Departamento de Investigación Operativa
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo – Uruguay

Proyecto de Grado

Ingeniería en Computación

Herramientas para la simulación del transporte público urbano colectivo

Informe Final

Autores

María Virginia López

Pablo César Lorenzo

Pablo Medina

Tutor

Dr. Antonio Mauttone

Resumen

En la actualidad, las ciudades conforman complejos centros urbanos donde se desarrollan un conjunto de actividades fundamentales para la vida y bienestar de sus habitantes. A medida que las ciudades se desarrollan y crecen, surgen nuevos desafíos para los sistemas básicos que se brindan a la población.

Uno de los sistemas más destacados es el sistema de transporte público urbano colectivo, el cual posibilita el movimiento de personas entre distintos puntos de la ciudad utilizando las infraestructuras viales existentes. Una correcta planificación del transporte público de una ciudad, es un aspecto vital ya que es uno de los tantos factores que inciden para determinar la calidad de vida de las personas y el grado de desarrollo de una sociedad.

La planificación de un sistema de transporte público urbano colectivo es una actividad que debe ser abordada cuidadosamente y comprende entre otras cosas determinar, en lo posible de forma óptima, el diseño de recorridos sujetos a frecuencias de pasadas de ciertas líneas en determinada franja horaria. Además debe considerar la asignación de personal y de flota.

Existen dos grandes participantes con distintos objetivos detrás de la planificación de transporte público: los proveedores de las flotas, los cuales brindan físicamente los componentes necesarios para brindar un servicio al público con determinada calidad, minimizando el costo operativo de un conjunto de variables como ser los buses, el combustible y el personal. Y las autoridades gubernamentales, gobiernos departamentales o intendencias, los cuales proveen una infraestructura para que el servicio de transporte garantice el funcionamiento de las flotas a un precio razonable.

Este proyecto se ubica en el contexto del desarrollo de una herramienta que permite planificar un sistema de transporte público urbano colectivo a través del diseño de un conjunto de recorridos sobre una red vial determinada (planificación estratégica) y definiendo frecuencias para los mismos a partir de un horario de servicio establecido (planificación táctica). La evaluación de dicha planificación se realiza mediante un proceso de simulación donde se estudia la interacción de los usuarios con un conjunto dado de líneas de buses, teniendo en cuenta información de la red vial y la demanda de transporte para un escenario dado.

Los objetivos de este proyecto son continuar y extender el desarrollo de una herramienta de planificación para brindar la incorporación y/o mejora de los siguientes componentes: interfaz de manipulación de datos geográficos, introduciéndose mejoras a nivel del detalle del modelo de red, permitiendo así una ubicación más detallada de las paradas en las calles; lógica del comportamiento de los pasajeros en el modelo de simulación, implementándose un nuevo módulo de selección de buses por parte de los pasajeros que permite viajes por trasbordos entre distintas líneas; y salida visual interactiva del simulador, diseñándose una interfaz gráfica correspondiente a la salida visual utilizando tecnologías actuales que permiten observar los detalles en el modelo de red vial. La validación de la implementación de los componentes diseñados se realizará utilizando diferentes casos de prueba.

Tabla de contenido

1. Introducción

1.1. Contexto de trabajo

Este proyecto se desarrolla en el grupo del Departamento de Investigación Operativa que estudia la aplicación de metodologías y herramientas de la Investigación Operativa a problemas de transporte público. Está relacionado a un proyecto CSIC en ejecución, una tesis de doctorado [1], proyectos de grado de la carrera Ingeniería en Computación finalizados [2] [3] y una tesis de maestría en curso.

IgoR-tp (Interfaz Gráfica para la Optimización de Recorridos de Transporte Público) es una herramienta de software que nace de sucesivas investigaciones sobre modelos, herramientas y algoritmos dentro del área de Investigación Operativa, para la planificación estratégica y táctica de recorridos y frecuencias para transporte público de pasajeros.

La primera versión, igoR-tp v1.0, desarrollada inicialmente por el proyecto de grado [2], dio origen a una herramienta de apoyo a la solución del problema de transporte de pasajeros para la ciudad de Rivera, utilizando una interfaz de usuario que manipulaba datos de demanda de transporte público e información geográfica relativa a recorridos. Basándose en la construcción de un modelo de red, un conjunto de recorridos sobre dicho modelo y una matriz de demanda de pasajeros, permite interactuar con distintos algoritmos para apoyar a la toma de decisiones de planificación del transporte de pasajeros.

La segunda versión, igoR-tp v2.0, desarrollada por el proyecto de grado [3] profundizó el nivel de detalle del modelo de red de la versión anterior y mediante la utilización de la metodología de Simulación a Eventos Discretos desarrolló un simulador. Este simulador modela la interacción de los pasajeros con los buses, dado un diseño del sistema de transporte público generado con igoR-tp. El simulador representa el comportamiento de los usuarios en un escenario dado por un conjunto de líneas y una demanda existente entre distintos puntos de una ciudad. De esta forma igoR-tp v2.0 junto al simulador, permite modelar diferentes comportamientos de pasajeros en la elección de líneas de transporte a partir de los datos de entrada.

La Figura 1 muestra en un alto nivel de agregación, las etapas que componen un ciclo típico de trabajo con todas las herramientas que serán presentadas a lo largo de este proyecto. A medida que se profundice sobre cada herramienta del proyecto, se complementará esta figura con un nuevo nivel de detalle.

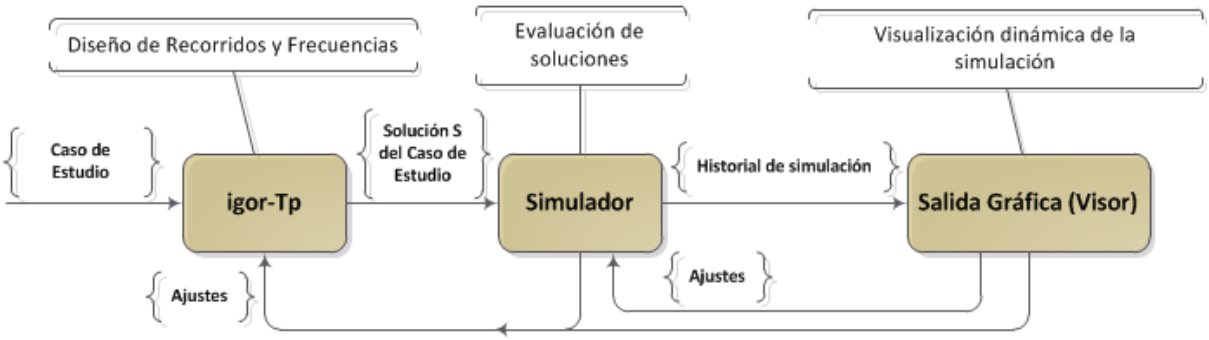

Figura 1: Ciclo de un caso de estudio

1.2.Objetivos

Basándose en lo implementado por los Proyectos de Grado de años anteriores [2] [3], el objetivo general de este proyecto es continuar la línea de trabajo previa para profundizar el área de conocimiento del departamento de Investigación Operativa sobre sistemas de transporte de pasajeros. Se deben extender las herramientas existentes de forma tal, que puedan ser utilizadas en casos de estudio de baja complejidad (en términos de tamaño) y por último validar la aplicación utilizando casos de estudio reales referentes a ciudades de tamaño pequeño como ser el caso de Rivera.

1.3.Resultados esperados

Los resultados esperados para el proyecto son los siguientes:

1. Una nueva versión de la herramienta igoR-tp, que incluya tratamiento detallado del trazado de los recorridos y la ubicación de las paradas
2. Implementar al menos un módulo para el simulador a eventos discretos, que modele diferentes comportamientos de la conducta de los pasajeros.
3. Implementar una nueva versión de la salida visual del simulador que incluya funcionalidades de zoom y visualización selectiva de los diferentes componentes del sistema de transporte público.
4. Verificar que los módulos desarrollados en el punto 3 concuerdan con los resultados de experimentación existentes del caso de estudio de la ciudad de Rivera [3].

1.4.Estructura del informe

El presente informe se encuentra estructurado de la siguiente manera: en el capítulo 2: *Marco Teórico,* se presentan los conceptos necesarios para la comprensión de los sistemas de transporte público urbano colectivo y su modelado. Luego en el capítulo 3: *IgoR-Tp,* se plantea la motivación para extender igoR-tp v2.0 a la nueva versión y se detallan los cambios realizados a cada módulo. El capítulo 4: *Simulador,* trata todo lo referente al simulador: marco teórico, diseño, diagramas de actividad y eventos. Se especifica cómo es la entrada y salida de datos, se describe el nuevo módulo de comportamiento de pasajeros implementado y se muestra tanto el resultado así como también el diseño del plan de pruebas. La nueva herramienta de visualización del simulador es referida en el capítulo 5: *Herramienta de visualización del Simulador,* donde se detallan los requerimientos (funcionales y no funcionales), la tecnología utilizada, así como su análisis, diseño e implementación. Posteriormente, se presentan las conclusiones y posibles extensiones del presente trabajo en el capítulo 0: *Conclusiones y trabajos futuros*.

