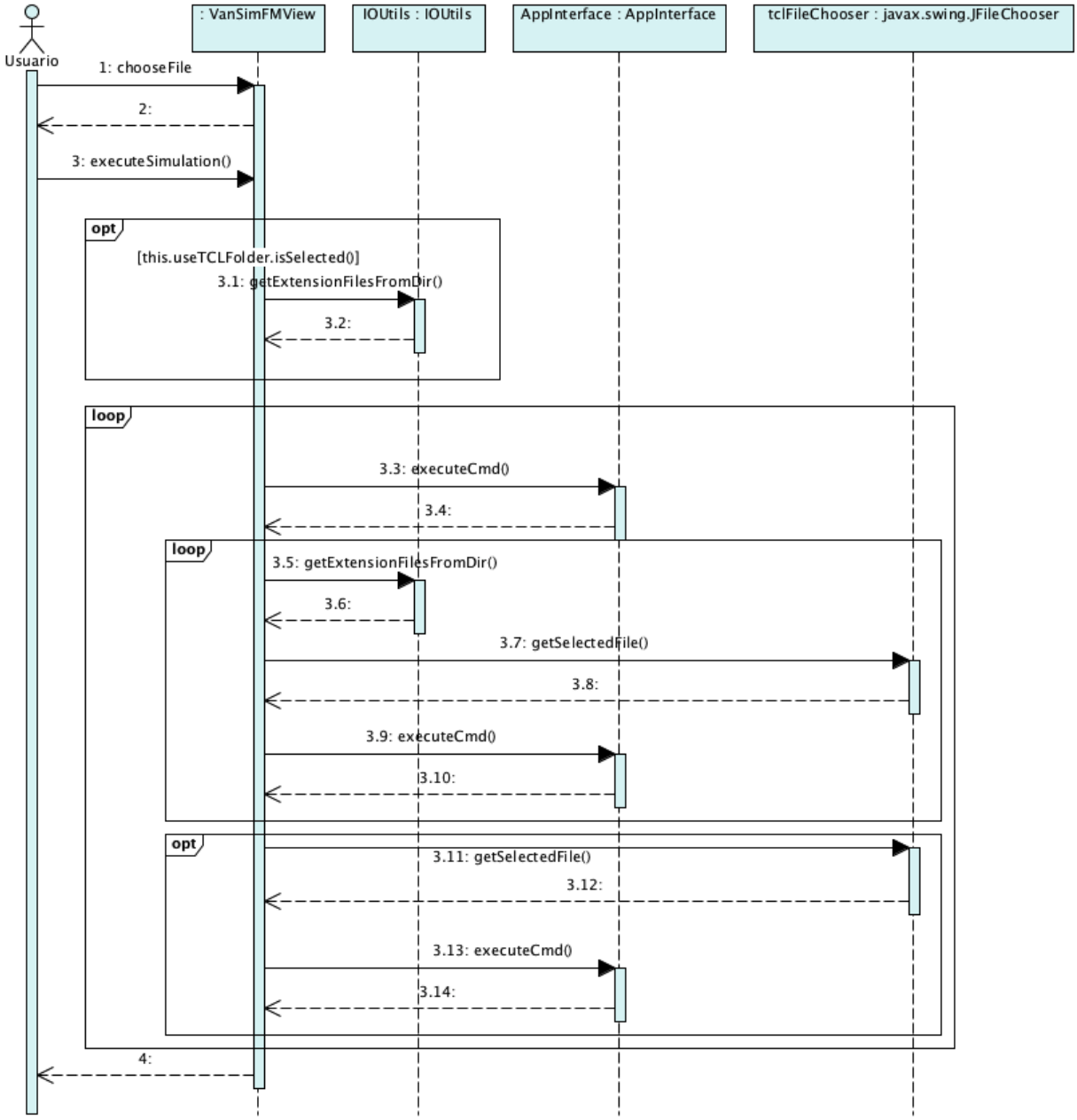

Figura 21 - Diagrama de secuencia del caso de uso CU-03

El proceso de ejecución de experimentos de simulación se inicia con el usuario seleccionando un fichero de simulación de red y marcando las opciones de la ejecución que desee. Si se indica que se quieren ejecutar todas las simulaciones contenidas en la misma carpeta que la simulación elegida, se recuperarán todas las simulaciones de la carpeta mediante el método *getExtensionFilesFromDir* del componente *IOUtils*. Tras esto se iterará sobre el listado de ficheros de simulación de red, que puede ser de longitud uno si sólo se desea ejecutar el fichero seleccionado. En cada iteración se ejecutará la simulación de red invocando NS-2 mediante línea de comandos, a través del método *executeCmd* del componente *AppInterface*.

Una vez se ha ejecutado la simulación si el usuario ha indicado que deseaba ejecutar scripts para procesar los datos resultantes de la simulación, se recuperarán todos los ficheros de script presentes en la carpeta de la simulación de red tras lo cual se iterará sobre

los encontrados ejecutándolos de nuevo a través del método *executeCmd*. Tras finalizar la ejecución de los scripts de procesado de datos se mostrará el visualizador de trazas Nam en caso de que el usuario lo haya indicado.

Capítulo 5

Implementación del software

5.1 Decisiones de implementación

En esta sección se detallan las decisiones que se han tomado con respecto a la implementación de la aplicación durante el desarrollo de la misma tras haber realizado el análisis y diseño. En particular, se muestran las decisiones más relevantes tomadas sobre el uso de los componentes externos detallados en capítulos anteriores, más concretamente CityMob, SUMO y NS-2.

5.1.1 Modificaciones al código de CityMob

Para integrar CityMob en el sistema y permitir a sus usuarios generar más de una traza de movilidad por cada ejecución se han llevado a cabo varias modificaciones en su código fuente. Para esto se ha tenido que modificar, además del código relativo a la funcionalidad de la aplicación, el código relacionado con la interfaz para añadir la opción de generar más de una traza de movilidad cada vez que el usuario ejecute la generación de trazas con esta configuración.

Los cambios realizados en el código para cambiar la funcionalidad de la aplicación se han centrado en reestructurarlopara que la generación de trazas de movilidad se ejecute en un bucle *for*. Dicha generación se hace a partir de la configuración especificada por los usuarios y variables aleatorias por lo que dos generaciones con los mismos parámetros dan como resultados trazas de movilidad diferentes. También se ha modificado la funcionalidad de la aplicación para que guarde los ficheros con las trazas generadas de acuerdo a la estructura que utilizará el sistema.

5.1.2 Uso de la interfaz de CityMob

En un principio se consideró implementar una interfaz de usuario específica para introducir los parámetros necesarios para invocar métodos de Citymob, modificando en lo que fuera necesario sus métodos. Esta opción se acabó por descartar a favor de utilizar la propia interfaz de CityMob modificándola para adaptarla a las nuevas funcionalidades que se iban a incluir ya que de esta forma se evitaba el volver a implementar algo que estaba creado y probado, reduciendo también las pruebas y validaciones a realizar que de haber implementado una nueva interfaz serían bastantes.

Las modificaciones realizadas sobre la interfaz gráfica han ido orientadas a permitir el uso de las modificaciones realizadas en la funcionalidad de la aplicación. La Figura 22 muestra la interfaz de usuario original de CityMob mientras que la Figura 23 muestra la interfaz modificada.

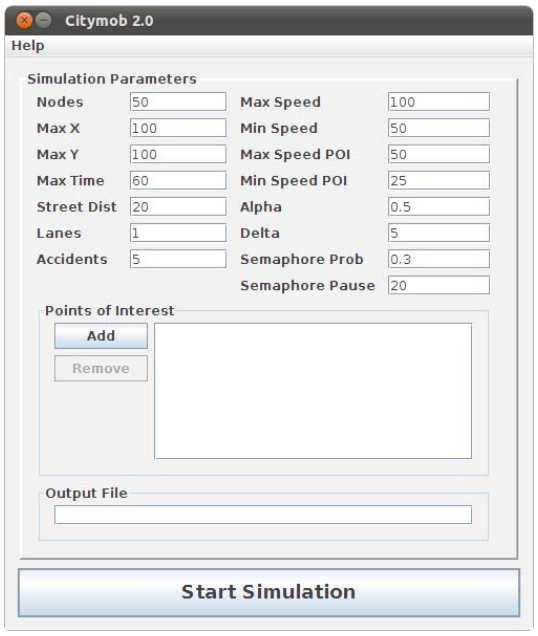

Figura 22 - Interfaz original de CityMob

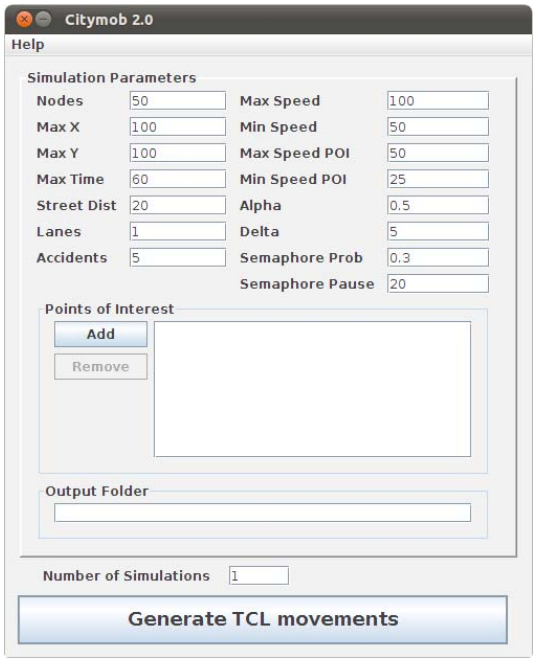

Figura 23 - Interfaz modificada de CityMob

Como se puede apreciar en la Figura 23, se ha incluido en la interfaz de usuario el campo *Number of Simulations* en el que el usuario deberá indicar el número de trazas de simulación que desea generar. Para que el texto de la interfaz fuera coherente con la funcionalidad de esta versión modificada, también se ha cambiado el texto del botón para iniciar la generación de trazas pasando dicho texto a ser *Generate TCL movements* dejando más clara así la funcionalidad del mismo.

5.1.3 Modificaciones al código de MOVE utilizado

Como se ha indicado en el apartado 4.1.3, la generación final de los ficheros de simulación para NS-2 se realizará utilizando como base el código de MOVE destinado a esta función. Pero para que dicho código se adapte a las funcionalidades del sistema se han debido realizar diversas modificaciones sobre el código de las clases de MOVE utilizadas. La primera de ellas ha tenido su foco en la inclusión de las funcionalidades del sistema no contempladas por MOVE como son el uso de RSU, el uso de trazas de movilidad generadas con CityMob, la posibilidad de definir un porcentaje de vehículos que participarán en la simulación de red o la posibilidad de incluir bloques de código definidos por el usuario desde la aplicación.

Adicionalmente a estas modificaciones se han decidido llevar a cabo diversas tareas de refactorización sobre el código de la clase *tclGenerator* para agrupar la generación de las distintas partes de un fichero de simulación de red en métodos separados de forma que se puedan agrupar y combinar en nuevos métodos facilitando así la inclusión de nuevas funcionalidades o la modificación de las ya existentes.

5.1.4 Visualización de simulaciones de movilidad de SUMO

Dentro de las opciones ofrecidas al usuario al generar trazas de movilidad con SUMO se encuentra la posibilidad de visualizar las simulaciones de movilidad que dan lugar a las trazas de movilidad utilizadas para la creación de simulaciones de red. En el caso de que el usuario haya seleccionado la creación de diversas trazas de movilidad a partir de un único escenario, se ha decidido que el sistema sólo muestre la visualización de la primera simulación de la serie a realizar. Esta decisión de mostrar únicamente la visualización de la primera simulación de movilidad ha venido motivada por el hecho de que el funcionamiento de SUMO hace que se deba visualizar toda la simulación para que se generen todos los movimientos en las trazas de movilidad de la simulación.

En el caso de hacer que se visualizaran todas las simulaciones, el usuario debería visualizarlas todas para que las trazas de movilidad se generaran correctamente. Puesto que las simulaciones son de un mismo escenario de movilidad se ha considerado que lo que aportaría ver todas las simulaciones no compensaba el tiempo que podría exigir o los problemas que podrían venir derivados de que el usuario interrumpiera las visualizaciones antes de que estas terminaran. Por tanto se ha decidido que, en caso de indicarse que se quiere visualizar la simulación de movilidad, se muestre la primera simulación de movilidad de la serie de simulaciones a ejecutar y el resto se ejecute sin la interfaz gráfica de SUMO.

De esta forma se proporciona al usuario una panorámica de lo que ocurrirá con las demás simulaciones sin los problemas antes mencionados.

5.1.5 Manipulación de ficheros XML

Inicialmente se pensó agrupar toda la funcionalidad de manipulación de ficheros XML en la clase *XmlUtils* (ver Sección 4.1.4). Con esto se buscaba que se invocaran sus funcionalidades desde cualquier clase del sistema, pero durante la implementación del mismo se detectó que a través de los métodos ofrecidos por la librería JDOM la manipu-

lación de ficheros XML a través de las funciones más comunes era directa. Por lo tanto utilizar estas funcionalidades a través de la clase *XmlUtils* no hacía sino complicar el manejo de ficheros XML. Es por esto por lo que se ha decidido dejar en la clase *XmlUtils* únicamente la funcionalidad de carga y guardado de ficheros, dejando las funcionalidades de manipulación de nodos a cargo de los métodos ofrecidos por la librería JDOM.

La decisión de mantener la clase *XmlUtils* se ha tomado teniendo en cuenta que en futuras extensiones de la aplicación con, por ejemplo, nuevas fuentes de trazas de movilidad, se podrían detectar operaciones complejas sobre ficheros XML que podrían ser implementadas en *XmlUtils* y ser invocadas desde ahí.

5.1.6 Espera a la finalización de procesos externos

La llamada a componentes externos al sistema como SUMO o NS-2 a través de la línea de comandos se ha implementado de forma que el sistema quede a la espera de la finalización de la llamada antes de continuar con su ejecución. De esta forma el usuario deberá esperar a la finalización de la ejecución de estas llamadas antes de poder continuar con su flujo de trabajo. Esta decisión acarrea el inconveniente de forzar la espera del usuario a que termine el proceso externo, pero evita el problema de que el usuario continúe con su flujo de trabajo ejecutando llamadas a componentes externos que requieren de los resultados de llamadas previas, que pueden no haber terminado aún. Además, se evita que se realicen más llamadas a componentes externos, lo que puede llegar a provocar situaciones indeseables como un incremento en el consumo de recursos o problemas derivados del acceso concurrente a ficheros de estos componentes externos.

Se ha logrado la espera del sistema mientras se ejecuta el comando de línea de comandos utilizando el método *waitFor* de la clase *Process* de Java, que pausa el hilo desde el que se ejecuta hasta que el subproceso especificado, en este caso la llamada a uno de los componentes externos que se han mencionado antes, haya finalizado.

5.2 Resultados de las pruebas de aceptación

En esta sección se muestran los resultados de las pruebas de aceptación definidas en la sección 3.8. Como se puede apreciar en la Tabla 9 se han superado todas las pruebas de aceptación que se habían definido.