Por otro lado, el informe cuenta con diferentes apéndices: uno donde se detalla la implementación de los cambios más importantes de igoR-tp (*Apéndice A*), uno donde se explican todos los cambios y problemas tecnológicos resueltos (*Apéndice B*), un manual de usuario de la herramienta de visualización (*Apéndice C*), un apéndice donde se relevan aplicaciones para simulación y tecnologías para visualización de mapas (*Apéndice D*) y un apéndice donde se encuentra material complementario para profundizar sobre el tema Simulación a Eventos Discretos (*Apéndice E*). El último (*Apéndice F*), describe como es la comunicación entre el simulador y la herramienta de visualización. Al final se encuentran todas las referencias bibliográficas utilizadas.

2. Marco teórico

El presente capítulo presenta en forma introductoria los conceptos necesarios para la comprensión de los sistemas de transporte público urbano colectivo y los modelos utilizados a lo largo de este proyecto.

2.1. Introducción

Un sistema de transporte convencional requiere de al menos un elemento de cada una de las siguientes categorías (siempre que esta combinación sea posible) interactuando entre sí [4]:

- **Infraestructura**: carretera, vías de tren, aeródromos, etc.
- **Vehículo**: automóvil, motocicleta, ómnibus, camión, avión, barco, etc.
- **Operador**: persona o sistema que guía un vehículo.
- **Usuarios**: cargas o personas.

Se puede considerar un quinto elemento que es el **servicio**, el cual permite que la actividad se realice en forma segura. Por ejemplo, en un sistema de transporte urbano, los semáforos ofrecen un servicio de ordenamiento del tráfico (capítulo 3 de [4]).

Los sistemas de transporte público urbano colectivo, de aquí en más TPUC, son un tipo particular de sistema de transporte convencional cuya finalidad es el transporte comercial de personas. En los TPUC intervienen tres actores: los usuarios (también referenciados como pasajeros), los operadores y las autoridades. Los usuarios son los destinatarios del servicio, para transportarse deben invertir tiempo y dinero. Los operadores (o empresas de transporte), son los que poseen los recursos económicos y físicos para ofrecer el servicio de transporte (flota de vehículos, mano de obra, mantenimiento y combustible). Las autoridades son entidades reguladoras, generalmente gubernamentales, que fijan ciertos componentes del sistema, como recorridos, tarifas y frecuencias.

El principal objetivo de un TPUC es ofrecer un servicio de calidad a los viajeros mediante el pago de una tarifa. Los TPUC cumplen una misión social pues permiten el acceso a la movilidad a toda la población, reducen la contaminación y la congestión de tráfico [5].

El principal problema que enfrentan los operadores de TPUC consiste en determinar cómo ofrecer servicios de buena calidad a los pasajeros mientras mantienen costos operativos razonables (salarios de la mano de obra, tamaño de flota, costos de mantenimientos y combustibles, instalaciones, etc.).

Según Ceder y Wilson (pág. 2 de [6]), la planificación de un sistema de TPUC implica determinar entre otras cosas: un plan de recorridos, frecuencias, horarios, asignación de personal y de flota; en lo posible óptimas. Este proceso se divide en las siguientes etapas bien definidas:

1. Diseño de las rutas: cantidad de líneas y el trazado de sus recorridos.
2. Determinación de frecuencias: de pasadas para cada línea, variable en el tiempo.
3. Determinación de horarios: tablas de horarios de cada línea y sincronización de despachos entre aquellas que comparten trasbordos.
4. Asignación de flota: en base a los vehículos disponibles para realizar los viajes.
5. Asignación de personal y recursos disponibles a los viajes programados por línea.

Las etapas 1 y 2 son llevadas a cabo por entidades reguladoras con posible participación de los operadores de servicio, mientras que las etapas 3, 4 y 5 son generalmente ejecutadas únicamente por los operadores de los servicios (empresas de transporte).

Dichos autores destacan la dificultad de resolver el proceso de forma global, por lo tanto la mejor aproximación está dada por la composición de las soluciones en cada etapa. Del mismo modo las soluciones obtenidas en las etapas 3, 4 y 5 están fuertemente condicionadas por las dos primeras.

Para resolver el problema de asignación de flota y de personal (etapas 4 y 5) se tienen en cuenta los objetivos de los operadores de servicio y se utilizan modelos de optimización para su solución. Se modelan como problemas de optimización combinatoria, programación lineal entera, y, en muchos casos, se resuelven en forma exacta.

Para el caso del diseño de rutas y frecuencias (etapas 1 y 2) las autoridades deben asegurar un servicio de calidad (seguro, confiable y rápido) a los usuarios, cumpliendo un rol social. Esto implica que se tengan en cuenta objetivos de usuarios y operadores, los cuales frecuentemente son contrapuestos. Este problema es conocido como el problema del diseño de rutas y frecuencias o TNDP (Transport Network Design Problem) y será definido más adelante.

Desaulniers y Hickman [7] realizan un estudio del estado del arte sobre los problemas que enfrentan los planificadores de TPUC, y observan que dichos problemas no se pueden tratar como un todo sino que se deben dividir en sub-problemas que se deben atacar secuencialmente, enmarcados en un proceso de planificación. Los autores dividen al proceso de planificación en tres grandes etapas según el alcance de la misma. Las etapas son:

- Planificación estratégica: es el proceso relacionado con la toma de decisiones a largo plazo, involucra el diseño de los recorridos sobre la red de vial y la evaluación de los mismos. Las herramientas desarrolladas a lo largo de este proyecto se encuentran enmarcadas mayormente dentro del plan estratégico. En esta etapa primero se diseña un trazado de recorridos que buscan satisfacer la demanda de pasajeros y luego se evalúa la calidad de dicha solución mediante un modelo de comportamiento de pasajeros. Se ubica en el contexto de la etapa 1 de la división de Ceder y Wilson.
- Planificación táctica: es el proceso cuyo objetivo es determinar las frecuencias de los recorridos y la generación de tablas de horarios a partir de estos últimos. La asignación de frecuencias generalmente tiene como objetivo maximizar el servicio a los pasajeros, para luego realizar ajustes a la realidad operacional de la empresa (por ej. disponibilidad de unidades en ese período de tiempo). El plan táctico incluye la toma de decisiones en períodos de tiempo de duración media y suele aplicarse en distintos intervalos de tiempo de forma de ajustar los servicios a la demanda real en cierta temporada [5]. Se ubica en el contexto de las etapas 2 y 3 de la división de Ceder y Wilson.
- Plan operacional: es el proceso que tiene como objetivo minimizar los costos operativos del proveedor del servicio de transporte. Dentro de esta etapa se enrolan todas las actividades relacionadas: asignaciones de vehículos a las rutas y a los estacionamientos, de fechas para el mantenimiento de las unidades y de choferes a las mismas [5]. Involucra decisiones que se

toman una vez por mes o por día, por ejemplo: asignación de buses a recorridos y asignación de choferes a los buses. Se ubica en el contexto de las etapas 4 y 5 de la división de Ceder y Wilson.

2.2. Optimización de Recorridos y Frecuencias

El problema de diseño de recorridos y frecuencias óptimos para el transporte público, conocido como TNDP, básicamente consiste en determinar un conjunto de recorridos y sus frecuencias, utilizando información geográfica de la ciudad de estudio (donde se trazan los recorridos) y la demanda de pasajeros.

Todo problema de optimización de recorridos y frecuencias define un objetivo. Muchos de los estudios realizados en el área, detallados en [6], utilizan por lo general dos objetivos: objetivos de usuarios y de operadores. De esta forma, se busca optimizar de forma simultánea los objetivos de los usuarios y de los operadores con restricciones sobre la disponibilidad de servicio y la satisfacción de la demanda, la red vial subyacente y el comportamiento de los pasajeros.

Una forma sencilla de expresar el problema del TNDP es la siguiente:

$$\min f\{R, F\}$$

Donde *R* es el conjunto de recorridos sobre la red vial y *F* es la frecuencia de los buses que recorren los mismos expresadas en salidas por unidad de tiempo. La función *f*, representa los objetivos de usuarios y de operadores.