Pruebas de aceptación		
Id	**Elementos probados**	**Resultado**
PA-01	RF-01, RF-03	Superada
PA-02	RF-01, RF-04	Superada
PA-03	RF-01, RF-03	Superada
PA-04	RF-01, RF-04	Superada
PA-05	RF-02	Superada
PA-06	RF-02	Superada
PA-07	RF-06, RF-09, RF-10, RF-11, RF-13	Superada
PA-08	RF-06, RF-08, RF-13	Superada
PA-09	RF-06, RF-07, RF-13	Superada
PA-10	RF-12	Superada
PA-10	RF-03	Superada
PA-11	RF-05	Superada
PA-13	RF-04	Superada
PA-14	RF-01, RF-14	Superada
PA15	RF-01, RF-15	Superada
PA-16	RF-02, RF-17	Superada
PA-17	RF-02, RF-16	Superada
PA-18	RF-18	Superada

Tabla 9 - Resultados de las pruebas de aceptación

Capítulo 6

Uso de VanSimFM para la generación de experimentos

6.1 Banco de pruebas para simulaciones VA-NET

Tener un marco común para la prueba y validación de propuestas VANET es crítico para confirmar la compatibilidad entre distintas propuestas. El uso de experimentos de simulación con grandes diferencias entre sí puede hacer imposible la comparación de distintas propuestas, por ejemplo, no es lo mismo validar un protocolo exigente en términos de ancho de banda con un experimento situado en el centro de una ciudad con una alta densidad de vehículos que un experimento situado en una carretera rural con escaso tráfico. La validación el último podría parecer, a priori mejor, pero en realidad puede ser al contrario al ejecutar las propuestas en un entorno real.

Habiéndose planteado como objetivo la creación de una herramienta que permitiera crear experimentos fácilmente y conociendo las ventajas que un banco de pruebas común podría ofrecer se decidió añadir como nuevo objetivo de este proyecto la creación de un banco de pruebas que tuviera en consideración en sus configuraciones las prácticas de simulación más extendidas., poniendo así a disposición de la comunidad investigadora un banco de pruebas al que acceder fácilmente para conseguir experimentos que estén ya pre configurados en los que se pueda editar manualmente si se desea cualquier parte de la simulación o también utilizarlos directamente indicando únicamente algunos parámetros básicos.

6.2 Características del banco de pruebas

En esta sección es muestran las principales características de los experimentos que forman el banco de pruebas creado con VanSimFM así como la motivación que hay detrás de cada decisión.

De acuerdo al segundo objetivo establecido para este proyecto se ha creado un banco de pruebas compuesto por varios experimentos de simulación para VANET. Cada uno de estos experimentos cuenta con 5 simulaciones de red para obtener conclusiones más fiables y evitar resultados influenciados por un movimiento de vehículos concreto.

Para construir este banco de pruebas se han tomado como referencia las prácticas de simulación estudiadas en el Capítulo 2. A pesar de la gran heterogeneidad vista en el análisis de las publicaciones recientes se han podido identificar determinados comportamientos más comunes, mapas más utilizados, etc. Con esta información es con la que se ha elaborado el banco de pruebas que aparece detallado en la Tabla 10.

	Número de nodos	Tiempo de simulación	Tamaño
Urbano (Manhattan)	50, 250, 750	100,600,1000	5x5 km
Urbano (Eichstätt, Alemania)	50, 250, 750	100,600,1000	4,5x3 km
Autopista	50, 250, 750	100,600,1000	4,5x5,5 km
Cruce de autopistas	50, 250, 750	100,600,1000	3,8x3 km
Carreteras secundarias	50, 250, 750	100,600,1000	12,5x9,5 km

Tabla 10 - Características de los experimentos de simulación del banco de pruebas

A continuación se explicarán las decisiones toman cada uno de los parámetros analizados das para la creación de los experimentos que figuran en la Tabla 10.

Escenarios de simulación (tipo y tamaño). El banco de pruebas hace uso de cinco tipos distintos de mapas: un mapa urbano con forma de rejilla, un mapa urbano del centro de la ciudad alemana de Eichstätt (obtenido en [66]), un tramo recto de autopista de 3 Km de largo, un cruce de autopistas a distinto nivel con 3 carriles para cada sentido y, por último, un mapa de carreteras secundarias. Con estos mapas no solo se replica la variedad de mapas detectadas en el estudio si no que se amplía con nuevos mapas que pueden contribuir a una validación más completa de las propuestas como por ejemplo el cruce entre autopistas, que no se ha visto en ninguna publicación y es un escenario cada vez más común en las grandes ciudades. Respecto al tamaño de los mapas se ha optado por tamaños en media más grandes que los utilizados normalmente ya que el uso de mapas más grandes permite reproducir más situaciones reales de movilidad como atascos, tendencias de movimientos, etc.

Número de nodos. El número de nodos para cada experimento de simulación será variable, partiendo de un mínimo de 50 nodos a un máximo de 750. Debido a la gran diferencia entre ambos valores se decidió añadir un valor intermedio, 250 nodos. Para obtener estos valores se han descartado valores extremos presentes en algunas de las publicaciones analizadas como por ejemplo 1 y 916. El valor intermedio para el número de nodos de este banco de pruebas se ha calculado a partir del valor medio del número de nodos utilizados en las publicaciones analizadas. Es por esto por lo que se espera que estos

tres valores para los números de nodos den lugar a simulaciones con baja, media y alta densidades de tráfico.

Tiempo de simulación. Se han definido tres valores 100, 600 y 1000 segundos de acuerdo a los valores máximo, mínimo y medio de las publicaciones estudiadas. Utilizar tiempos de simulación más elevados podría arrojar resultados de más valor en las simulaciones pero alargaría la fase final de simulación dramáticamente en algunos casos.

El banco de pruebas descrito en esta sección es una versión inicial que aspira a ser ampliado con contribuciones de personal investigador de todo el mundo.

Capítulo 7

Conclusiones y líneas futuras

7.1 Conclusiones sobre el proyecto

En esta sección se muestran las conclusiones obtenidas una vez finalizado este proyecto fin de carrera analizando las principales dificultades encontradas durante su desarrollo junto con los resultados obtenidos.

7.1.1 Resultados obtenidos

Se ha construido una herramienta que permite generar experimentos de simulación para VANET sin tener un conocimiento profundo de la generación de trazas de movilidad por medio de simuladores de movilidad o generadores de trazas de movilidad. Esto cumple con los objetivos marcados al comienzo del proyecto y aporta una posible solución a la falta de criterios de simulación comunes detectada en la investigación llevada a cabo al comienzo del proyecto. La herramienta creada, VanSimFM, integra un simulador de movilidad, SUMO, y un generador de trazas de movilidad, CityMob, para crear las trazas de movilidad que se utilizarán posteriormente para generar simulaciones de red. VanSimFM permite también la edición de diferentes características de la simulación de red y su posterior ejecución.

VanSimFM ofrece la posibilidad de generar experimentos de simulación de forma sencilla sin tener un conocimiento específico de simuladores de red o generadores de trazas de movilidad, a diferencia de herramientas similares como MOVE o TraNS en las que la generación de trazas de movilidad requiere un conocimiento detallado del simulador de movilidad utilizado, SUMO. VanSimFM también permite editar distintos parámetros de la configuración de la simulación de red, pero a diferencia de MOVE y TraNS permite definir la aplicación que se simulará, el uso de RSU junto con sus posiciones ó bloques de código que se desean incluir directamente en los ficheros de simulación de red, para definir estas opciones en MOVE o TraNS es necesario editar los ficheros de simulación de red uno a uno.

Adicionalmente se ha utilizado VanSimFM para generar un conjunto de experimentos de simulación que reúnan las características más comunes observadas en el estudio de las prácticas actuales de simulación detallado en la sección 2.2.

Tanto VanSimFM como el catálogo de experimentos son de libre distribución, con el objetivo final de que sean extendidos por otras personas haciendo estas herramientas más ricas y completas. Todo esto hace que el trabajo realizado sea especialmente reconfortante al saber que se ha abierto una línea de trabajo que puede ser seguida por otras personas gratuitamente y puede llegar a contribuir a mitigar los problemas sobre los que se ha querido llamar la atención en este proyecto fin de carrera.

Como resultado de este proyecto fin de carrera se ha elaborado un artículo [67] enviado para su consideración[1] a la novena edición de la conferencia ESCAR [68]. La elaboración de dicho artículo ha requerido un esfuerzo adicional pero ha supuesto una experiencia muy interesante y gratificante al ver cómo el trabajo que se ha desarrollando puede llegar a convertirse en la base para otros desarrollos y proyectos que mejoren el siste-

[1] En el momento de escribir esta memoria el artículo estaba en proceso de revisión

ma y contribuyan a la solución de los problemas en el ámbito de la simulación VANET detectados en este proyecto.

7.1.2 Dificultades del proyecto

Una de las dificultades más importantes de las afrontadas durante este proyecto vino una vez realizado el análisis de las principales soluciones utilizadas para realizar simulaciones sobre VANET, debido a la escasez de soporte y documentación de un importante número de estas soluciones. A esta escasez de documentación hay que sumar el hecho de que el desarrollo algunas soluciones esté interrumpido desde hace años, como por ejemplo TraNS, lo que las hace incompatibles con versiones más modernas de simuladores como SUMO o NS-2. El problema de la falta de documentación de buena parte de las soluciones utilizadas afectó también al desarrollo del proyecto en etapas posteriores ya que el propio código de MOVE utilizado en este sistema carecía de documentación o comentarios por lo que su refactorización y modificación retrasaron el desarrollo.

Otra dificultad que se tuvo que afrontar es consecuencia del problema que aborda el proyecto: la falta de un criterio común a la hora de realizar simulaciones y validaciones de aplicaciones para VANET. Esta falta de criterio ha obligado a analizar un gran número de publicaciones para poder extraer unas conclusiones válidas que permitieran observar las tendencias que se estaban siguiendo en la comunidad investigadora a la hora de simular. A esta dificultad en el análisis de las publicaciones se añadió la falta de claridad en la definición de las simulaciones en algunas publicaciones lo que obligaba a una lectura sumamente detallada para comprender el criterio de simulación escogido.

Durante la implementación de las partes del sistema que se integran con SUMO se presentaron dos problemas con la definición de los parámetros con los que ejecutar las llamadas a la funcionalidad de SUMO que importa los mapas en formato OSM y al script que genera rutas aleatorias sobre un mapa de SUMO. En ambos casos se pueden definir diversos parámetros que se tuvieron que probar con distintas combinaciones para determinar los parámetros adecuados para invocar estas funcionalidades desde el sistema.

7.1.3 Conclusiones personales

La mayor dificultad afrontada durante del desarrollo de este proyecto de fin de carrera no ha venido dada por el desarrollo del sistema objetivo del mismo sino debido a la dificultad de compaginar el desarrollo de este proyecto con un trabajo a jornada completa especialmente exigente en tiempo, junto con la asistencia a un máster. Esta escasez de tiempo ha afectado seriamente a la planificación del proyecto a su desarrollo general, ya que el hecho de tener que trabajar principalmente en sesiones de pocas horas es mucho menos eficiente que trabajar el mismo número de horas agrupadas en una única sesión. A esto hay que sumar el hecho de que por exigencias puntuales del trabajo o del máster ha habido épocas en las que no se ha podido dedicar el número necesario de horas al proyecto, teniendo que recuperarlas sacrificando un gran número de horas de sueño. Estas circunstancias han supuesto sin duda alguna la mayor dificultad que se ha afrontado durante el desarrollo de este proyecto fin de carrera, aunque no las únicas.

A pesar de haber tenido un desarrollo más complejo de lo inicialmente esperado, principalmente por problemas de tiempo mencionados anteriormente, la realización de

este proyecto ha aportado una importante y valiosa experiencia. Esta experiencia se debe a haber tenido que saber reaccionar ante las circunstancias impuestas tanto por el proyecto como por factores externos, lo que no siempre ha sido fácil.

El proyecto ha tenido una importante vertiente investigadora lo que ha resultado muy interesante, aunque más complejo, ya que no se ha limitado al desarrollo de un sistema de acuerdo a una serie de requisitos, si no que se ha partido de un estudio de la situación actual en el campo de las simulaciones VANET con el que se han podido observar una serie de problemas que han motivado el desarrollo del sistema final. En esta investigación no sólo se han visto los problemas que afectan a las simulaciones VANET, también se han observado las prácticas de simulación más populares y se ha comprendido muy claramente el proceso de simulación y las distintas alternativas disponibles.

Desarrollar un sistema que pretende cubrir una serie de necesidades detectadas en un proceso de investigación ha sido, además de interesante, gratificante, al poder verse claramente el proceso entre la investigación de un problema o necesidad y el desarrollo de la solución propuesta.

7.2 Líneas futuras

En esta sección se detallan las posibles extensiones que se pueden realizar al sistema desarrollado en este proyecto fin de carrera. Dichas extensiones se han identificado a lo largo del desarrollo del sistema y se centran principalmente en mejorar el proceso de generación de trazas de movilidad.

7.2.1 Nuevas fuentes de trazas de movilidad

Si bien el sistema es capaz de generar trazas de movilidad para buena parte de los escenarios posibles con las dos herramientas con las que se integra, CityMob y SUMO, la inclusión de nuevas fuentes de trazas de movilidad ha sido uno de los puntos a tener en cuenta, tal y como se ha indicado en capítulos anteriores, por lo que la primera línea a seguir en la mejora del sistema podría pasar por la inclusión de nuevas fuentes de trazas de movilidad para su uso en la creación de experimentos de simulación. Son muchas las opciones que existen a la hora de añadir nuevas fuentes de trazas de movilidad, pero se recomienda empezar por incluir trazas generadas con VanetMobiSim ya que es una de las fuentes de trazas de movilidad más utilizadas y permite la exportación de trazas de movilidad para NS-2 lo que facilita bastante la integración con el sistema. Otra opción interesante para añadir como fuente de trazas de movilidad es BonnMotion [69] que también permite exportar trazas de movilidad para NS-2.

7.2.2 Nuevas opciones de creación de escenarios de SUMO

Un vector importante de extensión de las funcionalidades del sistema es la definición de escenarios de movilidad para SUMO, ya sea centrándose en la definición de la movilidad de los vehículos o en la definición de los mapas sobre los que se desarrollará la simulación de movilidad.

La generación de movimientos de vehículos aleatoria para SUMO es una manera rápida y simple de generar movimientos con los que obtener trazas de movilidad, pero esta sencillez (sólo dos parámetros configurables) presenta problemas como la falta de control sobre más parámetros de la movilidad o el comportamiento poco realista de los vehículos. El objetivo buscado era permitir a usuarios no familiarizados con el uso de simuladores de movilidad crear movimientos para los vehículos pero dicho objetivo se puede lograr ofreciendo al usuario una gama más amplia de opciones sin requerir conocimientos específicos de SUMO.

La primera opción sería la definición de movilidad de forma manual mediante la generación de tráfico basada en la actividad en un mapa [70]. Esta forma de generar movilidad de vehículo se basa en la premisa de que el tráfico es generado por personas que necesitan desplazarse de un lugar a otro para llevar a cabo alguna actividad. Implementando un sistema que permita mediante menús desplegables asignar actividades a distintos puntos de un mapa se lograría una generación de trazas más realista. Para esta mejora se podría utilizar la aplicación *SUMO Traffic Modeler* [71] incluida en la última versión de SUMO.

Otra opción pasaría por la implementación de un sistema que permita la definición manual de rutas y flujos de tráfico sin necesidad de editar manualmente los ficheros que definen la movilidad. Una opción para esto sería analizar el fichero que contiene el mapa y almacenar todos sus nodos, permitiendo al usuario definir una ruta entre dos nodos mediante un menú desplegable que muestre todos los nodos disponibles.

Respecto a la definición de mapas sobre los que se desarrollarán las simulaciones de movilidad se han detectado dos posibles extensiones de la funcionalidad ofrecida actualmente por el sistema: la primera es la inclusión de nuevos parámetros para la definición de mapas como por ejemplo la posibilidad de escalar las velocidades definidas en el mapa, la segunda opción, y la más interesante, es la integración del sistema con un editor de mapas de SUMO como es netEditor [72], esta mejora permitiría a un usuario sin conocimientos específicos de SUMO dibujar su propio mapa sobre el que se desarrollará la simulación y, en combinación con las mejoras especificadas anteriormente en este apartado, definir flujos de tráfico o actividades sobre este mapa de forma que se podrían crear trazas de movilidad sobre un mapa que él mismo ha definido con una movilidad definida a alto nivel, mediante flujos o actividades, o a bajo nivel, definiendo las rutas entre nodos una a una; todo ello sin tener un conocimiento específico del funcionamiento de SUMO ni de su estructura interna.

7.2.3 Pestaña de configuración de aplicación a simular

Otra posibilidad de mejora detectada sería la inclusión de una pestaña para la configuración de la aplicación a simular en la interfaz de configuración de la simulación de red. El contenido de dicha pestaña variaría en función de un fichero de configuración y su objetivo sería mostrar los parámetros principales de configuración de la aplicación que se va a simular en la simulación de red. De esta forma, un usuario que deseara validar la aplicación de otra persona podría utilizar este fichero para que el sistema le mostrara los parámetros de configuración que puede modificar, pudiendo prescindirse así de la edición manual de los ficheros de simulación de red.

7.2.4 Nuevos métodos para definir posiciones de RSU

La versión actual del sistema permite la inclusión manual de posiciones de RSU especificando las coordenadas de cada una de las RSU que se desean incluir en la simulación. Una mejora muy clara en este aspecto sería la posibilidad de que el sistema definiera posiciones para un número de RSU especificado por el usuario, este generación de posiciones podría hacerse de forma aleatoria o atendiendo a una diversos parámetros como por ejemplo a la concentración de RSU en un área determinada. Estas funcionalidades se estudiaron durante el desarrollo del sistema pero no se abordaron finalmente debido a que implicaban un importante riesgo y dificultad añadida en comparación con los beneficios que se podían obtener. Una de estas dificultades fue el control de la creación automática de RSU en las proximidades de una carretera ya que se podrían definir, por ejemplo, un número de RSU en un área determinada pero estas quedar ubicadas aleatoriamente sin ninguna carretera dentro de su alcance.

Otra posible mejora para la definición de posiciones de RSU pasa por simplificar el proceso de definición manual de las posiciones, mostrando al usuario el mapa sobre el

que se realizará la simulación y creando una RSU en aquellos lugares del mapa donde el usuario pulse.

7.2.5 Integración de las simulaciones de movilidad y de red

En línea con los avances que se están produciendo en el campo de los simuladores de VANET sería interesante extender la funcionalidad de la aplicación para integrar la simulación de movilidad con la de red, esto se podría hacer con SUMO como simulador de red a través de la interfaz TraCI [73] que es capaz de comunicar SUMO con el simulador de red, de esta forma se conseguiría crear simulaciones en las que el comportamiento de los vehículos cambiara en función de la información intercambiada con otros vehículos o RSU. De esta forma se conseguiría ampliar la utilidad del sistema a protocolos y aplicaciones que requieran de esta interacción entre la simulación de movilidad y la de red. Cabe destacar que esta extensión del sistema sería muy exigente debido a su complejidad.

7.2.6 Edición de experimentos ya existentes

Otra mejora detectada durante el desarrollo del sistema fue la posibilidad de modificar experimentos de simulación ya existentes, permitiendo al usuario seleccionar un experimento ya creado y modificar distintos parámetros del mismo para crear un nuevo experimento modificado. De esta forma se permitiría que un usuario que tuviera un experimento de simulación que le resulte interesante pudiera modificarlo para su uso con otras aplicaciones o para compartirlo con otros usuarios sin tener que editar manualmente los distintos ficheros de simulación de red que pueden conformar el experimento. Esta funcionalidad permitiría también modificar experimentos generados previamente sin tener que repetir todo el proceso de generación ni editar manualmente los ficheros de simulación de red

7.2.7 Nuevos simuladores de red

Pese a que NS-2 es el simulador de red más utilizado, con diferencia, para la simulación de VANET en las publicaciones analizadas, tal y como se puede ver en la sección 2.2, podría ser interesante en ciertos ámbitos la inclusión de otros simuladores de red populares como Qualnet. La inclusión de un nuevo simulador de red supondría la modificación buena parte del código por lo que se deberían de estudiar muy a fondo los beneficios obtenidos con esta extensión ya que de llevarse a cabo acarrearía un gran esfuerzo de desarrollo.

7.2.8 Convertir VanSimFM en un servicio online

Una de las mejoras más interesantes de todas las que se han identificado, y posiblemente también la más compleja, es portar VanSimFM a una plataforma de aplicaciones online, como Amazon Web Services [74], de forma que se pudiera utilizar a través de una interfaz web. De esta forma se aumentaría significativamente el número de usuarios que podrían hacer uso de la aplicación, al poderse prescindir de la instalación de todas las

dependencias necesarias. El único inconveniente de esta mejora es que se no se portaría toda la funcionalidad si no únicamente la relacionada con la generación de trazas de movilidad, ya que para la ejecución de distintos protocolos y aplicaciones en de red es necesario recompilar el simulador NS-2 haciendo inviable la ejecución de estos experimentos en un sistema totalmente online. Por tanto se dejaría en manos del usuario final la ejecución de las dos etapas finales de la simulación de VANET mostradas en la Figura 1.

Referencias

[1] S. Olariu and M.C. Weigle, *Vehicular Networks from Theory to Practice*, Chapman & Hall, 2008.

[2] SUMO, Simulation of Urban Mobility [En línea]
Diponible: http://sumo.sourceforge.net/

[3] CityMob [En línea]
Disponible: http://www.grc.upv.es/Software/citymob.html

[4] VanetMobiSim [En línea]
Disponible: http://vanet.eurecom.fr/

[5] NS-2 [En línea]
Disponible: http://nsnam.isi.edu/nsnam/index.php/Main_Page

[6] Qualnet [En línea]
Disponible: http://www.scalable-networks.com/products/qualnet/

[7] NCTUns [En línea]
Disponible: http://www.estinet.com/products.php

[8] MOVE [En línea]
Disponible:
http://lens1.csie.ncku.edu.tw/wiki/doku.php?id=%E2%80%A7realistic_mobility_generator_for_vehicular_networks

[9] TraNS [En línea]
Disponible: http://lca.epfl.ch/projects/trans

[10] Tiger Maps [En línea]
Disponible: http://www.census.gov/geo/www/tiger/

[11] P. Urmeneta, *Simulation and Improvement of the Handover process in IEEE 802.11p based VANETs*, Tongji University, 2010.

[12] A. Hassan, *VANET Simulation*, Halmstad University, 2009.

[13] F.J. Martinez, C.K. Toh, J.Carlos Cano, C.T. Calafate, and P. Manzoni, *A survey and comparative study of simulators for vehicular ad hoc networks (VANETs)*, Wirel. Commun. Mob. Comput., 2009.

[14] M. Jerbi, S.-M. Senouci, T. Rasheed, Y. Ghamri-Doudane, *Towards Efficient Geographic Routing in Urban Vehicular Networks*, IEEE Transactions on Vehicular Technology *58*, 2009

[15] M. Gerla, *Proof-of-relevance: Filtering false data via authentic consensus in Vehicle Ad-hoc Networks*, IEEE INFOCOM, 2008

[16] M. Ghosh, A. Varghese, A. Gupta, A.A. Kherani, and S.N. Muthaiah, Ad *Hoc Networks Detecting misbehaviors in VANET with integrated root-cause analysis*, Ad Hoc Networks Volume 8, Issue 7, 2010.

[17] A. Wasef, Y. Jiang, and X. Shen, *DCS: An Efficient Distributed-Certificate-Service Scheme for Vehicular Networks*, IEEE Transactions on Vehicular Technology, Volume 59, Issue 2, 2010.

[18] A. Wasef and X. Shen, *EDR: Efficient Decentralized Revocation Protocol for Vehicular Ad Hoc Networks*, IEEE Transactions on Vehicular Technology, Volume 59, Issue 2, 2009.

[19] Y. Park, C. Sur, C. D. Jung and K-H Rhe, "*An Efficient Anonymous Authentication Protocol for Secure Vehicular Communications*, Journal of Information Science and Engineering 26, 2010.

[20] P. Cencioni and R. Di Pietro, *A Mechanism To Enforce Privacy In Vehicle-to-infrastructure Communicatio*, Computer Communications 31, 2008.

[21] J. Nzouonta, N. Rajgure, G. Wang, and C. Borcea, *VANET Routing on City Roads Using Real-Time Vehicular Traffic Information*, IEEE Transactions on Vehicular Technology, Volume 58, Issue 7, 2009.

[22] E. Schoch and F. Kargl, *On the Efficiency of Secure Beaconing in VANETs*, Wi-Sec '10 Proceedings of the third ACM conference on Wireless network security, 2010.

[23] Z. Wan, K. Ren, B. Zhu, B. Preneel, and M. Gu, *Anonymous User Communication for Privacy Protection in Wireless Metropolitan Mesh Networks*, IEEE Transactions on Vehicular Technology, Volume 59, Issue 2, 2010.

[24] M.E. Mahmoud and X.S. Shen, *PIS : A Practical Incentive System for Multihop Wireless Networks*, IEEE Transactions on Vehicular Technology, Volume 59, Issue 8, 2010.

[25] M. Strassberger, C. Schroth, and R. Lasowski, *Data Dissemination in Vehicular Networks*, Vehicular Networks: Techniques, Standards and Applications, 2009.

[26] E. van den Berg, T. Zhang, and S. Pietrowicz, *Blend-In: A Privacy-Enhancing Certificate-Selection Method for Vehicular Communication*, IEEE Transactions on Vehicular Technology, Volume 58, Issue 9, 2009.

[27] A. Kherani and A. Rao, *Performance of Node-Eviction Schemes in Vehicular Networks*, IEEE Transactions on Vehicular Technology, Volume 59, Issue 2, 2010.

[28] C.H. Yeh, Y.M. Huang, T.I. Wang, and H.H. Chen, *DESCV—A Secure Wireless Communication Scheme for Vehicle ad hoc Networking*, Mobile Networks and Applications, Volume 14, Issue 5, 2009.

[29] M.S. Kakkasageri and S.S. Manvi, *Intelligent Information Dissemination in Vehicular Ad hoc Networks*, International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC) Vol.2, No.1, 2011.

[30] A. Studer, F. Bai, B. Bellur, and A. Perrig, *Flexible, Extensible, and Efficient VANET Authentication*, Proceedings of the 6th Annual Conference on Embedded Security in Cars (ESCAR), 2008.

[31] M. Raya, P. Papadimitratos, V.D. Gligor, and J.-P. Hubaux, *On Data-Centric Trust Establishment in Ephemeral Ad Hoc Networks*, IEEE INFOCOM, 2008.

[32] P. Urmeneta, *Simulation and Improvement of the Handover process in IEEE 802.11p based VANETs*, Tongji University, 2010.

[33] D. Huang and M. Verma, *ASPE: attribute-based secure policy enforcement in vehicular ad hoc networks*, Privacy and Security in Wireless Sensor and Ad Hoc Networks, Volume 7, Issue 8, 2009.

[34] B. Aslam, S. Park, C.C. Zou, and D. Turgut, *Secure traffic data propagation in Vehicular Ad Hoc Networks*, International Journal of Ad Hoc and Ubiquitous Computing, Volume 6, Issue 1, 2010.

[35] X. Ma, X. Chen, and H.H. Refai, *Performance and reliability of DSRC vehicular safety communication: a formal analysis*, EURASIP Journal on Wireless Communications and Networking, 2009.

[36] A. Wasef, R. Lu, *Complementing public key infrastructure to secure vehicular Ad-hoc networks*, *IEEE Wireless Communications*, Vol. 17, No. 5, 2010.

[37] Y. Sun, S. Member, R. Lu, X. Lin, X.S. Shen, and J. Su, *An Efficient Pseudonymous Authentication Scheme With Strong Privacy Preservation for Vehicular Communications*, IEEE Transactions on Vehicular Technology, Volume 59, Issue 7, 2010.

[38] A. Rajabi, *Performance Evaluation of Vehicular Ad Hoc Networks using simulation tools*, Universidad Politécnica de Cataluña, 2010.

[39] N.-W. Lo and H.-C. Tsai, *A Reputation System for Traffic Safety Event on Vehicular Ad Hoc Networks*, EURASIP Journal on Wireless Communications and Networking, 2009.

[40] M. Abuelela and S. Olariu, *Automatic Incident Detection In VANETs: A Bayesian Approach*, VTC Spring 2009 - IEEE 69th Vehicular Technology Conference.

[41] T. Umedu, K. Isu, T. Higashino, and C.K. Toh, "An Intervehicular-Communication Protocol for Distributed Detection of Dangerous Vehicles," IEEE Transactions on Vehicular Technology

[42] A. Viejo, F. Sebé, and J. Domingo-Ferrer, "Aggregation of Trustworthy Announcement Messages in Vehicular Ad Hoc Networks," IEEE 69th Vehicular Technology Conference VTC2009

[43] Q. Wu, J. Domingo-Ferrer, and Ú. Gonzalez-Nicolas, *Balanced Trustworthiness, Safety, and Privacy in Vehicle-to-Vehicle Communications*, IEEE Transactions on Vehicular Technology, Volume 59, Issue 2, 2010.

[44] H.D. Weerasinghe, R. Tackett, and H. Fu, *Verifying position and velocity for vehicular ad-hoc networks*, Security Comm. Networks, 2010.

[45] La, T; No, X; La, A; No, R; On, G, *Research on Security and Privacy in Vehicular Ad Hoc Networks*¸ Universidad Rovira i Virgili, 2010.

[46] Z. Movahedi, R. Langar, G. Pujolle , *A Comprehensive Overview of Vehicular Ad hog Network Evaluation Alternatives*, 8th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT), 2010.

[47] B. Stroustrup, The C++ Programming Language, Addison–Wesley, 2000.

[48] Cygwin [En línea]
Disponible: http://www.cygwin.com/

[49] Java [En línea]
Disponible: http://www.java.com/es/

[50] C# [En línea]
Disponible: http://msdn.microsoft.com/es-es/vcsharp/default.aspx

[51] Microsoft .NET [En línea]
Disponible: http://www.microsoft.com/net/

[52] Mono [En línea]
Disponible: http://www.mono-project.com/Main_Page

[53] Python [En línea]
Disponible: http://www.python.org/

[54] Ruby [En línea]
Disponible: http://www.ruby-lang.org/es/

[55] Windows Forms [En línea]
Disponible: http://windowsclient.net/

[56] Microsoft Visual Studio [En línea]
Disponible: http://msdn.microsoft.com/es-es/vstudio/aa718325

[57] Swing [En línea]
Disponible: http://download.oracle.com/javase/1.5.0/docs/guide/swing/

[58] AWT [En línea]
Disponible: http://download.oracle.com/javase/1.5.0/docs/api/java/awt/package-summary.html

[59] Netbeans [En línea]
Disponible: http://netbeans.org/

[60] Shoes [En línea]
Disponible: http://shoesrb.com/

[61] GTK [En línea]
Disponible: http://www.gtk.org/

[62] Qt [En línea]
Disponible: http://qt.nokia.com/

[63] ESA [En línea]
Disponible:
http://cisas.unipd.it/didactics/STS_school/Software_development/Guide_to_the_SW_requirements_definition_phase-0503.pdf.

[64] Erich Gamma, Richard Helm, Palph Jhonson y John M. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software.