Cualquier solución *S* al problema consiste en un conjunto de recorridos con frecuencias asociadas:

$$S = \{Rs, Fs\}$$

Donde *Rs* es el conjunto de recorridos que respeta las restricciones de *R* y *Fs* es el conjunto de los valores de las frecuencias que respeta las restricciones de *F*. Según [7], dentro del plan estratégico se observan dos sub-actividades independientes, estas son:

- *Diseño de red*: El trazado de recorridos y la determinación de las frecuencias asociadas, se realizan sobre una red para tal fin. La finalidad del diseño de dicha red, es que la misma sea el contexto de definición de los recorridos donde se calculan las funciones objetivo de usuarios y operadores, para una solución particular del TNDP. Como entrada de esta sub-actividad se dispone además de una matriz de demanda la cual refleja las necesidades de transporte de los pasajeros entre diferentes puntos de la ciudad.
- *Asignación de pasajeros:* La asignación de pasajeros es la actividad que modela el comportamiento de los pasajeros (dados mediante la matriz de demanda), respecto a un conjunto de recorridos y frecuencias. El estudio del comportamiento de los pasajeros se define más adelante en la sección 2.3.4: *Modelos de Comportamiento*.

Según [7] la definición de los recorridos está condicionada por la demanda de viajes de los pasajeros, logrando que el problema de diseño de red esté relacionado con el de asignación de pasajeros, el cual a su vez depende de los recorridos definidos. De esta aseveración se deduce que, al momento de diseñar una solución al TNDP, se debe tener especial atención a la información provista por la demanda y su comportamiento.

Dificultades en la resolución del TNDP

El TNDP se modela como un problema de optimización combinatoria de alta complejidad, por lo tanto no puede ser resuelto de forma exacta mediante algoritmos de complejidad polinómica. Esto implica que deben emplearse técnicas heurísticas que refinan las soluciones de cada paso de forma iterativa en tiempos razonables [5]. Baaj y Mahmassani enumeran algunas dificultades para el problema del TNDP [6]:

1. Formulación del Problema: para definir las variables de decisión (en particular la elección de la línea por parte del pasajero) y la función objetivo.
2. No linealidad del problema.
3. Naturaleza combinatoria del problema con variables discretas.
4. Múltiples objetivos: debido al conflicto entre interesados (usuarios y operadores), no existe una solución óptima que optimice cada objetivo en forma individual, sino varias soluciones no dominadas[1].

Por lo general, las soluciones al diseño de recorridos, se basan en un amplio conocimiento del problema por parte de los planificadores en base a experiencia utilizando guías y recomendaciones [5].

De todas formas, debido a que el TNDP es un problema de optimización sobre redes con un componente de comportamiento de pasajeros, existen un conjunto de modelos que son ampliamente utilizados para su estudio.

[1] Una solución es no dominada, cuando no existe otra solución que mejore la función objetivo sin perjudicar al resto.

2.3.Modelos

Cada modelo de optimización del TNDP contempla intereses de los usuarios y de los operadores en base a ciertas restricciones. Como se comentó, estos objetivos son generalmente contrapuestos, una mejora en uno implica un deterioro del otro. Por lo tanto, la importancia de los componentes de la función objetivo es una decisión política y se define por las entidades reguladoras del sistema.

El objetivo de los usuarios del transporte es viajar desde su origen a su destino de forma confortable, segura y rápida. Debido a que generalmente las tarifas no son aplicadas en estos modelos, la función objetivo a optimizar (minimizar) es el tiempo de viaje y de espera en parada.

Por otro lado, el principal objetivo de los operadores es brindar un servicio que les sea beneficioso. Esto se puede traducir como transportar la mayor cantidad de pasajeros posibles utilizando el menor costo de operaciones para sustentar el servicio.

Las entidades reguladoras o autoridades tienen el papel de controlar el transporte asegurando que el servicio sea provisto en un costo razonable, tercerizando el servicio a los operadores. La decisión más importante que recae sobre las autoridades es la determinación de un compromiso entre los costos de usuarios y de los operadores del sistema, actuando sobre las variables de decisión del TNDP (en este caso, trazados de recorridos y frecuencias) [5].

2.3.1. Modelo de optimización

En el contexto del TNDP se busca minimizar simultáneamente las funciones objetivo de usuarios y operadores, *fU* y *fO* respectivamente. El modelo de optimización de recorridos y frecuencias para el TNDP se expresa como:

$$\min(fU, fO)$$

Sujeto a restricciones de:

- La red vial, que sirve de infraestructura donde se diseñan los recorridos.
- El comportamiento de los pasajeros en el sistema.
- La frecuencia factible (toda frecuencia es mayor o igual que la mínima permitida).
- Factor de carga (ninguna ruta puede saturarse).
- La cantidad de buses operando, está acotada por el tamaño de flota.

El comportamiento de los usuarios también debe estar reflejado en el modelo de optimización, por lo tanto se asocia un modelo de asignación que indica qué hipótesis se asumen para dicho comportamiento. La función objetivo de los usuarios es definida y evaluada por el modelo de asignación.

2.3.2. Modelo de red

La red donde se diseñan los recorridos se modela sobre un grafo de nodos y aristas. Dicho grafo es de la forma:

$$G = (N; A)$$

Donde *N* es el conjunto de nodos y *A* es el conjunto de aristas que conectan pares de nodos. Los elementos del modelo de red pueden tener distintos significados según la representación con la realidad que se desee modelar.

Los nodos de *N* pueden representar intersecciones o cruces de calles, pero también pueden representar centroides (zonas geográficas concentradas lógicamente en un punto sin correspondiente geográfico real) o paradas de buses.

Los arcos de *A*, de igual manera, pueden representar tramos de calles de la red vial que conectan nodos generando intersecciones. Una secuencia de nodos y arcos del grafo, representa un recorrido por donde transitan los buses.

La representación del grafo no es única, está condicionada por el volumen de datos a utilizar y el nivel de detalle a considerar.

El modelo de red considerado en este proyecto es el mismo que se definió en el proyecto de grado [3], el cual es un modelo extendido donde se destacan los siguientes conceptos:

- Centroide: Elemento ficticio que no se corresponde con ningún punto real y concentra la demanda de una zona. Es el destino y origen del tráfico de pasajeros.
- Nodo vial: Una red vial incluye calles e intersecciones, dichos cruces se denominan nodos viales. En los nodos viales los buses pueden cambiar de dirección.
- Tramo de calle: Representa la conexión entre cada par de nodos viales.
- Parada: Es un elemento del transporte público donde los usuarios del sistema de transporte suben y bajan de los buses.
- Caminata: Las caminatas modelan una conexión sobre la cual los peatones se trasladan desde el centroide de una zona a una parada y viceversa, o desde una parada hacia otra las cuales pueden estar en la misma o en distintas zonas.

La Figura 2 se corresponde con un modelo de red.

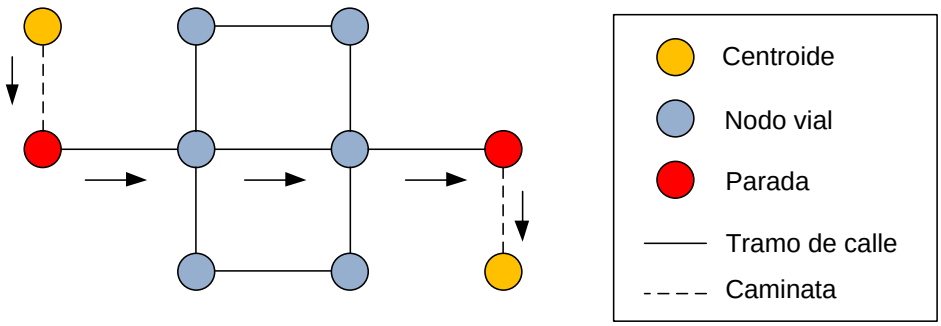

Figura 2: Modelo de red

Evidentemente, el desempeño y tiempo de respuesta de las soluciones por parte de los algoritmos se verá afectada a medida que el nivel de detalle del modelo crezca. Pero en contra partida, modelos con pobre nivel de detalle podrían llevar a soluciones muy simples y con limitada correspondencia con la realidad.

2.3.3. Modelo de demanda

La demanda de viajes expresa la necesidad de movimientos de pasajeros entre distintas zonas de un caso de estudio. La demanda se modela como una matriz origen-destino *D*= $\{d_{ij}\}$ donde cada entrada *ij* expresa la necesidad de viajes de los pasajeros entre la zona *i* y la zona *j*. Todo valor d_{ij} de la matriz se expresa en viajes por unidad de tiempo en una ventana horaria determinada.

Si el modelo de red vial consta de centroides de zona y tramos que los comunican, es posible vincular los elementos de la matriz demanda y red vial identificando cada valor d_{ij} con los centroides de las zonas *i* y *j*.

2.3.4. Modelos de comportamiento

Este tipo de modelos indican qué hipótesis se asumen para el comportamiento de los pasajeros respecto a una solución al problema del TNDP.