[65] JDOM [En línea]
Disponible: http://www.jdom.org/

[66] Mapa de la ciudad de Eichstätt en formato OSM [En línea]
Disponible:
http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Eichstaett.osm

[67] J. Munera, J. M. de Fuentes, A. I. González-Tablas, *Towards a comparable evaluation for VANET protocols: NS-2 experiments builder assistant and extensible test bed*, Universidad Carlos III Madrid, 2011.

[68] ESCAR [En línea]
Disponible: https://www.escar.info/

[69] BonnMotion [En línea]
Disponible: http://net.cs.uni-bonn.de/wg/cs/applications/bonnmotion/

[70] Generación de tráfico con SUMO a partir de actividad [En línea]
Disponible:
http://sourceforge.net/apps/mediawiki/sumo/index.php?title=ACTIVITYGEN

[71] SUMO Traffic Modeler [En línea]
Disponible:
http://sourceforge.net/apps/mediawiki/sumo/index.php?title=SUMO_Traffic_Modeler

[72] SUMO netEditor [En línea]
Disponible: http://paginas.fe.up.pt/~ee06201/thesis/

[73] SUMO Traci [En línea]
Disponible: http://sourceforge.net/apps/mediawiki/sumo/index.php?title=TraCI

[74] Amazon Web SErvices [En línea]
Disponible: http://aws.amazon.com/es/

[75] Visual Paradigm [En línea]
Disponible: http://www.visual-paradigm.com/

[76] Virtual Box [En línea]
Disponible: http://www.virtualbox.org/

[77] Apple OSX 10.6 [En línea]
Disponible: http://www.apple.com/es/macosx/

[78] Ubuntu 10.10 [En línea]
Disponible: http://www.ubuntu.com/

[79] Textmate [En línea]
Disponible: http://macromates.com/

[80] Microsoft Office 2011 para mac [En línea]
Disponible: http://emea.microsoftstore.com/es/es-ES/Microsoft/Office/Office-para-Mac-2011?WT.mc_id=pointitsem_ES_officemac_2011_Buy&WT.srch=1

[81] Microsoft Project 2010 [En línea]
Disponible: http://emea.microsoftstore.com/es/es-ES/Microsoft/Project-Standard-2010

[82] Apple Safari [En línea]
Disponible: http://www.apple.com/es/safari/

[83] Plantilla para memoria de PFC ofrecida por la Universidad [En línea]
Disponible:

https://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Plantilla_PFC.doc

[84] AWK scripts [En línea]
Disponible: http://www.gnu.org/software/gawk/gawk.html

Acrónimos

DOM	*Document Object Model*
GPS	*Global Positioning System*
JVM	*Java Virtual Machine*
MANET	*Mobile Ad-hoc Network*
OBU	*On Board Unit*
RSU	*Road Side Unit*
VANET	*Vehicular Ad-hoc Network*
XML	*Extensible Markup Language*

Anexo 1: Gestión del proyecto

En este anexo se detallan aspectos referentes a la gestión del proyecto que incluyen la planificación del trabajo, los medios técnicos utilizados para el desarrollo de todo el proyecto así como el análisis económico del mismo.

1. Planificación del trabajo

En esta sección de detallan la planificación inicial y el desarrollo real del proyecto así como un estudio de las desviaciones observadas entre ambos

1.1 Planificación inicial

En esta sección se muestra la planificación inicial elaborada para el desarrollo del proyecto, en ella se pueden observar las distintas fases en las que se ha descompuesto el proyecto junto con las estimaciones de esfuerzo realizadas para cada una de ellas. En la elaboración de esta planificación se han tenido en cuenta jornadas de 5 horas diarias en días laborables. El motivo de esta planificación se debe a que se ha tenido que compaginar el desarrollo de este proyecto con la realización de un máster los fines de semana y un trabajo a jornada completa.

La planificación inicial del proyecto abarca desde el 10 de febrero hasta el 13 de junio de 2.011 lo que hace un total de 88 días laborables.

La Figura 24 muestra el diagrama de Gantt con la planificación inicial mientras que la Tabla 11 muestra el calendario previsto para cada tarea.

Proyecto fin de carrera	**88 días**	**jue 10/02/11**	**lun 13/06/11**
Planificación inicial	**2 días**	**jue 10/02/11**	**vie 11/02/11**
Estudio del estado del arte	**5 días**	**lun 14/02/11**	**vie 18/02/11**
Análisis de herramientas existentes	4 días	lun 14/02/11	jue 17/02/11
Prácticas de simulación actuales	5 días	lun 14/02/11	vie 18/02/11
Análisis	**10 días**	**lun 21/02/11**	**vie 04/03/11**
Arquitectura del sistema	3 días	lun 21/02/11	mié 23/02/11
Estudio tecnológico	2 días	jue 24/02/11	vie 25/02/11
Definición de casos de uso	2 días	lun 28/02/11	mar 01/03/11
Definición de requisitos software	3 días	mié 02/03/11	vie 04/03/11
Diseño	**12 días**	**lun 07/03/11**	**mar 22/03/11**
Diseño de software	6 días	lun 07/03/11	lun 14/03/11
Diagramas de secuencia	6 días	mar 15/03/11	mar 22/03/11
Implementación	**35 días**	**mié 23/03/11**	**mar 10/05/11**
Pruebas	**4 días**	**mié 11/05/11**	**lun 16/05/11**
Documentación	**20 días**	**mar 17/05/11**	**lun 13/06/11**

Tabla 11 - Planificación inicial detallada

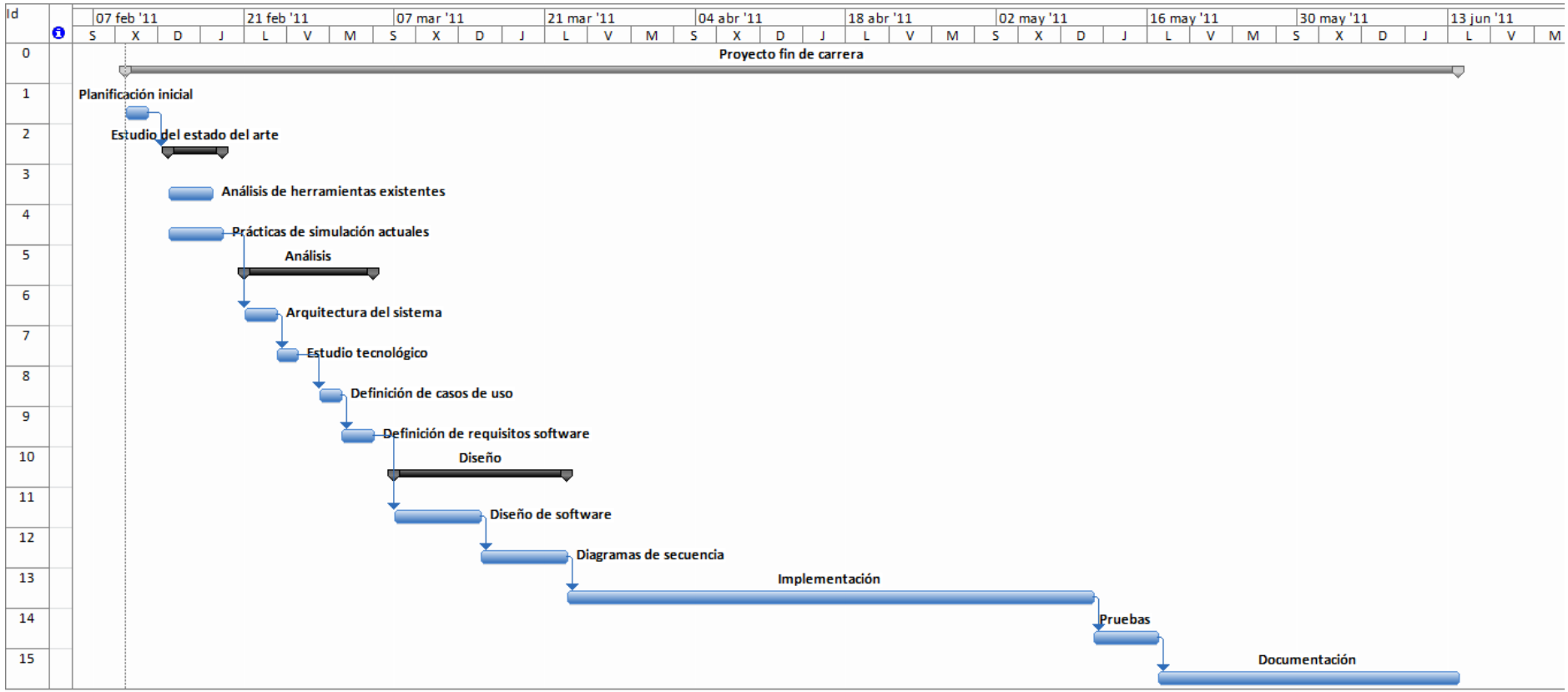

Figura 24 - Planificación inicial del proyecto

Como se puede apreciar en esta planificación inicial más de la mitad del tiempo estimado para el desarrollo de este proyecto está destinado al diseño y la implementación, sin olvidar la documentación, parte esencial en todo proyecto.

1.2 Desarrollo real del proyecto

En esta sección se muestra el desarrollo real del proyecto y se compara con la planificación inicial para estudiar las desviaciones surgidas a lo largo del proyecto.

La Figura 25 muestra el diagrama de Gantt del desarrollo final del proyecto, en ella se puede apreciar que el reparto de tiempos es similar al inicialmente estimado, si bien la duración de algunas de las tareas ha sido mayor. Esto se puede observar con más detalle en la Tabla 12 que muestra el desarrollo detallado del proyecto. En ella se puede apreciar que la duración final ha sido de 115 días frente a los 88 inicialmente planificados. Se puede observar en esta tabla una nueva tarea añadida a la planificación adicional y que está dedicada a la elaboración de un artículo a partir de los resultados obtenidos durante el desarrollo del proyecto.

Proyecto fin de carrera	**115 días**	**jue 10/02/11**	**mié 20/07/11**
Planificación inicial	**2 días**	**jue 10/02/11**	**vie 11/02/11**
Estudio del estado del arte	**4 días**	**lun 14/02/11**	**vie 18/02/11**
Análisis de herramientas existentes	4 días	lun 14/02/11	jue 17/02/11
Prácticas de simulación actuales	5 días	lun 14/02/11	vie 18/02/11
Análisis	**11 días**	**lun 21/02/11**	**lun 07/03/11**
Arquitectura del sistema	3 días	lun 21/02/11	mié 23/02/11
Estudio tecnológico	3 días	jue 24/02/11	lun 28/02/11
Definición de casos de uso	2 días	mar 01/03/11	mié 02/03/11
Definición de requisitos software	3 días	jue 03/03/11	lun 07/03/11
Diseño	**14 días**	**mar 08/03/11**	**vie 25/03/11**
Diseño de software	8 días	mar 08/03/11	jue 17/03/11
Diagramas de secuencia	6 días	vie 18/03/11	vie 25/03/11
Implementación	**42 días**	**lun 28/03/11**	**mar 24/05/11**
Pruebas	**2 días**	**mié 25/05/11**	**jue 26/05/11**
Elaboración del artículo	**18 días**	**vie 27/05/11**	**mar 21/06/11**
Documentación	**28 días**	**lun 13/06/11**	**mié 20/07/11**

Tabla 12 – Desarrollo real del proyecto detallado

Adicionalmente, la Tabla 13 muestra un análisis comparativo entre la planificación inicial y el desarrollo real del proyecto. Esta tabla ayudará a visualizar más rápidamente las desviaciones producidas durante el desarrollo del proyecto.

Tal y como se puede observar en dicha tabla, se ha producido una desviación total de cerca del 30% sobre la duración indicada en la estimación inicial. Se puede buscar el origen de esta importante desviación en la falta de experiencia en la planificación de proyectos, en la estimación de la duración de las distintas tareas, en las variaciones en la dispo-

nibilidad que se han producido a lo largo del proyecto y, por último, en la ampliación de las tareas a realizar para incluir la elaboración del artículo.

A pesar de que parte de la tarea de elaboración del artículo se hizo en paralelo con la documentación del proyecto, en concreto y 7 días, el resto de los días en los que se trabajó en el artículo exclusivamente han supuesto la casi la mitad del tiempo total que se ha desviado la ejecución del proyecto respecto a la planificación inicial.

Dado que la elaboración del artículo no se tuvo en consideración en la planificación inicial, se estudiarán las desviaciones de la planificación inicial sobre la final sin tener en cuenta esta tarea. De esta forma se puede ver cómo la mayor parte de la desviación no asociada a la elaboración del artículo se ha producido en las tareas de implementación y documentación. Estas desviaciones se deben a las dificultades que se han ido encontrando durante la ejecución de estas tareas y que, en algunos casos, han llegado a suponer más de un día de retraso. Si a esto le sumamos importantes variaciones en la disponibilidad, como se ha mencionado anteriormente, podemos encontrar una explicación a las principales desviaciones surgidas.

Además de las desviaciones debido a subestimaciones también podemos encontrar desviaciones debidas a sobreestimaciones hechas en la planificación inicial. Estas desviaciones tienen una importancia marginal, 3 días menos en total, pero en el caso de las pruebas ha implicado una reducción del 50% del tiempo total planificado. Esta diferencia en el tiempo dedicado finalmente a las pruebas se debe a que al llegar a esta fase el sistema estaba ampliamente probado lo que hizo que las pruebas transcurrieran más rápido y con más fluidez de lo esperado.

	Planificado	Real	Diferencia	Variación
Proyecto fin de carrera	**88 días**	**115 días**	**27 días**	**30,68%**
Planificación inicial	**2 días**	**2 días**	**0 días**	**0,00%**
Estudio del estado del arte	**5 días**	**5 días**	**0 días**	**0,00%**
Análisis de herramientas existentes	4 días	4 días	0 días	0,00%
Prácticas de simulación actuales	5 días	5 días	0 días	0,00%
Análisis	**10 días**	**11 días**	**1 día**	**10,00%**
Arquitectura del sistema	3 días	3 días	0 días	0,00%
Estudio tecnológico	2 días	3 días	1 día	50,00%
Definición de casos de uso	2 días	2 días	0 días	0,00%
Definición de requisitos software	3 días	3 días	0 días	0,00%
Diseño	**12 días**	**14 días**	**2 días**	**16,67%**
Diseño de software	6 días	8 días	2 días	33,33%
Diagramas de secuencia	6 días	6 días	0 días	0,00%
Implementación	**35 días**	**42 días**	**7 días**	**20,00%**
Pruebas	**4 días**	**2 días**	**-2 días**	**-50,00%**
Elaboración del artículo	**--**	**18 días**	**18 días**	**--**
Documentación	**20 días**	**28 días**	**8 días**	**40,00%**

Tabla 13 - Análisis de las desviaciones en la planificación

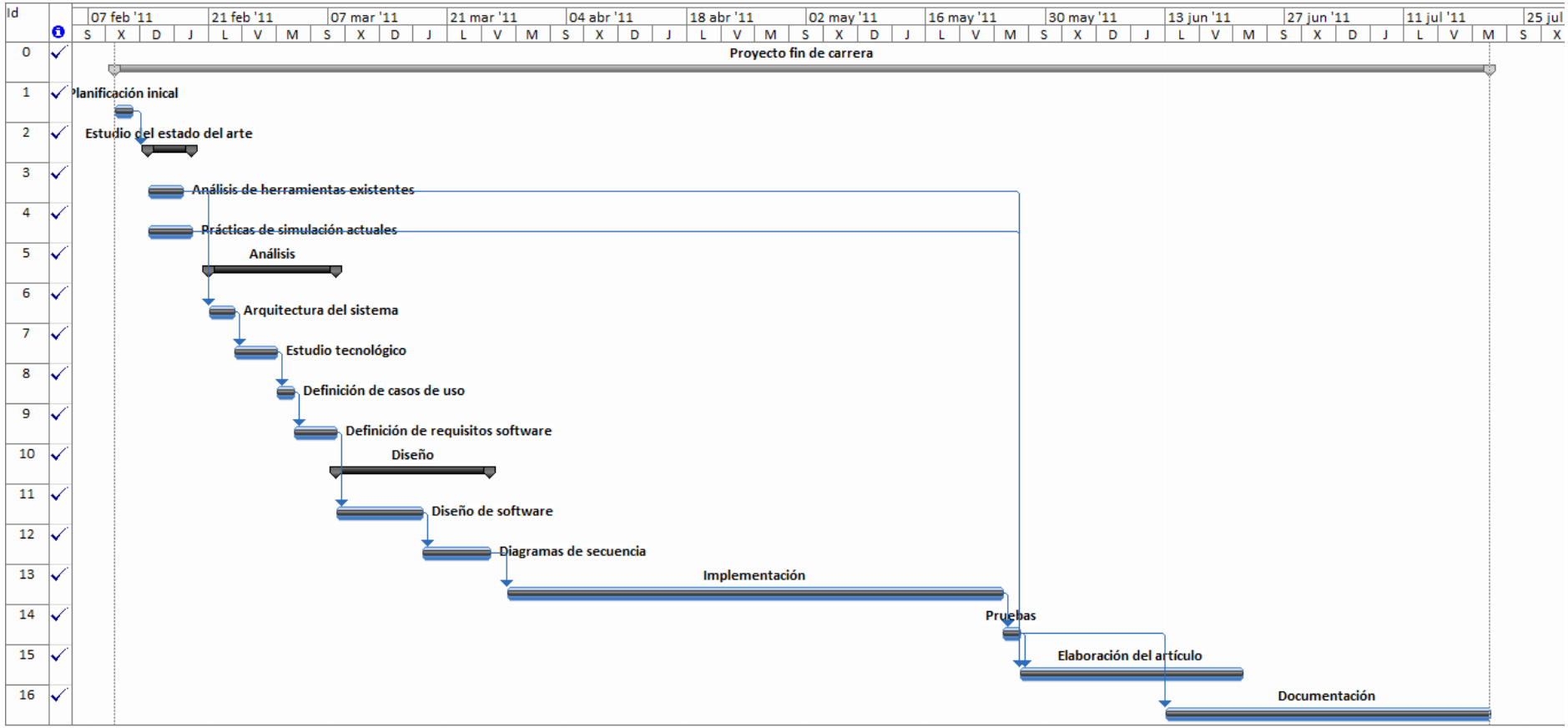

Figura 25 - Desarrollo real del proyecto

2. Medios técnicos empleados para el proyecto

En esta sección se detallan las herramientas que se han utilizado a lo largo del desarrollo del proyecto. La Tabla 14 muestra las distintas herramientas utilizadas junto con una breve descripción.

Herramienta	Descripción
NetBeans 6.9 [59]	Entorno de desarrollo utilizado para la implementación de todas las funcionalidades del sistema además de la interfaz de usuario.
Visual Paradigm [75]	Herramienta de modelado UML utilizada para la creación de todos los diagramas utilizados en la fase de análisis y diseño.
VirtualBox [76]	Software de virtualización utilizado para la creación de la máquina virtual sobre la que se ha instalado el sistema operativo Linux sobre el que se han realizado todas las pruebas de la aplicación.
Apple OSX 10.6 [77]	Sistema operativo sobre el que se ha desarrollado las tareas de análisis, diseño y documentación del proyecto además de parte de la implementación del sistema.
Ubuntu 10.10 [78]	Sistema operativo sobre el que se ha realizado parte de la implementación del sistema y las pruebas de aceptación.
TextMate [79]	Editor de texto con utilizado para la visualización de ficheros de simulación y código fuente tanto del propio sistema como de otras herramientas analizadas.
Microsoft Office 2011 (OSX) [80]	Suite ofimática utilizada para la creación de la documentación del proyecto.
Microsoft Project 2010 [81]	Aplicación para la gestión de planificaciones utilizada para realizar la planificación y el seguimiento del proyecto.
Apple Safari [82]	Navegador web utilizado a lo largo de todo el desarrollo del proyecto..

Tabla 14 - Detalle de las herramientas utilizadas en el proyecto

Junto con las herramientas de software utilizadas se detalla a continuación en la Tabla 15 el equipo utilizado para el desarrollo del proyecto.

Equipo	Descripción
Portátil: MacBook Pro 15-inch 2,4 GHz, 4GB de RAM	Ordenador sobre el que se ha realizado todo el desarrollo de este proyecto fin de carrera.
Impresora: HP Officejet 6500A	Impresora con la que se han impreso los distintos documentos que se han revisado durante el desarrollo de este proyecto.
Teléfono:	Teléfono móvil inteligente utilizado, también como agenda y

iPhone 4 32GB	lector de correos en ausencia del ordenador principal usado.

Tabla 15 - Detalle de los equipos utilizados en el proyecto

3. Análisis económico del proyecto

En esta sección se muestra el análisis económico del proyecto que incluye la especificación de la metodología utilizada para la estimación de costes, el presupuesto inicial, el presupuesto para el cliente y el coste real del proyecto.

Para el desarrollo de esta sección se ha tomado como base la plantilla proporcionada en la página web de la Universidad para este fin [83].

3.1 Metodología de estimación de costes

La estimación de costes se ha desarrollado tomando en consideración costes indirectos y directos.

Los costes directos se corresponden con los conceptos que están relacionados directamente con el desarrollo del proyecto. Entre estos conceptos se encuentran los gastos de mano de obra, equipos, herramientas de software, dietas y gastos de viaje.

Los gastos de mano de obra se estimarán contando que no se es autónomo y se está empleado para una empresa, los precios de los equipos y del software utilizado son los precios de venta al público, el valor de las dietas se fija como pactado entre empleado y empresa y los gastos de desplazamiento se fijan en función de estimaciones de kilometraje.

Los costes indirectos son aquellos costes en los que se incurre para el desarrollo del proyecto pero que no tienen que ver directamente con el desarrollo del sistema objetivo. Entre estos gastos se encuentra el teléfono, la conexión a internet y el alquiler del inmueble en el que se realiza la actividad, entre otros. Para este proyecto, y de acuerdo con la plantilla proporcionada en la web de la Universidad, los costes indirectos se han calculado como un 20% de los costes directos.

3.2 Presupuesto inicial

En esta sección se muestra el presupuesto inicial donde se detallarán los coste presupuestados para todo el proyecto junto con el presupuesto total estimado.

El 18% de IVA no está incluido en ninguno de los gastos reflejados en esta sección salvo en el coste total del proyecto en el que se especifica claramente que se ha añadido el IVA para su cálculo.

3.2.1 Gastos de personal

La Tabla 16 detalla los gastos de personal. Para su cálculo se ha tenido en cuenta la dedicación de un único ingeniero a lo largo de todo el proyecto, con una dedicación de 5 horas diarias en días laborables durante los 88 días de duración estimada del proyecto. El coste hombre mes del recurso se ha tomado del modelo proporcionado por la Universidad, si bien se ha considerado que es un importe bruto al que se ha añadido el importe a pagar a la Seguridad Social, 28,3% para el régimen general de la Seguridad Social.

Recurso	Dedicación	Coste hombre/mes	Coste bruto	Seg. Soc.	**Coste total**
Ingeniero	3,352 hombre/mes	2.694,39 €	9.032,62 €	2.556,23 €	**11.588,85 €**

Tabla 16 - Gastos de personal

Como se puede ver, los cálculos arrojan una dedicación de cerca de 3,3 hombres mes, lo que supone un coste total bruto de 9.032,62 € a los que se suma el importe a pagar a la Seguridad Social, quedando el coste total de personal en 11.588,85 €.

3.2.2 Gastos de equipos

En esta sección se detallan los gastos relacionados con los equipos utilizados a lo largo del proyecto. Dichos equipos son un ordenador portátil, una impresora y un teléfono móvil. La Tabla 17 muestra en detalle los costes imputables para cada equipo así como de los datos utilizados para calcularlos: porcentaje de uso dedicado, coste y periodo de depreciación.

El periodo de depreciación para el ordenador y la impresora es el utilizado en el ámbito de la Administración General mientras que el del teléfono es el periodo de lanzamiento entre nuevos modelos.

Descripción	Coste [1]	Dedicación	Periodo depreciación	Coste imputable
Portátil: MacBook Pro 15-inch 2,4 GHz, 4GB de RAM Teclado y ratón externos	2.193,00 €	2,93 meses	36 meses	178,69 €
Impresora: HP Officejet 6500A	105,78 €	2,93 meses	36 meses	8,62 €
Teléfono: iPhone 4 32GB	592,37 €	2,93 meses	24 meses	72,40 €
			Coste total imputable	**259,71 €**

Tabla 17 - Gatos de equipos

[1] Precio de venta al público, sin IVA.

3.2.3 Gastos de software

En esta sección se detallan los gastos relacionados con el software utilizado a lo largo del proyecto. La Tabla 18 muestra en detalle los costes imputables para cada aplicación utilizada de la misma forma que se hizo en la sección 3.2.2 con los gastos de equipos. Esta tabla solo muestra el software mencionado en la sección 2 de este anexo que no sea gratuito.

Adicionalmente se ha incluido una donación de 50 € para el desarrollo del software libre ya que durante el desarrollo del proyecto se ha hecho uso de diversas herramientas de software que han sido de gran utilidad.

Descripción	Coste [1]	Dedicación	Periodo depreciación	Coste imputable
Apple OSX 10.6	30,00 €	2,93 meses	36 meses	2,44 €
Microsoft Office 2011	116,81 €	2,93 meses	36 meses	9,52 €
Microsoft Project 2010	651,26 €	2,93 meses	36 meses	53,07 €
Visual Paradigm UML 8.2	405,74 €	2,93 meses	36 meses	33,06 €
TextMate	36,80 €	2,93 meses	36 meses	3,00 €
Donación Software Libre	50,00 €	--	--	50,00 €
		Coste total imputable		**151,09 €**

Tabla 18 - Gastos de software

[1] Precio de venta al público, sin IVA.

3.2.4 Gastos de consumibles

En esta sección se detallan los gastos que se han realizado en concepto de consumibles. Dichos gastos se reducen para este proyecto a cartuchos de tinta para la impresora y material de oficina diverso como son folios, bolígrafos, etc. La Tabla 19 muestra un resumen de los gastos realizados en consumibles a lo largo del proyecto.

Descripción	Coste unitario [1]	Cantidad	Coste total
Cartuchos de impresora	61,5 €	1	61,5 €
Material de oficina	24,6 €	1	24,6 €
	Coste total		**86,10 €**

Tabla 19 - Gastos de consumibles

[1] Precio de venta al público, sin IVA.

3.2.5 Gastos de viajes y dietas

En esta sección se detallan los gastos de viajes y dietas realizados durante el proyecto. A efectos de cliente se considera al tutor de este proyecto y por tanto, puesto que el cliente está ubicado en Leganés, fuera del término municipal de Madrid los gastos de viajes se han de imputar al proyecto y se calcularán en base a los kilómetros diarios con un coste de 0,104 €/Km. Además se ha asignado una importe diario en concepto de dietas de 8 euros al día.

La Tabla 20 muestra los gastos en concepto de viajes y dietas.

Descripción	Coste unitario	Cantidad	Coste total
Gasolina viaje a Leganés	0,104 €/Km	390 Km	40,56 €
Dietas	8 €/día	88 días	704,00 €
	Coste total		**744,56 €**

Tabla 20 - Gastos de viajes y dietas

3.2.6 Costes directos

En esta sección es muestran los costes directos asociados a este proyecto que son la suma de los conceptos calculados en los apartados anteriores: gastos de personal, gastos de equipos, gastos de software, gastos de consumibles y gastos de viajes y dietas. La Tabla 21 muestra la suma de estos conceptos dando como resultado el total de costes directos del proyecto.

Concepto	Coste
Gastos de personal	11.588,85 €
Gastos de equipos	259,71 €
Gastos de software	151,09
Gastos de consumibles	86,10 €
Gastos de viajes y dietas	744,56 €
Costes directos	**12.830,31 €**

Tabla 21 - Costes directos

3.2.7 Costes indirectos

Los costes indirectos se han calculado de acuerdo a lo visto en la sección 3.1 de este anexo como el 20% de los costes directos.

De esta forma y con los resultados obtenidos en la sección anterior, los costes indirectos del proyecto son 2.566,06 €.

3.2.8 Estimación de costes

En esta sección se muestra la Tabla 22, donde s detalla la estimación de costes del proyecto a partir de los cálculos realizados en las secciones anteriores. En ella se detallan los costes identificados en todas las secciones y la suma de todos ellos sin IVA, para añadirle más adelante el IVA y obtener así una estimación inicial con IVA.

Concepto	Coste
Gastos de personal	11.588,85 €
Gastos de equipos	259,71 €
Gastos de software	151,09
Gastos de consumibles	86,10 €
Gastos de viaje y dietas	744,56 €
Gastos directos	12.830,31 €
Gastos indirectos	2.566,06 €.
Total gastos sin IVA	**15.396,37 €**
IVA (18%)	2.771,35 €
Total	**18.167,72€**

Tabla 22 – Estimación de costes

3.3 Presupuesto para el cliente

En esta sección es muestra el presupuesto a presentar al cliente, en él se agrupan los gastos vistos en anteriores tablas y añadiendo a los cálculos un porcentaje de riesgo con el que poder hacer frente a imprevistos surgidos. Finalmente se incorporan al coste final los beneficios esperados por el desarrollo de este proyecto, dichos beneficios se calcularán como un porcentaje del coste total incluidos riesgos.

El porcentaje de riesgos definido para el proyecto es del 10%, dicho valor se ha definido de acuerdo a los valores que se han venido utilizando a lo largo de la carrera en otros proyectos.

Se ha definido en un 15% el porcentaje que se añadirá al presupuesto en concepto de beneficios. Se ha decidido utilizar un 15% porque, al igual que pasaba con el riesgo, entraba dentro del rango de valores vistos en otros proyectos. Estos beneficios serán el único dinero que se perciba por el desarrollo del sistema ya que se pondrá a disposición del público como herramienta de libre distribución.

Aún comercializándola, la aplicación no tendría un gran interés comercial, pero sin embargo cuenta con características que la hacen muy atractiva para personal que esté investigando en el campo de las VANET y necesite validar sus propuestas. Como se ha visto en el Capítulo 1, la experimentación real es prácticamente inviable, mucho más aún en un escenario de crisis como en el que vivimos. Esto hace que la simulación tenga más importancia aún si cabe, lo que ubica a nuestro sistema en un entorno favorable. Además de estar en un entorno con condiciones favorables, el sistema cuenta con características diferenciadoras como la facilidad de generación de trazas de movilidad o la posibilidad de configurar diversos parámetros de la simulación de red desde la aplicación, características que no incluyen las principales alternativas MOVE y TraNS.

La Tabla 23 muestra el presupuesto a entregar al cliente relativo a este proyecto de fin de carrera.