Un pasajero en un sistema de transporte se enfrenta permanentemente a un problema de toma de decisiones, debido a que para un par origen-destino puede haber más de un recorrido. El pasajero pretenderá seleccionar el recorrido que maximice su función objetivo (tiempo de viaje mínimo o costo total de viaje mínimo). El caso más general de recorridos múltiples entre origen y destino ha llevado al desarrollo del concepto de “estrategia”, reflejando posibles reglas de abordaje que el pasajero puede usar para completar su viaje entre el par origen-destino. El modelo de comportamiento, define criterios de distribución de la demanda de pasajeros sobre los recorridos de la solución del problema.

A continuación se describen dos tipos de modelos de comportamiento utilizados por este proyecto de grado: el modelo de asignación y el modelo de simulación.

Modelo de asignación

El modelo de asignación de pasajeros a recorridos, de ahora en más modelo de asignación, es una herramienta útil en el TNDP y está definido en la mayoría de los modelos de optimización. Es un componente clave debido a que su desempeño incide en la eficiencia global de los algoritmos de optimización.

Su finalidad es distribuir la demanda de viajes sobre los recorridos, para evaluar la función objetivo de los usuarios sobre una solución particular del TNDP (dato de entrada). Para ello, la mayoría de los modelos de asignación modela distintos criterios de selección de recorridos por parte de los usuarios, bajo la hipótesis de minimizar la función objetivo de los mismos. Una vez aplicado el modelo de asignación, los valores como los tiempos de viaje (*tv*) y tiempos de espera (*te*) para una determinada solución *S* serán conocidos.

Según la complejidad del modelo, algunos pueden considerar tiempos constantes de viaje sobre los tramos de los recorridos o que los mismos dependan del flujo de pasajeros. También muchos modelos consideran capacidades de los vehículos.

Los modelos de asignación generalmente se plantean como problemas de optimización, que reflejan la conducta de los pasajeros que buscan minimizar su tiempo de viaje. Dichos modelos se resuelven utilizando técnicas de optimización y devuelven resultados que indican tiempos promedio de viaje y ocupaciones promedio de las líneas.

Modelo de simulación

La Simulación a Eventos Discretos (SED) es el área dentro de la Investigación Operativa encargada de construir y estudiar modelos abstractos que representan sistemas de la vida real que evolucionan en el tiempo, cambiando su estado en determinados instantes específicos.

En sistemas complejos, como el sistema de transporte urbano de pasajeros, donde existen muchas interacciones internas y un alto uso de recursos importantes (buses por ejemplo), es posible utilizar un modelo de SED.

Utilizando la exploración de distintos escenarios, el modelo de simulación describe los aspectos relevantes del sistema mediante una serie de relaciones y sentencias lógicas que se escriben en un lenguaje de programación (programa) [8]. El objetivo del modelo de simulación en este contexto de trabajo, es estudiar un sistema de transporte público de pasajeros mediante:

- Evaluación de la función objetivo de los usuarios: Al igual que el modelo de asignación, el modelo de simulación (en este contexto) distribuye la demanda sobre los recorridos de la solución. La selección de criterios de recorridos por parte de los usuarios se codifica mediante programación buscando obtener diferentes comportamientos de selección que sean fácilmente intercambiables.
- Evaluación de la función objetivo de los operadores del servicio: el objetivo de los operadores de servicio también puede representarse en el modelo de simulación.

La ventaja de utilizar un modelo de simulación reside en la capacidad de explorar aspectos en los cuales el modelo de asignación es limitado. Por ejemplo, es posible codificar diferentes funciones objetivo basadas en distintas hipótesis de comportamiento y evaluar el uso de cada una. Por su característica dinámica, un modelo de SED además permite observar los estados intermedios por los que pasa el sistema para arribar a los mismos resultados que se obtienen con un modelo de asignación. Esto contribuye a un mejor entendimiento de las causas que llevan a obtener determinados resultados en un modelo de asignación. Otras características que poseen los modelos de simulación son: generación de múltiples reportes, estudio de largo de colas y uso de recursos; y en lo posible, representación visual detallada del sistema y sus participantes.

3. IgoR-tp

3.1.Introducción

La aplicación igoR-tp v2.0 es una herramienta que surge en el marco de un proyecto de investigación (PDT 48/02, Planificación de líneas de ómnibus en el transporte público urbano colectivo, Julio 2006 - Junio 2008.), para luego ser desarrollada y mantenida por estudiantes de Proyecto de Grado de la carrera de Ingeniería en Computación de la Facultad de Ingeniería [2] [3].

Su principal objetivo es diseñar, construir, editar y evaluar grafos del modelo de red vial relacionados al problema TNDP. Para diseñar y construir dichos grafos, igoR-tp se vale de la tecnología *MapWinGis* y archivos de tipo shapefile[2]. Al utilizar esta tecnología, cada concepto del modelo de red (ver sección 2.3.2 *Modelo de red*) se representa mediante una capa a la que se le asocia un archivo shapefile.

Las relaciones de los elementos que componen el modelo de red vial y las capas que maneja igoR-tp se presentan en la Tabla 1.

Nombre Capa	Tipo	Concepto del modelo de red que representa
Aristas	Polilínea	Tramo de calle
Nodos Viales	Punto	Nodo Vial
Paradas	Punto	Parada
Centroides	Punto	Centroide
Caminatas	Polilínea	Caminata

Tabla 1: Capas de igoR-tp vs conceptos del TNDP

Mediante la manipulación de estas capas, igoR-tp permite trabajar con los elementos del modelo de red vial en alto nivel generando como resultado una representación de grafo del modelo de red.

A partir del grafo generado se pueden diseñar soluciones al problema TNDP, las cuales pueden ser generadas y evaluadas con algoritmos de optimización de recorridos y frecuencias, que implementan modelos para la planificación estratégica. Es por esto que igoR-tp se puede considerar como una herramienta de diseño y evaluación de soluciones del problema TNDP.

La Figura 3 ilustra cómo la herramienta igoR-tp trabaja mediante capas con los conceptos de red vial.

[2] Shapefile: Formato vectorial de almacenamiento digital desarrollado por la empresa *ESRI*. Estos tipos de archivos son los que maneja IgoR-tp en todas sus versiones.

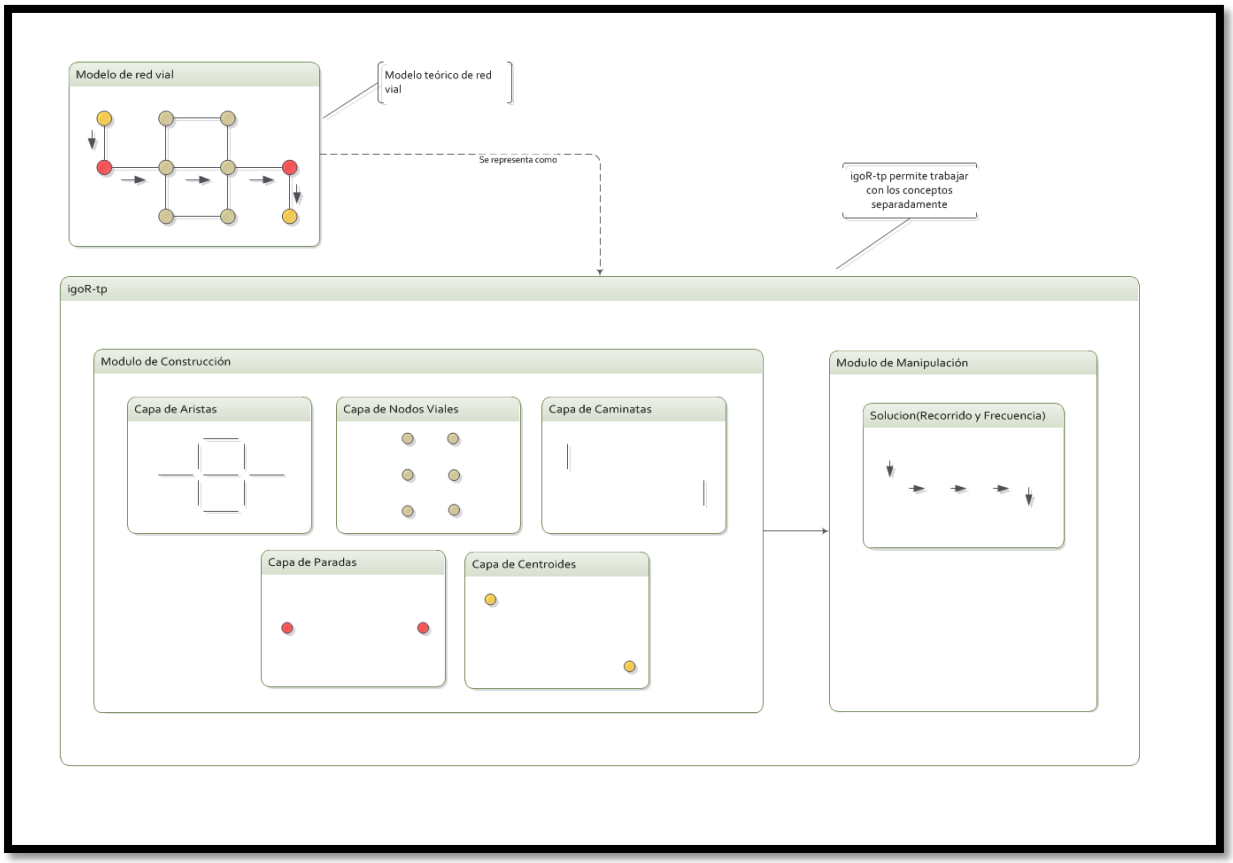

Figura 3: Representación interna de un modelo de red vial en igoR-tp

La herramienta igoR-tp v2.0 se compone de tres módulos, los mismos se conocen como *módulo de construcción*, *módulo de manipulación* y *módulo de algoritmos*. A continuación se presenta una breve descripción de las principales funcionalidades de cada módulo.