Concepto	Coste
Gastos de personal	11.588,85 €
Gastos de equipos	259,71 €
Gastos de software	151,09
Gastos de consumibles	86,10 €
Gastos de viaje y dietas	744,56 €
Gastos directos	12.830,31 €
Gastos indirectos	2.566,06 €.
Total Gastos sin riesgo	**15.396,37 €**
Riesgo (10%)	1.539,64 €
Total Gastos sin beneficios	**16.936,01 €**
Beneficios (15%)	2.540,40 €
Total gastos sin IVA	**19.476,41 €**
IVA (18%)	3.505,75 €
Total	**22.982,16 €**

Tabla 23 - Presupuesto para el cliente

3.4 Coste final y análisis de la desviación

En este apartado se muestra el coste final que ha tenido el proyecto y se compara con el coste estimado inicialmente. Debido a la desviación en el número de días requeridos para el desarrollo del proyecto cabe esperar una desviación en el presupuesto no sólo por los costes de personal si no también en las amortizaciones de los equipos utilizados.

La Tabla 24 muestra una comparativa entre el presupuesto inicial y el coste final del proyecto, indicando las variaciones en cada concepto y el total. Para el cálculo del coste total de las distintas partidas del proyecto se han tomado los mismos costes base modificando los meses de desarrollo del proyecto así como el número de horas hombre para calcular así el coste real del proyecto.

Concepto	Coste presupuestado	Coste real	Variación
Gastos directos	12.830,31 €	16.712,67 €	3.882,36 €
Gastos de personal	11.588,85 €	15.144,52 €	3.555,67 €
Gastos de equipos	259,71 €	339,39 €	79,68 €
Gastos de software	151,09	182,10 €	31,01 €
Gastos de consumibles	86,10 €	86,10 €	0,00 €
Gastos de viaje y dietas	744,56 €	960,56 €	216,00 €
Gastos indirectos	2.566,06 €.	3.342,53 €	776,47 €
Total	**15.396,37 €**	**20.055,20 €**	**4.658,83 €**

Tabla 24 - Coste real del proyecto comparado con el presupuestado

Como se puede apreciar existe una desviación muy importante en el coste de 4.658,83 €. Esta diferencia entre el coste presupuestado y el coste real no se puede traducir en un aumento del precio a cobrar al cliente por lo que se descontará del beneficio presupuestado que, como se puede ver en la Tabla 23, es de 2.540,40 €. Por lo tanto restamos a estos beneficios presupuestados la variación en los costes totales, lo que arroja unas pérdidas de 2.118,43 €. Restando a estas pérdidas el importe íntegro reservado para riesgos se sigue perdiendo dinero: 578,79 €.

En vistas de las desviaciones tan importantes sobre la planificación inicial cabían esperar estos resultados ya que una variación tan grande implica no solo más gastos de personal si no que aumenta prácticamente el resto de gastos directos e indirectos. Este tipo de situaciones se puede evitar con un margen de beneficios mayor que ejerza de colchón para los posibles sobrecostes ó aumentando el porcentaje de riesgo aplicado. En este caso en particular se debería de haber aumentado sensiblemente el porcentaje de riesgo ya que se trataba del primer proyecto de esta envergadura que se desarrollaba y gestionaba. Además, se conocían riesgos importantes como es la disponibilidad variable para el periodo de trabajo que debían de haber hecho.

El resultado negativo de este proyecto se debe principalmente a dos motivos: un sobrecoste elevado debido a una planificación incorrecta que ha sufrido importantes desviaciones y en segundo lugar una mala estimación de los riesgos del proyecto o un exceso de confianza a la hora de asignar el porcentaje de riesgo especificado en el presupuesto. Se ha calculado que con un riesgo presupuestado del 14% se abandonarían las pérdidas.

La experiencia enseña a detectar mejor este tipo de situaciones y a saber ponerles freno a tiempo.

Anexo 2: Manual de usuario

1. Introducción

VanSimFM (**VAN**ET **Sim**ulation **F**ramework **M**odeller) es una aplicación extensible de libre distribución y código abierto que permite la generación de experimentos de simulación de aplicaciones y protocolos VANET. Además permite la ejecución y de los experimentos creados así como la visualización de las trazas y el procesado mediante scripts AWK de las trazas generadas por la simulación.

Este manual pretende ser una referencia para los usuarios de VanSimFM en la generación y ejecución de experimentos de simulación.

2. Requisitos previos e instalación

VanSimFM ha sido desarrollada y probada íntegramente en un entorno Linux por lo que se debe contar con un sistema operativo Linux para su ejecución. El usuario deberá descomprimir en una carpeta de su elección el archivo comprimido con la aplicación y sus carpetas. Es muy importante que no se renombre, mueva o borre ninguna carpeta o archivo de los incluidos el en archivo comprimido.

Si bien VanSimFM no requiere de instalación en el sistema que se va a utilizar para su correcto funcionamiento el sistema debe contar con las siguiente aplicaciones instaladas, así como sus dependencias:

- NS-2 versión 2.35-RC6
- NAM versión 1.15-RC4

- SUMO versión 0.12.3
- Java versión 1.6.0_24
- Python versión 2.7

A pesar de haberse desarrollado y probado con estas versiones, VanSimFM puede funcionar con versiones anteriores de estas aplicaciones instaladas, si bien el equipo desarrollador no se hace responsable de posibles problemas que puedan surgir con versiones previas.

El entorno Linux en el que se ha desarrollado y probado la aplicación ha sido ejecutado mediante una máquina virtual por lo que VanSimFM es perfectamente compatible con máquinas virtuales. Las características de la máquina virtual son las siguientes, si bien dependiendo del sistema en el que se ejecute la máquina virtual y la aplicación de virtualización el funcionamiento puede ser distinto.

- Aplicación de virtualización: Sun VirtualBox versión 4.0.8 r71778
- Memoria asignada: 1.024 MB
- Memoria de vídeo: 12 MB
- Tamaño del disco: 8 GB
- Sistema operativo: Ubuntu versión 10.10

3. Ejecución y funcionamiento

La ejecución de VanSimFM comienza haciendo doble click en el archivo *VanSimFM.jar*.

La Figura 26 muestra la ventana principal que se muestra al usuario al iniciar la ejecución de la aplicación. En esta pantalla el usuario de la aplicación podrá elegir la acción que desea que ejecute la aplicación además de poder configurar el directorio donde se almacenarán y recuperarán los escenarios de simulación utilizados por la aplicación.

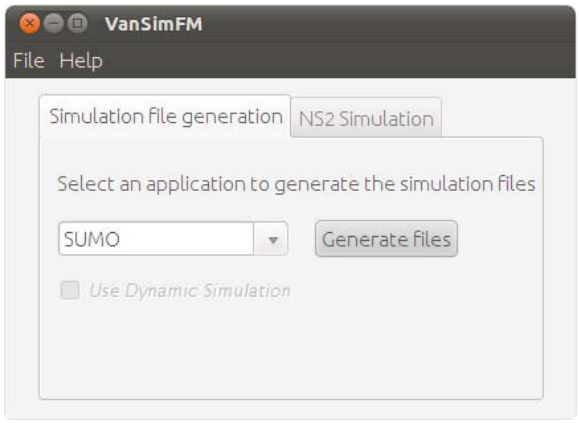

Figura 26 - Ventana principal de VanSimFM

3.1 Configuración del directorio de simulaciones

Por defecto VanSimFM recupera y almacena los ficheros de simulación que utiliza de una estructura de carpetas que tiene su origen en el directorio *simulations* situada en el mismo directorio que el archivo *VanSimFM.jar*. Si el usuario desea que VanSimFM recupere y almacene las simulaciones en una estructura de directorios almacenada en un directorio distinto deberá seleccionar, como muestra la Figura 27, la opción *File* del menú superior de la pantalla principal seleccionar la opción *Configuration* del desplegable que aparecerá.

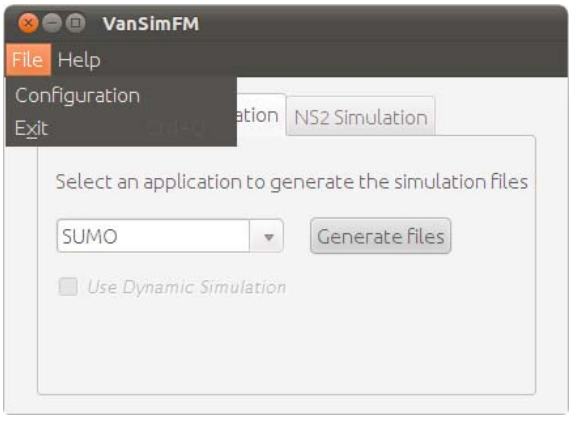

Figura 27 – Menú desplegable *File* en la ventana principal

Una vez seleccionada la opción *Configuration* la aplicación mostrará la ventana que muestra la Figura 28 en la que el usuario podrá cambiar el directorio de las simulaciones. Una vez el usuario haya definido el nuevo directorio deberá pulsar el botón *Save configuration* para que el cambio que ha realizado tengan efecto. En cualquier momento el usuario podrá cancelar la operación pulsando el botón *Cancel*.

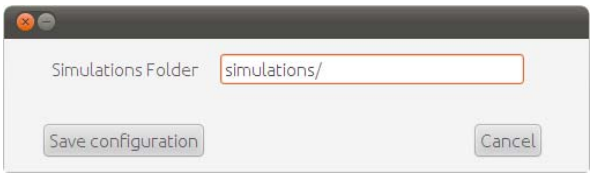

Figura 28 - Ventana de cambio de directorio de simulaciones

3.2 Creación de trazas de movilidad con SUMO

Para crear experimentos de simulación utilizando SUMO como origen de las trazas de movilidad el usuario deberá seleccionar SUMO en el desplegable mostrado en la ventana principal y pulsar el botón *Generate files*, tal y como muestra la Figura 29.

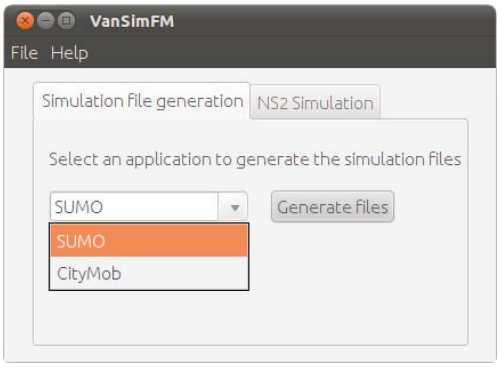

Figura 29 - Selección de SUMO para la generación de experimentos

La Figura 30 muestra la ventana de generación de trazas de movilidad específica de SUMO, mostrada tras pulsar el botón *Generate files*. En esta ventana el usuario podrá, en la primera pestaña, decidir si utiliza un entorno de simulación previamente generado, para ello deberá marcar, en la pestaña *Environment*, la opción *Select Existing Environment* y seleccionar el entorno creado previamente en el combo habilitado para ello. Una vez seleccionado el entorno deberá indicar si desea o no visualizar la simulación, marcando o no la opción *Visualize simulation* en la parte baja de la ventana. Junto a esta opción se encuentra el botón *Generate Traces*, que se pulsará para ejecutar la generación de las trazas de movilidad y su visualización si así se ha marcado.

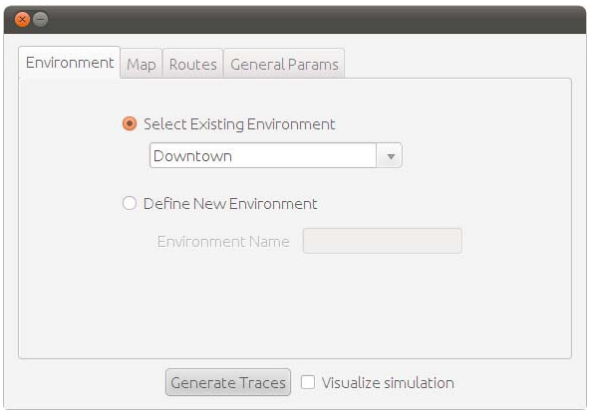

Figura 30 - Ventana de generación de trazas de movilidad con SUMO

En el caso de que se desee crear un nuevo entorno para generar las trazas de movilidad se deberá marcar, como muestra la Figura 31, la opción *Define New Environment*, esto hará que se habiliten las distintas opciones para la creación de escenario de el resto de pestañas de la ventana. Se deberá indicar en el campo de texto *Environment Name* un nombre para la nueva simulación.

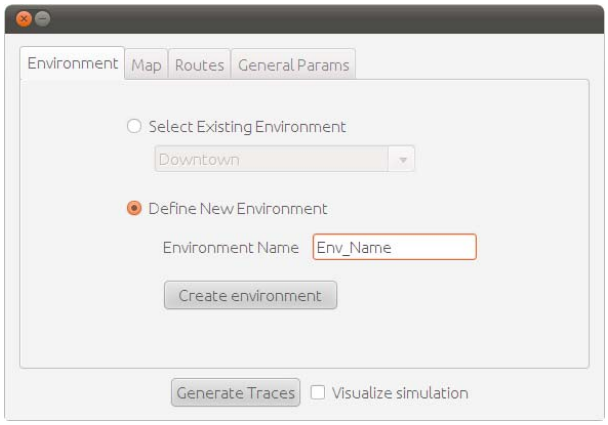

Figura 31 - Creación de un nuevo entorno de simulación

El primer paso para definir un nuevo entorno de simulación es especificar el escenario sobre el que se ejecutará. La Figura 32 muestra las distintas opciones que se ofrecen en la pestaña *Map* para la especificación del escenario, con la opción *Use Existing Map* marcada, con esta opción se utilizará un mapa ya existente para lo que se deberá pulsar el botón *Select map.net.xml file* y seleccionar el fichero con el escenario de SUMO en formato **.net.xml*.

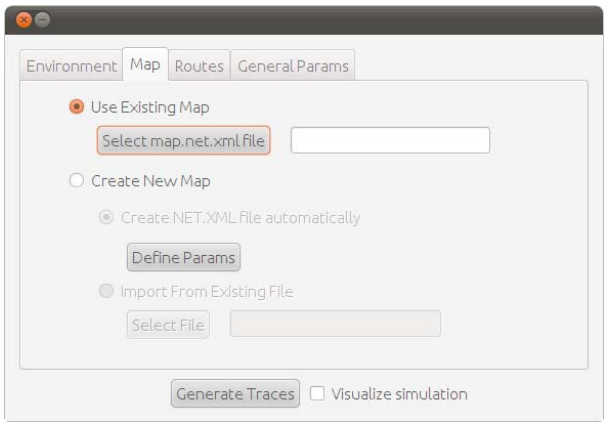

Figura 32 - Selección de un mapa ya existente

En caso de no disponer de un mapa para la simulación de SUMO se puede crear uno nuevo de forma automática marcando las opciones *Create New Map*, se habilitaran nuevas opciones en esta pestaña y después se deberá marcar la opción *Create NET.XML file automatically*, como muestra la Figura 33, tras lo cual se deberá pulsar el botón *Define Params*.

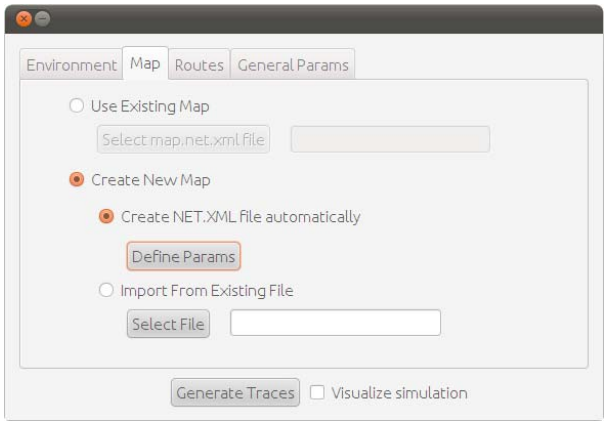

Figura 33 - Creación automática de un nuevo mapa

La Figura 34 muestra la ventana que mostrara la aplicación para la configuración del mapa que se creará automáticamente. En esta ventana el usuario podrá crear un mapa de manera totalmente aleatoria marcando la opción *Random map*.