3.1.1. Módulo de Construcción

El módulo de construcción permite crear, almacenar y editar un *modelo de red vial* como un conjunto de capas. Para crear un nuevo *modelo de red vial* es necesario disponer como entrada, de una capa de puntos de demanda y de una capa de red vial en forma de segmentos de ejes de calle de la ciudad a modelar (se entiende por eje de una calle a la línea que divide longitudinalmente su área en dos partes simétricas). Con esta entrada el módulo crea un modelo que contiene tres capas, una copia de las dos de entrada más una tercera inferida de las primeras, esta tercera capa se llama capa de nodos viales. Estas tres capas no son editables. En la etapa de creación se definen valores que serán constantes asociadas al modelo, estas constantes son: *velocidad media de flota*, *velocidad media de un peatón* y forma de medir distancias (euclídea y Manhattan) [pág. 27 [3]].

Una vez creado el modelo de red vial, es posible añadir y modificar nuevas capas con zonas, paradas y caminatas. El usuario puede añadir paradas a la capa de paradas con la restricción de que deben coincidir con la posición de un nodo vial. La capa de zonas está formada por polígonos, la herramienta no permite que ningún polígono de zona quede contenido dentro de otro. Cada polígono de zona genera automáticamente un centroide que se añade a la capa de centroides. El centroide generado se añade en el baricentro del polígono, esta ubicación no es al azar, se basa en la hipótesis de que la demanda está uniformemente distribuida en el área de la zona.

Este módulo ofrece funcionalidades visuales para facilitar el trabajo con distintas capas y permite modificar estilo, color y ancho de las polilíneas que representan los ejes de calle de la red vial. También permite editar el estilo, color y tamaño de los puntos que representan nodos viales, paradas, puntos de demanda y centroides. Es posible modificar color y transparencia de la capa de zonas. Todas las capas que componen el modelo se pueden ocultar.

Otra funcionalidad de este módulo es la generación de matrices de demanda entre zonas o, lo que es equivalente, entre centroides. Es posible generar varias matrices para el mismo modelo, lo que permite representar la demanda en distintas franjas horarias. Estas matrices forman parte del modelo y se persisten con él. En la sección 3.5.1 se puede encontrar una descripción detallada del proceso de generación de la matriz de demanda.

3.1.2. Módulo de Manipulación

El módulo de manipulación permite crear, almacenar y editar distintas soluciones con diseños de recorridos de buses sobre un modelo de red vial. Se define un recorrido como una sucesión de nodos viales y se utiliza para definir las rutas de las distintas líneas de ómnibus.

Un modelo creado en este módulo se denomina *modelo de manipulación*. Para crear un modelo de manipulación, es necesario disponer de un *modelo de red* creado con el módulo de construcción con al menos una matriz de demanda. Sobre este modelo se pueden añadir y editar distintas soluciones con diseños de recorridos. Un modelo de manipulación sólo puede tener una solución de recorridos activa a la vez para trabajar.

Un recorrido se define como secuencia de tramos (segmentos de ejes de calle) adyacentes. Los recorridos pueden ser abiertos o cerrados, pero no pueden tener huecos. Los recorridos tienen asociado un sentido, esto implica que el modelo no considera recorridos con circulación de buses en ambos sentidos, si el usuario desea modelar esta realidad, es necesario definir dos recorridos superpuestos con sentidos opuestos.

Este módulo, al igual que el anterior, cuenta con facilidades visuales para simplificar la tarea del usuario, permite editar estilo, color y ancho de las polilíneas que representan los recorridos y la red vial, así como también editar el estilo, color y tamaño de los puntos que representan nodos viales y paradas.

Este módulo, además de persistir el modelo de manipulación en formatos binarios estándar para representación de mapas, también mantiene un archivo paralelo llamado *grafo.txt,* el cual almacena la red vial (información detallada del archivo *grafo.txt* se puede encontrar en la sección 3.5.2). También se persiste la solución activa para que pueda ser evaluada con algoritmos externos. Dichos algoritmos pueden ser invocados directamente desde este módulo. Para invocar algoritmos es necesario que la forma de comunicación con el mismo esté definida en el *módulo de algoritmos*.

3.1.3. Módulo de Algoritmos

El módulo de algoritmos permite gestionar algoritmos externos a igoR-tp, éstos pueden ser de dos tipos:

- Algoritmos de evaluación: son algoritmos descriptivos que permiten obtener métricas para la comparación de soluciones del TNDP.
- Algoritmos de optimización: son algoritmos heurísticos que retornan un conjunto de soluciones evaluadas bajo algún criterio de optimización.

Si bien igoR-tp está diseñado para trabajar con los dos tipos de algoritmos, en este proyecto solamente se trabaja con algoritmos de evaluación. Dichos algoritmos son almacenados en archivos ejecutables y tienen distintas entradas indicadas como parámetros de línea de comando. Estos parámetros son dependientes de la naturaleza del algoritmo.

El módulo de algoritmos está diseñado para gestionar la interfaz de comunicación de dichos algoritmos. Permite asociar nombres a los ejecutables de los algoritmos y almacena los parámetros del algoritmo. Luego de registrado un algoritmo, es posible desde el módulo de manipulación, invocarlo mediante el nombre asociado.

A forma de resumen, se presenta la Figura 4 que ilustra la secuencia de actividades que se realizan en igoR-tp, especificando en qué módulo se debe realizar cada una y las dependencias entre ellas.

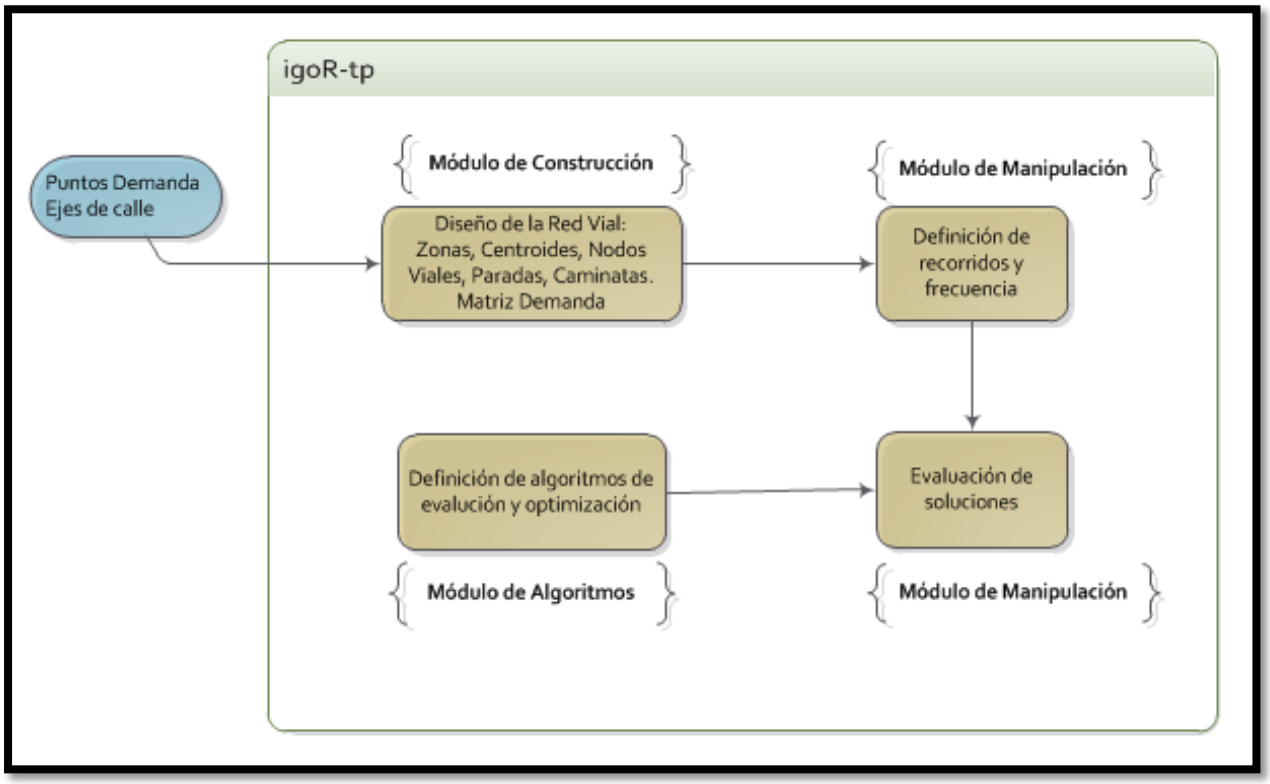

Figura 4: Flujo de actividades igoR-tp

3.2. Motivación para extender igoR-tp 2.0

La principal motivación para extender a la nueva versión, surge debido a que se requiere un mayor detalle en el manejo de las paradas de ómnibus en el modelo. El concepto de parada, como "sitio donde los pasajeros abordan y descienden de los ómnibus" sigue siendo el mismo, pero se introducen cambios en la representación.

En la versión 2.0, la red vial se representa por los ejes de calles y sus cruces (nodos viales). Una parada es un nodo vial y por lo tanto está ubicado en un cruce de calles. Si bien es una buena aproximación al problema, no es del todo precisa, ya que al estar las paradas en los cruces de calles, éstas tienen cuatro sentidos o direcciones posibles desde donde pueden arribar los buses (como muestra la Figura 5).

La nueva versión de igoR-tp, permite que las paradas se ubiquen en cualquier sector de la calle, salvo en los cruces, y los cuatro sentidos posibles se reducen solamente a uno (Figura 6). De esta forma se logra un modelo de parada que representa exactamente la realidad: una parada de ómnibus está ubicada de forma tal que todos los buses que paran allí circulan en el mismo sentido.

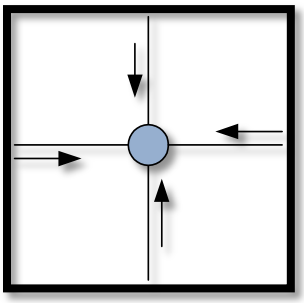

Figura 5: Parada con cuatro sentidos posibles

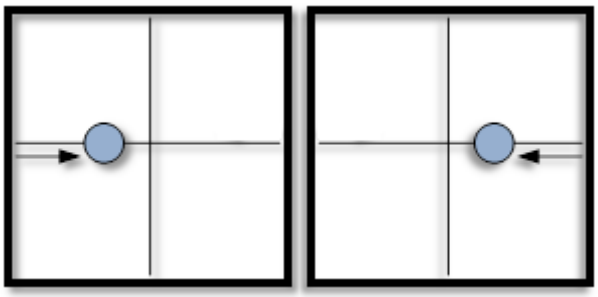

Figura 6: Parada con un único sentido

3.3.Requisitos funcionales de igoR-tp 3.0

En esta sección se describen brevemente las principales funcionalidades de los tres módulos que componen igoR-tp, detallando los cambios realizados en cada uno para soportar la nueva representación de parada mencionada anteriormente.

Las tablas presentadas a continuación resumen los cambios realizados en cada módulo.

Módulo Construcción
Se modifica la funcionalidad: agregar parada.
Se modifica la funcionalidad: eliminar parada.
Se agrega la funcionalidad: definir sentido de una parada.
Se agrega la funcionalidad: información de shape.

Tabla 2: Modificaciones en el módulo construcción

Módulo Manipulación
Se eliminan los recorridos no dirigidos.
Se modifica la validación de recorridos.
Se agrega la funcionalidad: carga externa de matrices de demanda.
Se modifica la forma de seleccionar los parámetros de tipo archivo para una evaluación.
Modificaciones en los archivos de persistencia.

Tabla 3: Modificaciones en el módulo de manipulación

Módulo Algoritmos
Se agrega un nuevo tipo de parámetro de entrada.

Tabla 4: Modificaciones en el módulo algoritmos

En las siguientes secciones se detalla cada uno de estos cambios.

3.3.1. Módulo de Construcción

Descripción

Utilizando el módulo de construcción, se diseña el grafo que representa la red de transporte público. Permite definir en detalle todos los componentes del modelo de red vial. Para ello, el módulo necesita un conjunto de datos de entrada dados por:

1. La red vial sobre la cual se desea representar (shapefile de ejes de calles).
2. La información sobre la demanda de los viajes (shapefile de puntos de demanda).
3. La velocidad media de los buses y peatones.

Estos datos se procesan y se genera el grafo que representa la red física del modelo de red vial, es aquí que aparecen los conceptos aristas y nodos viales.

Como se mencionó anteriormente, en igoR-tp la información geográfica se almacena en un conjunto de capas, donde cada capa contiene un tipo de elemento.

La Tabla 5 describe la información asociada a cada una de las capas del módulo Construcción.

Nombre Capa	Datos
Aristas	Nombre de calle, nodo vial inicio (from), nodo vial fin (to), costo y largo.
Nodos Viales	Identificador del nodo.
Paradas	Identificador del nodo vial correspondiente.
Zonas	Identificador de zona.
Centroides	Nodo vial próximo (cruce más cercano).
Caminatas	Identificador de centroide, identificador de Parada1, Identificador de Parada2 y costo. Dependiendo del tipo de conexión.
PtoDemanda	Identificador de la zona a la que pertenece.

Tabla 5: Datos de los elementos dentro del módulo de construcción

Además de la manipulación de los conceptos vistos, el módulo de construcción permite editar información sobre: velocidades medias de la flota y peatón, factor de conversión de unidades a metros, tipo de distancia que utilizan los pasajeros para desplazarse y el tipo de unidad de la capa de calles.

Cambios funcionales

Esta versión de igoR-tp maneja los mismos elementos que la versión antecesora, pero como se ha mencionando previamente, presenta cambios en referencia a la representación de las paradas.

El primer elemento afectado por este cambio es la capa de nodos viales. En la versión anterior representaban únicamente cruces de calles y era sobre éstos que se podían definir las paradas. Ahora los cruces siguen representándose como un nodo vial, pero las paradas no se podrán ubicar allí. Al utilizarse los nodos viales para definir los tramos de los recorridos de los buses, es necesario que las paradas sean nodos viales para que los buses pasen por ellas. Por lo tanto, las paradas serán un subconjunto de aquellos nodos viales que no sean cruces de calles.

El segundo elemento que sufre cambios es obviamente la capa de paradas. Para pasar de una representación de cuatro sentidos a una con uno solo, no basta con ubicar las paradas fuera de los cruces de calles. El representar las calles por sus ejes, crea una indeterminación en cuanto al sentido de las paradas. La misma se correspondería en la realidad con decidir en qué acera de la calle se ubica exactamente la parada, lo que haría que quedara perfectamente determinado el sentido de los buses que podrían parar allí.

De lo anterior surgen los siguientes cambios:

1. Los nodos viales ya no serán fijos y calculados una única vez por proyecto. Sino que se deben actualizar cada vez que se agrega o quita una parada.

2. La funcionalidad llamada “Marcar/Desmarcar paradas”, que se utilizaba para agregar o quitar un nodo vial del conjunto de paradas, debe convertirse en “Agregar/Eliminar paradas”.
3. Se debe agregar la funcionalidad de asignar un único sentido a la parada.

A continuación se detallan las funcionalidades: agregar y eliminar una parada de la nueva versión, donde estos cambios fueron implementados.

Agregar Parada

Con la funcionalidad “Agregar Parada” el usuario puede crear una nueva parada en cualquier punto de una calle, exceptuando las esquinas, y definir el sentido de los buses que paren allí.

Para esto debe seleccionar la funcionalidad, hacer clic en cualquier punto de un eje de calles y a continuación se marcarán en verde dos puntos, para que el usuario mediante un menú emergente, seleccione el sentido de la nueva parada. El nodo vial origen de la calle tendrá el sentido A, mientras que el nodo vial destino de la calle tendrá el sentido B, tal como se muestra en la Figura 7.