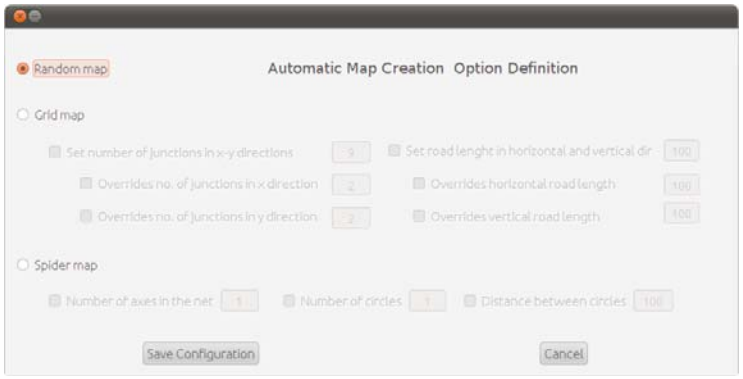

Figura 34 - Creación automática de mapa aleatorio

La ventana de configuración de creación automática de mapas permite también definir parámetros para la creación de mapas con forma de rejilla ó de tela de araña. Para esto el usuario deberá marcar, tal y como muestran la Figura 35 y la Figura 36, las opciones *Grip map* o *Spider map* para la creación de un mapa con forma de rejilla o de tela de araña respectivamente. Una vez seleccionado el tipo de mapa a crear automáticamente y especificados sus parámetros el usuario deberá pulsar el botón *Save Configuration* para que esta configuración tenga efecto. Se podrá cancelar la configuración en cualquier momento pulsando el botón *Cancel*.

Figura 35 - Creación automática de mapa con forma de rejilla

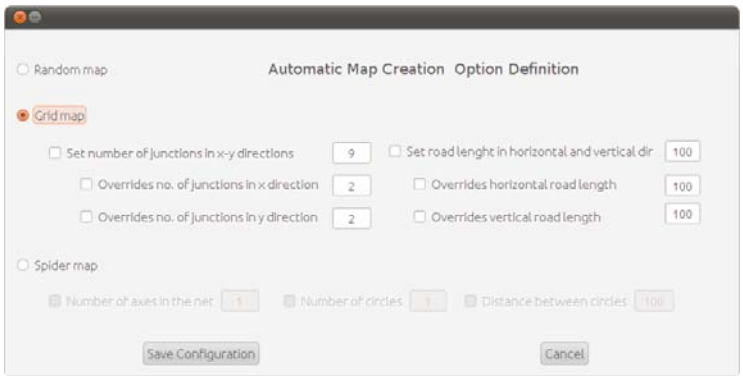

Figura 36 - Creación automática de mapa con forma de tela de araña

VanSimFM también ofrece la posibilidad de importar mapas para ser utilizados por SUMO en la generación de movilidad. En la versión actual sólo están soportados los mapas de Open Street Map. Para importar un mapa en formato OSM se deberán marcar, en la pestaña *Map* de la ventana de generación de trazas de movilidad las opciones *Create New Map* y *Import From Existing File*, como se muestra en la Figura 37, tras esto se deberá pulsar el botón *Select File* y seleccionar el fichero a importar.

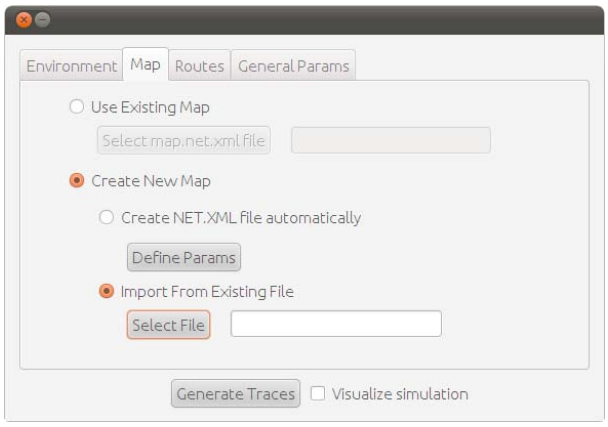

Figura 37 - Importación de un mapa existente

Una vez se ha configurado el escenario sobre el que se llevará a cabo la simulación de movilidad se debe definir la movilidad de los vehículos que intervendrán en la simulación. Para ello se puede especificar un fichero ya existente en formato **.rou.xml* marcando la opción *Use Existing Route File* y pulsando sobre el botón *Select map.rou.xml file* como se muestra en la Figura 38.

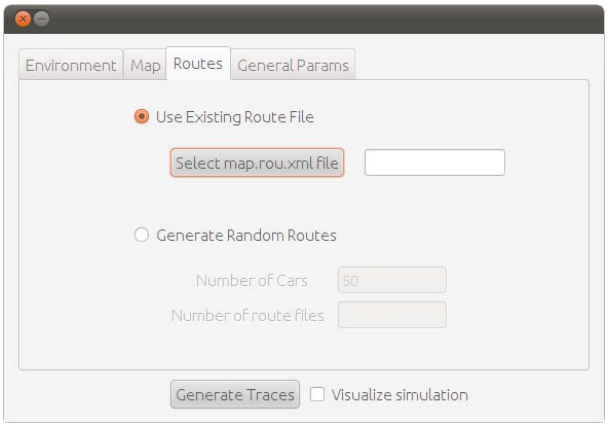

Figura 38 - Selección de fichero de movilidad existente

Una importante característica de VanSimFM es la posibilidad de crear rutas aleatorias para el mapa seleccionado, de esta forma se pueden crear archivos de movilidad sin necesidad de tener conocimientos de SUMO. La Figura 39 muestra las opciones para la creación de movilidades aleatorias que son el número de coches a generar, en el campo de texto *Number of Cars* y el número de ficheros de rutas con ese número de coches y para el mapa seleccionado que se van a crear, en el campod e texto *Number of route files*.

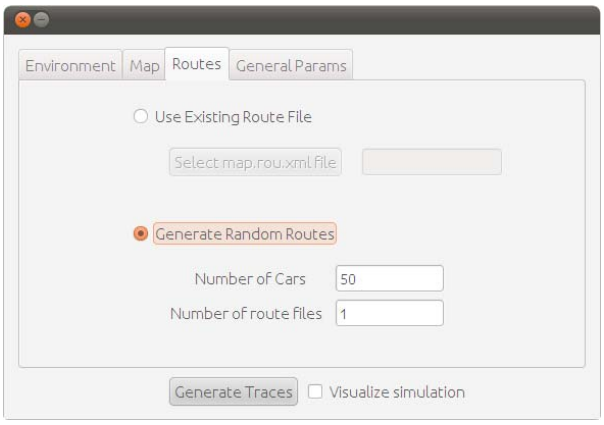

Figura 39 - Creación de rutas aleatorias

Por último se deberán definir los instantes inicial y final de la simulación, como muestra la Figura 40, en los campos de texto *Simulation Start* y *Simulation End* respectivamente en la pestaña *General Params*.

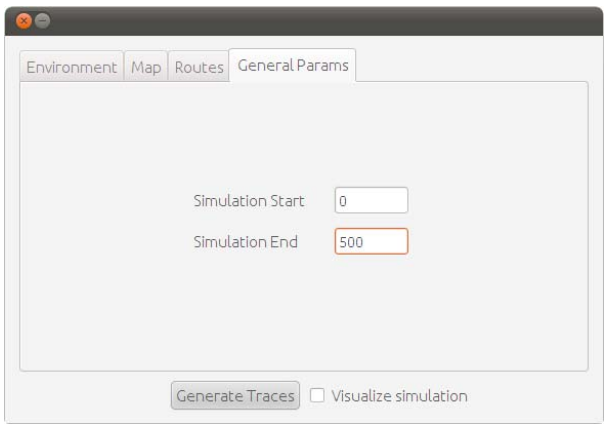

Figura 40 - Definición de instantes inicial y final de la simulación

Una vez se han definido todos los parámetros de la simulación de movilidad se deberá pulsar el botón *Create Environment* en la pestaña *Environment*, de esta forma se creará toda la estructura de archivos y directorios necesaria para ejecutar la simulación. Para esto se creará un directorio con el nombre dado al entorno y dentro de este se creará un directorio llamado *mobilitySimulation* en el que se guardarán los archivos creados.

Una vez creado el entorno este aparecerá en el menú desplegable de selección de entornos previamente creados pudiéndose seleccionar y ejecutar tal y como se ha explicado al principio de esta sección, pulsando el botón *Generate Traces*. Si se ha marcado la opción *Visualize simulation* se abrirá la ventana de SUMO mostrando la simulación que se ha seleccionado, tal y como muestra la Figura 41.

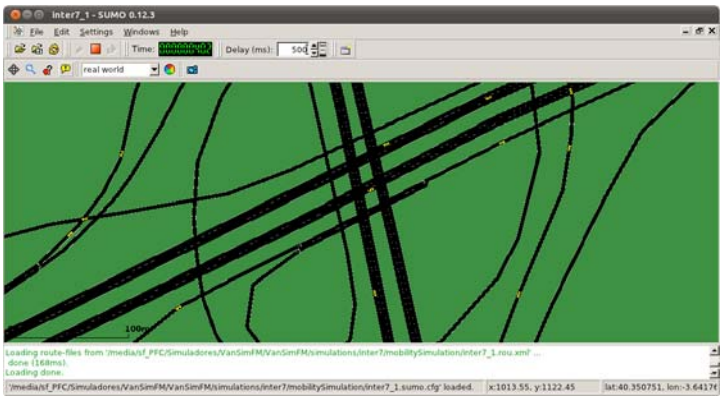

Figura 41 - Visualización de la simulación con SUMO

Se deberá cerrar la ventana de simulación de SUMO para pasar al siguiente paso del proceso de simulación, en caso de no haberse marcado esta opción se pasará directamente al siguiente paso: la configuración de la simulación de red del experimento.

3.3 Creación de trazas de movilidad con CityMob

Para crear experimentos de simulación utilizando CityMob como origen de las trazas de movilidad el usuario deberá seleccionar CityMob en el desplegable mostrado en la ventana principal y pulsar el botón *Generate files*, tal y como muestra la Figura 42.

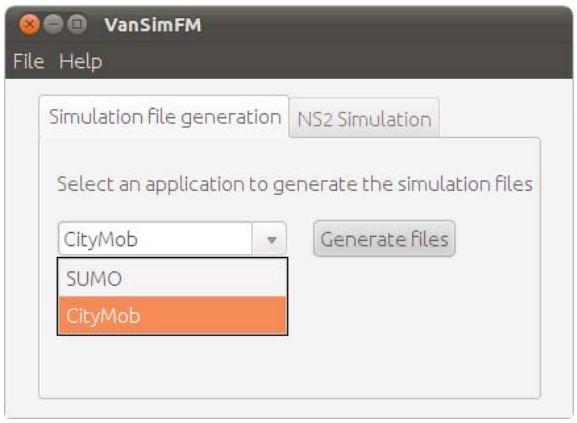

Figura 42 - Selección de CityMob para la generación de experimentos

A continuación la aplicación mostrará las ventanas de generación de trazas de movilidad con CityMob y la ventana de selección de trazas de movilidad creadas previamente.

La Figura 43 muestra la ventana de generación de trazas de movilidad con CityMob. Se trata de la interfaz de CityMob ligeramente modificada para adaptarla a su uso con VanSimFM. La principal modificación es la inclusión del campo de texto *Number of Simulations* donde se indicará el número de ficheros de trazas de movilidad que se generarán con CityMob. Para una información más detallada sobre los parámetros de configuración de la generación de trazas de movilidad se deberá pulsar en el menú superior *Help* la opción *Simulation* o consultar la documentación específica de la herramienta. El nombre del experimento, que es el que se usará para nombrar el directorio con los

ficheros generados, se indica en el campo *Output Folder*. Dentro de este directorio se creará un directorio llamado *mobilitySimulation* en el que se guardarán los archivos creados por CityMob con las trazas de movilidad.

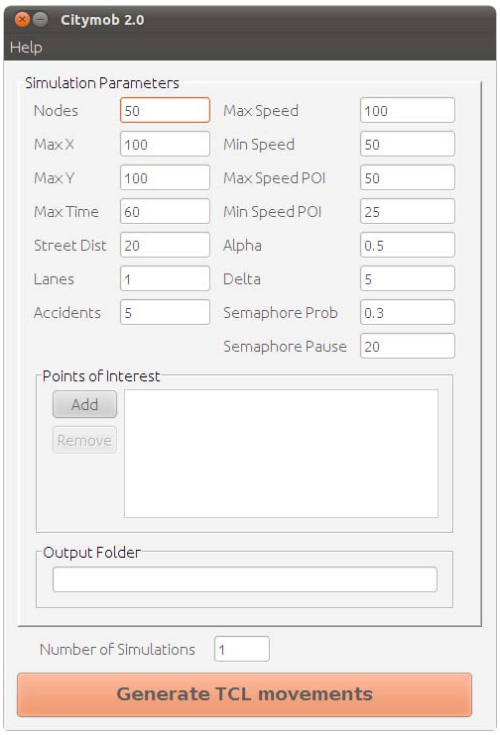

Figura 43 - Ventana de generación de trazas con CityMob

Una vez se han definido los parámetros de configuración, el número de trazas a a generar y el nombre del experimento se debe pulsar el botón *Generate TCL movements* momento en el que CityMob generará los ficheros con las trazas de movilidad.

Paralelamente a la ventana de generación de trazas con CityMob se muestra la ventana de selección de trazas de movilidad para la creación de experimentos. En esta ventana, tal y como se muestra en la Figura 44, se deberá seleccionar un fichero del directorio con las trazas de movilidad generadas por CityMob previamente y después pulsar el botón *Configure final TCL generation*. Se utilizarán para la generación de los ficheros finales todos los ficheros contenidos en la carpeta del fichero seleccionado. De esta forma si en la carpeta hay tres ficheros con trazas de movilidad al final del proceso se generarán tres ficheros con experimentos de simulación, cada uno usando la movilidad de uno de los tres ficheros de trazas de movilidad.

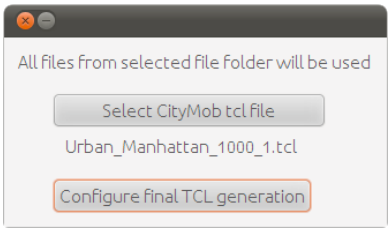

Figura 44 - Ventana de selección de trazas de movilidad

Tras pulsarse el botón *Configure final TCL generation* las ventanas de generación y selección de trazas de movilidad desaparecerán dando paso a la siguiente etapa de la generación de experimentos: la configuración de la simulación de red.

3.4 Configuración de la simulación de red

Una vez se ha definido la movilidad de los vehículos del experimento, tanto con CityMob como con SUMO, falta por definir la configuración de la simulación de red. Para esto se podrán configurar distintos aspectos de la simulación de red en las distintas pestañas de la ventana de configuración que aparecerá una vez terminada la definición de las trazas de movilidad. La Figura 45 muestra la pestaña *General Configuration* donde, marcando la opción *Edit default configuration* el usuario podrá modificar los valores por defecto que se encuentran especificados.