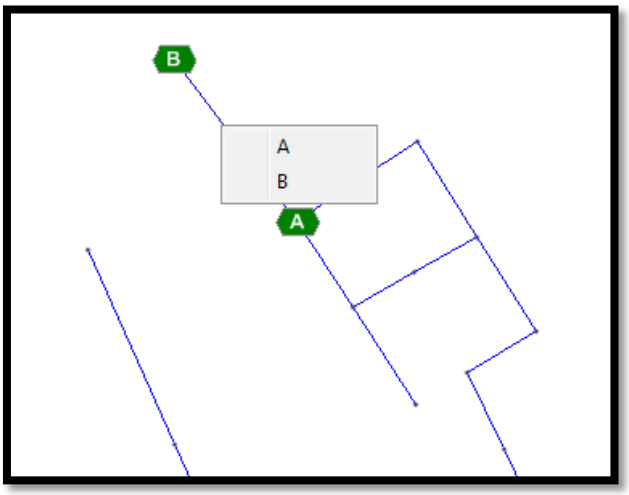

Figura 7: Agregar parada

De seleccionar como sentido el extremo A, significará que en esa parada podrán detenerse únicamente los buses que circulen en dirección B-A. Caso contrario, seleccionando B, pararán los buses que circulan en dirección A-B.

Una vez seleccionado el sentido de la parada desde el menú emergente, se creará una nueva parada en el proyecto Construcción. El crear una nueva parada realiza varias acciones: en primer lugar se crea un nodo vial en la posición de la parada y se agrega éste a la capa nodos viales. Esto no significa que en esa posición se defina un nuevo cruce de calles, sin embargo se agrega a esta capa por compatibilidad con el modelo de la versión anterior que supone que toda parada es un nodo vial. Luego de creado el nodo vial se agrega la parada a la capa de paradas, guardando la información sobre el nodo vial que determina el sentido de la misma (origen y destino de la arista). Por último se deben actualizar las aristas. Hay que componer las aristas que se generan al introducir un nuevo nodo vial. Para esto se crean dos nuevas aristas que son el resultado de dividir la arista original por un punto de corte dado por la nueva parada. Finalmente se elimina la arista original. (En el *Apéndice*

A se puede encontrar información sobre la implementación de esta funcionalidad). Los datos de las nuevas aristas quedan de la siguiente manera:

	Nodo origen	Nodo destino	Largo	Costo
Arista Nueva1	Nodo origen de la arista original	Nodo vial correspondiente a la nueva parada	Calculado	Calculado
Arista Nueva2	Nodo vial correspondiente a la nueva parada	Nodo destino de la arista original	Calculado	Calculado

Tabla 6: Datos de las nuevas aristas

Visualmente la funcionalidad "Agregar Parada" requiere que, mientras se está agregando una parada en el proyecto, se deberá contar siempre con la visibilidad de la capa de nodos viales, ya que las "etiquetas" que muestran los extremos de la calle están asociadas a los nodos viales origen y destino. Por lo tanto, el usuario siempre tendrá habilitada la capa de nodos viales mientras esté agregando paradas y podrá deshabilitar dicha capa (si así lo desea) una vez que terminó el proceso, pero no antes. La Figura 8 es un ejemplo de dicha situación:

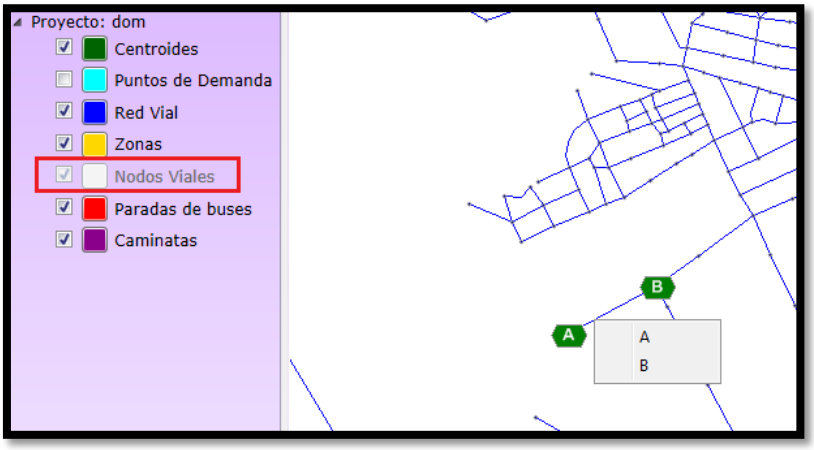

Figura 8: Visibilidad de capa Nodos Viales

Eliminar Parada

En la versión anterior las paradas no se eliminaban propiamente, en vez de esto se podía hacer que un nodo vial dejara de ser parada. En esta versión sí es necesario eliminar la parada, por lo tanto, se agregó la funcionalidad "Eliminar Parada". Como se muestra en la Figura 9, se modificó la interfaz agregando un nuevo ícono para que se sea posible borrar una parada.

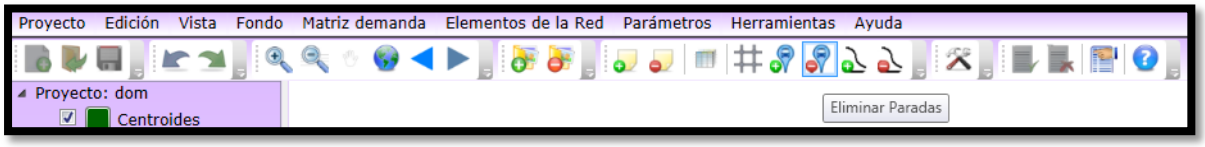

Figura 9: Borrar parada desde la interfaz

Luego de seleccionar la funcionalidad indicada, el usuario lo único que debe hacer es un clic sobre la parada que desea eliminar.

Si se agrega y luego quita una parada, es de esperar que el estado de la red vial se vea inalterado. Para esto entonces, al eliminar una parada no solo se debe quitar la misma de la capa de paradas sino que también se debe eliminar el nodo vial correspondiente y restablecer las aristas al estado original. Recordemos que cuando se agrega una parada, se crean dos aristas nuevas que sustituyen en el modelo a la arista que contiene el punto donde se colocó la nueva parada.

El borrado de una parada realiza las siguientes acciones:

1. Elimina las dos aristas divididas por la parada y restablece la arista original.
2. Elimina de la capa de nodos viales el nodo correspondiente a la parada.
3. Elimina la parada de la capa paradas.

Como consecuencia de los cambios introducidos, las funcionalidades agregar parada y borrar parada resultaron relativamente complejas en cuanto a lógica y procesamiento. Es por esto que se decidió para esta nueva versión deshabilitar la funcionalidad *Undo* y *Redo* para estas opciones.

Sentido de una parada

Como se explicó anteriormente, para que el sentido de una parada quede completamente determinado, se debe seleccionar un extremo de la arista donde se va a ubicar la parada (Figura 8).

A continuación, se analizará por medio de un ejemplo la situación que se presentaba cuando se agregaba más de una parada en una misma arista. Supongamos que se desea agregar una parada en la arista determinada por los nodos A (idNodo = 1) y B (idNodo=2) (Figura 10).

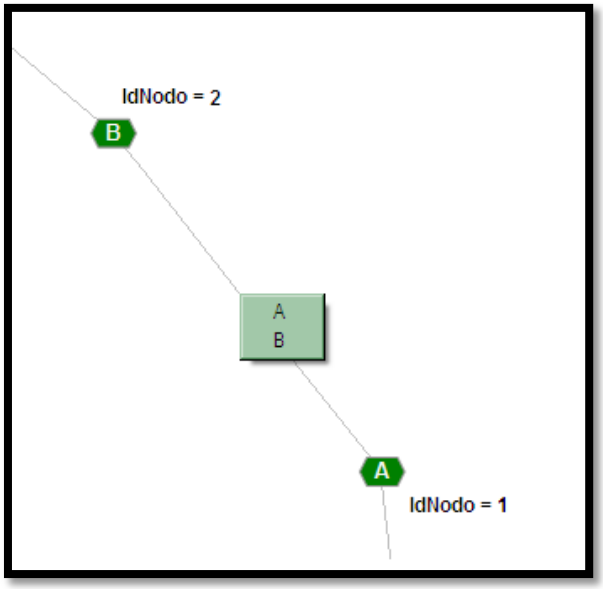

Figura 10: Ejemplo agregar parada

La tabla asociada a la arista (Tabla 7) tiene la siguiente información:

IdArista	fromNodo	toNodo	largo(m)	costo(min)
1	1	2	300	1,874

Tabla 7: Información arista

Luego de seleccionar el punto de la arista donde se quiere ubicar la parada, se debe elegir uno de los dos extremos para determinar su sentido.