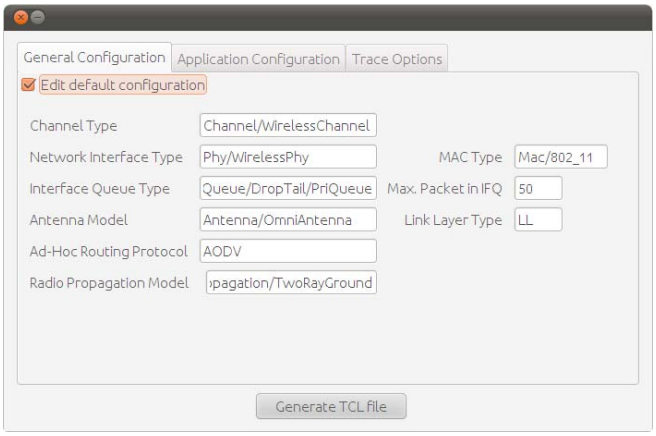

Figura 45 - Modificación de parámetros de simulación de red

Además de los parámetros básicos de configuración, en la pestaña *Application Configuration* se pueden definir distintos parámetros de configuración de las aplicaciones a simular en la simulación de red y que incorporarán los vehículos cuya movilidad se ha generado previamente.

La Figura 46 muestra la pestaña *Application selection* en donde se podrá indicar si se quiere que la aplicación genere automáticamente el código para la asignación de las aplicaciones epecificadas a vehículos y RSU. Para ello se debe marcar la opción *Auto generate vehicles and RSU attachment code* e indicar en el campo *Vehichle Application* la aplicación que se asignará a los vehículos así como el ratio de penetración de la misma en el campo *Penetration Ratio*. En el caso de querer incluirse RSU en la simulación se deberá marcar la opción *Use RSU* e indicar la aplicación que utilizarán las RSU en el campo *RSU Application*.

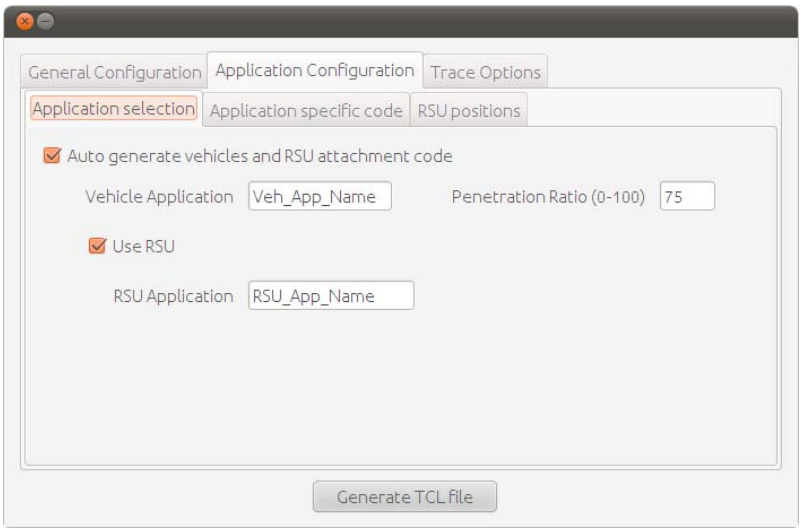

Figura 46 - Selección de aplicaciones para vehículos y RSU

Si se desea, la aplicación permite al usuario la inclusión de código directamente en los campos de texto de las pestañas *Initial code*, *Application functions* y *Final code* dentro de la pestaña *Application specific code*, como muestra la Figura 47. El código que se incluya en estos campos se escribirá directamente en los ficheros de los experimentos generados.

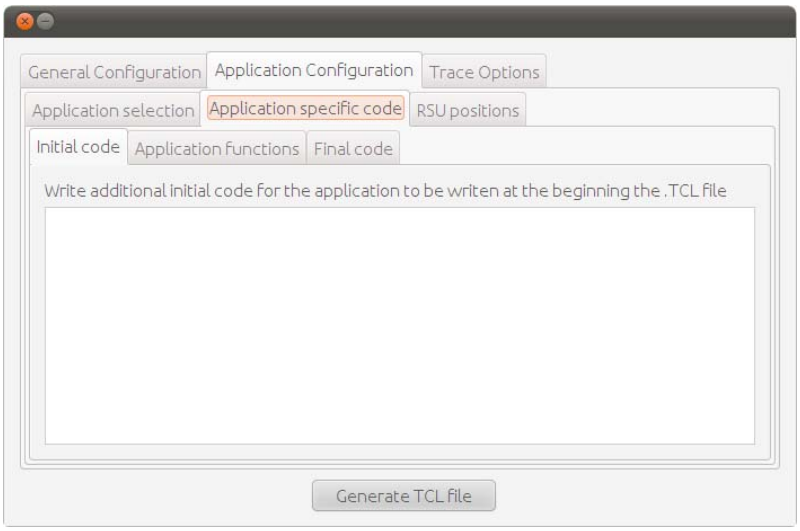

Figura 47 - Inclusión de código directamente en los archivos de los experimentos

La Figura 48 muestra la tabla de posiciones en la pestaña *RSU positions*, donde se deberán añadir manualmente pulsando el botón *Add RSU* e indicando después las coordenadas X e Y de cada RSU que se haya añadido a la simulación y a la que se adjuntará la aplicación indicada en el campo *RSU Application* mostrado en la Figura 46.

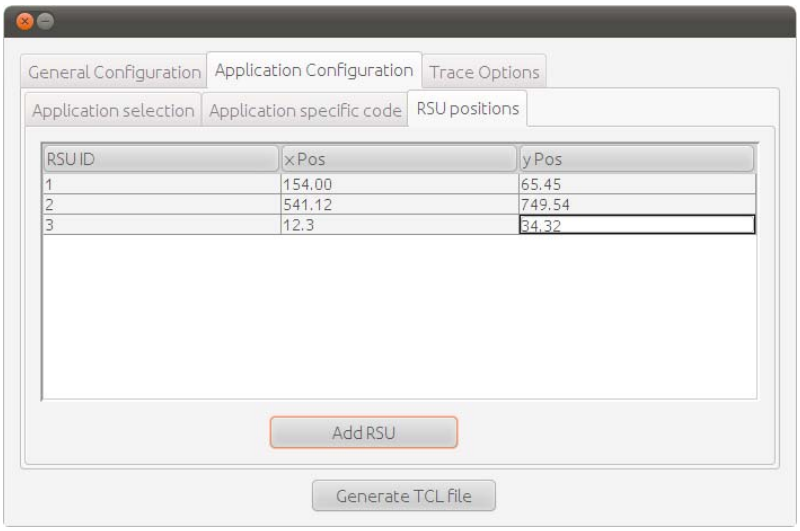

Figura 48 - Adición de RSU a la simulación

Por último, se podrán indicar distintas opciones de creación de trazas con los resultados de la simulación de red. La Figura 49 muestra la pestaña *Trace Options* en la que se podrán indicar las trazas que se desean generar marcando las distintas opciones disponibles. Además se podrá definir el nombre de los ficheros de trazas que se generarán tras la simulación de red.

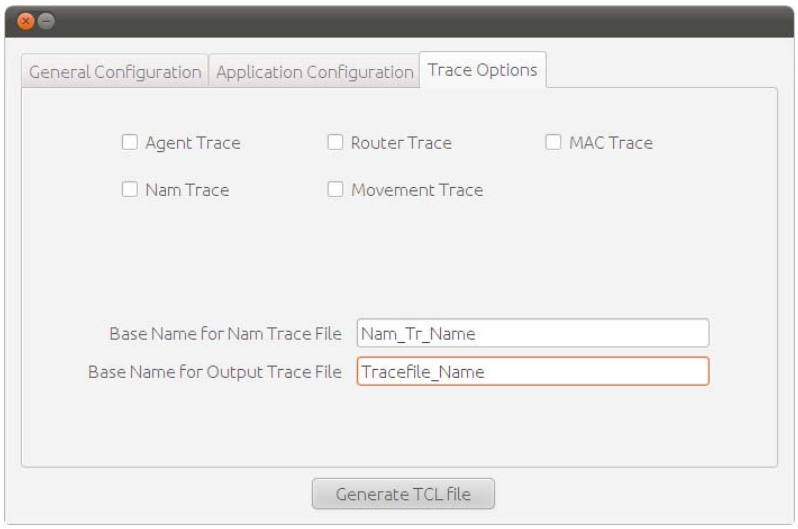

Figura 49 - Definición de opciones de generación de trazas

Una vez se han configurado todos las opciones disponibles en la ventana de configuración de la simulación de red se deberá pulsar el botón *Generate TCL file* situado en la parte baja de la ventana. Tras pulsar el botón VanSimFM generará en el directorio del experimento que se está creando un nuevo directorio llamado *networkSimulation* en el que se crearán tantos ficheros en formato *tcl* como trazas de movilidad se han generado en el paso previo de la creación del experimento.

Tras esto la ventana de configuración de red desaparecerá y se volverá a mostrar la ventana principal de la aplicación.

3.5 Ejecución de los experimentos

VanSimFM permite la ejecución de distintos experimentos de NS-2 creados previamente con la aplicación o de forma externa. La Figura 50 muestra la pestaña de ejecución de simulaciones de NS-2 en la ventana principal. La ejecución de estos experimentos se realiza invocando el simulador NS-2.

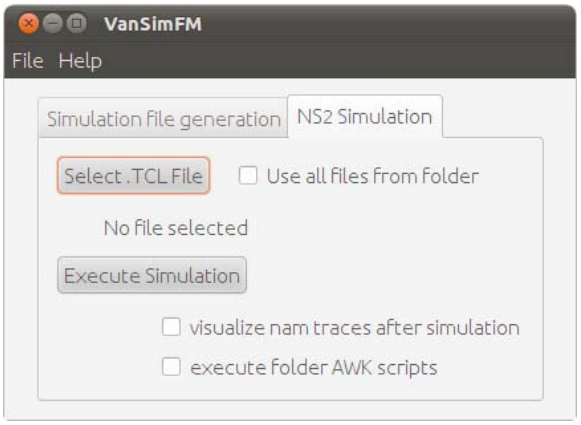

Figura 50 - Pestaña de ejecución de simulaciones

Para la ejecución de los experimentos se deberá seleccionar el fichero que se desea simular pulsando el botón *Select .TCL File*, la aplicación da al usuario la opción de que se simulen todos los ficheros *tcl* que se encuentren en el mismo directorio del archivo seleccionado, marcando la opción *Use all files from folder*, de forma que con pulsando un botón se puede ejecutar un gran número de simulaciones. Además de estas opciones, VanSimFM permite ejecutar el visualizador de trazas Nam tras la ejecución de las simulaciones, marcando la opción *visualize nam traces after simulation*, ó aplicar a las trazas generadas los scripts AWK que se encuentren en el mismo directorio, marcando la opción *execute folder AWK scripts*.

Una vez se ha configurado la ejecución de la simulación se deberá pulsar el botón *Execute Simulation* para comenzar la simulación.

4. Importar experimentos

VanSimFM permite al usuario utilizar experimentos generados en otra máquina con la aplicación. Para esto en primer lugar hay que comprender la estructura de directorios que utiliza la aplicación para almacenar los experimentos de simulación.

4.1 Estructura de directorios

VanSimFM accede y almacena todos los experimentos que crea dentro de una estructura de directorios que tiene como origen un directorio especificado por el usuario,

como se ha visto en la sección 3.1 del manual, y por defecto es el directorio *simulations* dentro del directorio donde se encuentra la aplicación.

Dentro de este directorio la aplicación crea una carpeta para cada experimento de simulación dentro de la cual almacenará de forma separada los ficheros de las simulaciones de red y de movilidad en los directorios *networkSimulation* y *mobilitySimulation* respectivamente.

4.2 Simulaciones de movilidad con SUMO

Para que la aplicación reconozca simulaciones de movilidad de SUMO creadas previamente con VanSimFM o con otra aplicación, esta deberá cumplir las siguientes características para aparecer en el desplegable de entornos ya creados mostrado en la Figura 30:

- Los ficheros deben encontrarse en el directorio *mobilitySimulation* dentro de un directorio con el nombre del experimento. La Figura 51 muestra un ejemplo de estructura de carpeta para un experimento llamado simTest
- Estos ficheros deberán de incluir un archivo *sumo.cfg* y además de los archivos que se encuentran referenciados en él.

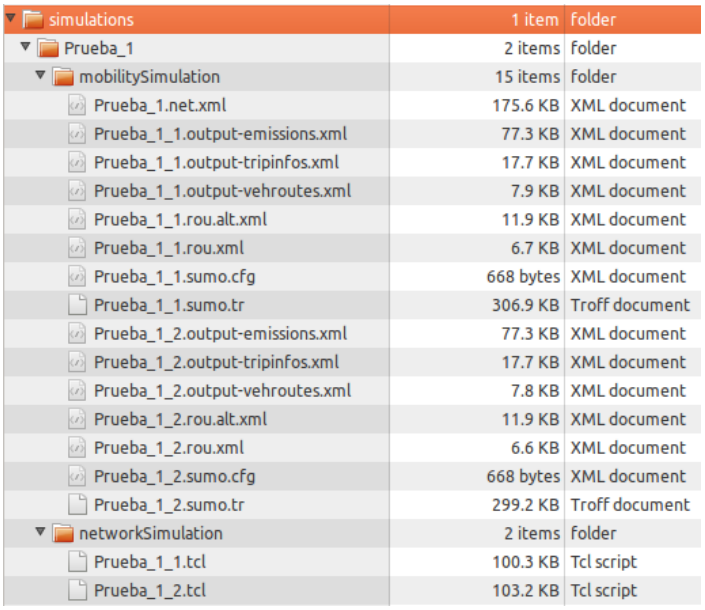

Figura 51 - Estructura de ficheros de un experimento de simulación

Como se puede ver en la Figura 51 el experimento se encuentra en una carpeta con su nombre, dentro de la carpeta *simulations*. Dentro de esta carpeta está la carpeta *mobilitySimulation* en donde SUMO o CityMob (SUMO en este caso) almacenan todos os ficheros que necesitan y que generan. Se pueden ver ficheros con el nombre del experimento pero con suficos "_1" y "_2", se trata de dos simulaciones de movilidad distintas para generar un experimento de simulación con dos simulaciones de red distintas sobre el mismo escenario. Mirando la carpeta *networkSimulation* se ven dos ficheros correspondientes a las dos simulaciones de red que forman el experimento Prueba_1.

4.3 Simulaciones de movilidad con CityMob

En el caso de CityMob los ficheros se almacenan de acuerdo a la estructura que hemos visto en la sección anterior, si bien para ser usados posteriormente por VanSimFM no tienen por qué estar dentro de esta estructura, aunque por supuesto es recomendable. Las única diferencia en lo que a estructura de ficheros se refiere entre un experimento hecho con SUMO y uno hecho con CityMob está en la carpeta *mobilitySimulation*, CityMob contendrá únicamente tantos ficheros como trazas se hayan creado al ejecutar CityMob, en la carpeta *networkSimulation* habrá de nuevo tantos ficheros de simulación de red como trazas se hayan generado previamente con Citymob para ese experimento.

Anexo 3: Plantillas

1. Plantilla para la definición de casos de uso

Identificador:	
Nombre:	
Autor:	
Descripción	
Actores:	
Precondiciones:	
Flujo Alternativo:	
Poscondiciones:	

Tabla 25 - Plantilla para la definición de casos de uso extendidos

2. Plantilla para la especificación de requisitos de software

Requisitos de Software				
Tipo:				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
Tipo:				
Id	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**

Tabla 26 - Plantilla para la especificación de requisitos de software

3. Plantilla para la especificación de pruebas de aceptación

Pruebas de aceptación			
Id	**Elementos probados**	**Entrada**	**Salida**

Tabla 27 - Plantilla para la especificación de pruebas de aceptación