Supongamos que se elige el extremo A, a continuación se agregará la parada resultando las aristas como se muestran en la Figura 11 y los datos asociados de la siguiente manera (Tabla 8):

IdArista	fromNodo	toNodo	Largo(m)	costo(min)
1	1	3	170	1,054
2	3	2	130	0,82

Tabla 8: Información nuevas aristas

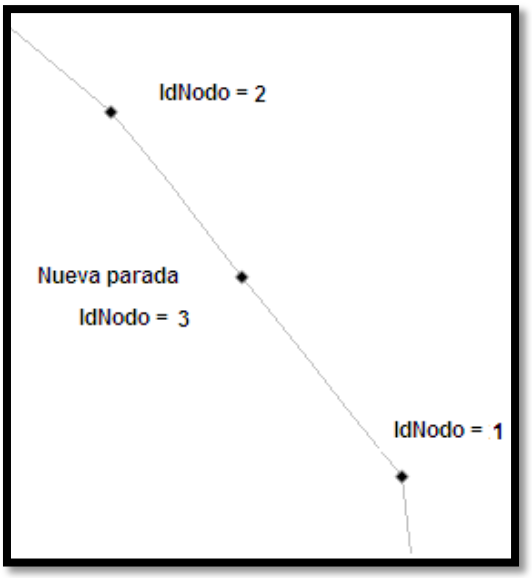

Figura 11: Ejemplo nueva parada

En la información asociada a la parada es donde se guarda el sentido de la misma de la siguiente manera (Tabla 9):

Id_Parada	Id_Destino
3	1

Tabla 9: Información de la nueva parada

Supongamos ahora que se desea agregar una nueva parada entre los nodos 2 y 3, por otra parte se selecciona el nodo 3 para determinar su sentido. Luego se crea otra parada entre esta última y la parada 3 (Figura 12).

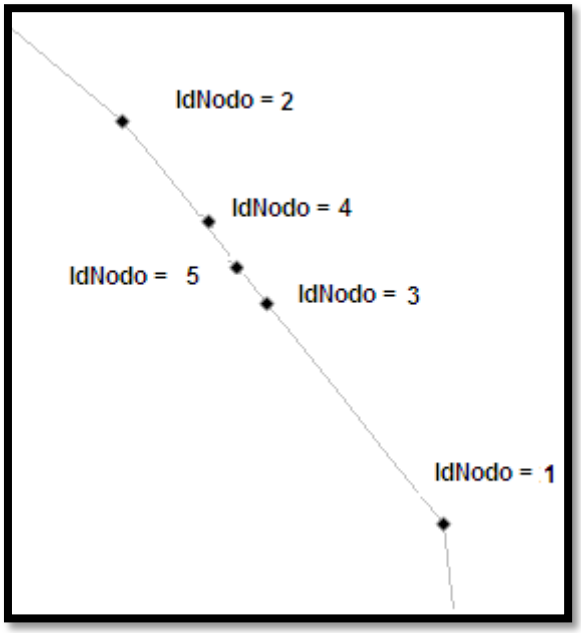

Figura 12: Se ingresan dos nuevas paradas

La información de las aristas y paradas entonces se observa en Tabla 10 y Tabla 11.

IdArista	fromNodo	toNodo	largo	costo
1	1	3	170	1,054
2	3	5	30	0,20
3	5	4	40	0,30
4	4	2	70	0,32

Tabla 10: Actualización de la información de las aristas

Id_Parada	Id_Destino
3	1
4	3
5	3

Tabla 11: Actualización de la información de las paradas

En este escenario si se quiere eliminar la parada 3, además de recomponer las aristas como se explicó en la sección anterior, se debería actualizar la información de las paradas ya que las paradas 4 y 5 tienen referencia a la parada 3. Para simplificar este proceso y evitar actualizar toda la tabla de paradas cada vez que se elimina una, es que se agregan dos campos a la tabla de aristas: fromNodoO y toNodoO, en ellos se guardarán los extremos de las aristas originales ya que estos corresponden a nodos viales de cruces que nunca podrán ser eliminados.

La tabla de información de las aristas del ejemplo se observa en la Tabla 12.

IdArista	fromNodo	toNodo	largo	costo	fromNodoO	toNodoO
1	1	3	170	1,054	1	2
2	3	5	30	0,20	1	2
3	5	4	40	0,30	1	2
4	4	2	70	0,32	1	2

Tabla 12: Nuevos campos fromNodoO y toNodoO

Para determinar el sentido de las paradas (Figura 13) se utilizarán siempre los nodos origen de la arista, así no quedarán referencias en la tabla de paradas a nodos que podrían ser eliminados.

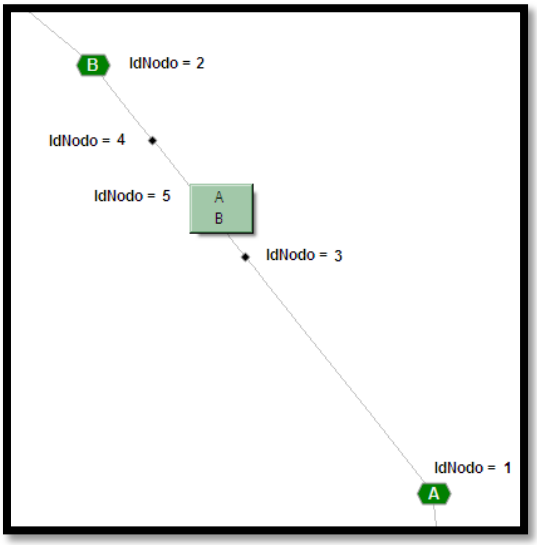

Figura 13: Determinar sentido de una parada

Información de shape

Al diseñar la red vial, es muy importante tener un conocimiento desde la interfaz gráfica sobre los datos de los elementos que se van creando. Por ejemplo, saber el sentido de las paradas, para determinar por donde trazar los recorridos. Aunque esta información es de vital importancia en el módulo de manipulación para el trazado de recorridos, también es útil en el módulo construcción.

IgoR-tp cuenta con dos formas de visualizar datos de los elementos geográficos:

1. Notas del mapa: despliega información sobre el mapa cuando el cursor del ratón pasa por encima del mismo. Permite definir qué campos de las capas que contienen información, se desplegarán cuando las notas del mapa estén activas. Disponible únicamente para las capas puntos de demanda y red vial.
2. Información de la capa: Permite mostrar información de las distintas capas o de la matriz de demanda en formato de tabla.

Sin embargo, ninguna de estas funcionalidades era de utilidad cuando se quería consultar por los datos de un elemento específico. Por lo tanto, se crea la funcionalidad “Información de shape”, que queda disponible tanto en el módulo de construcción como en el de manipulación.

Mediante un mensaje emergente (Figura 14) se brinda información cada vez que se hace clic sobre los siguientes elementos: (i) una arista, (ii) un centroide, (iii) un nodo vial o (iv) una parada.

La Tabla 13 describe la información brindada sobre cada elemento.

Elemento	Información brindada
Arista	Conexión: Id de arista Origen: Id nodo from - Destino: id nodo to
Centroide	Centroide: Id de centroide
Nodo Vial	Nodo Vial: Id de nodo vial
Parada	Parada: Id de la parada - Destino: Id del nodo vial que indica el sentido de la parada

Tabla 13: Información de shape

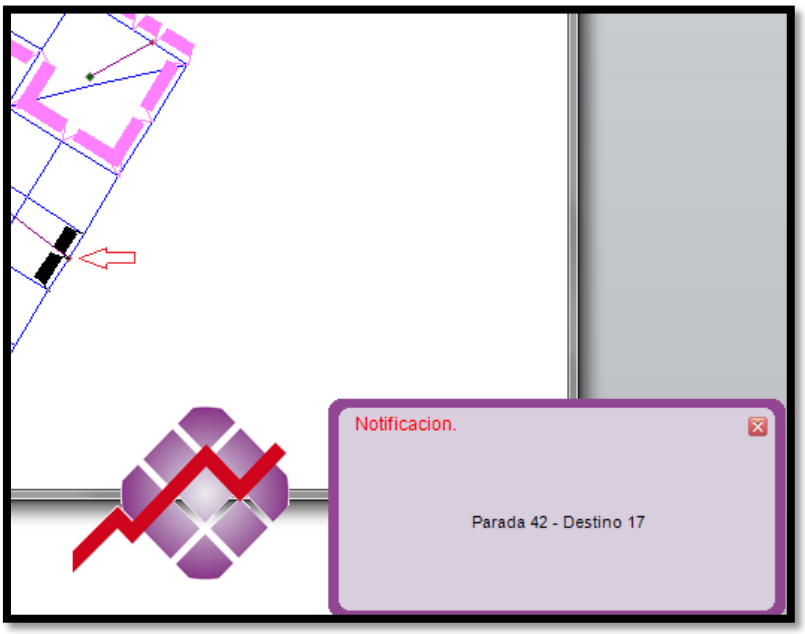

Figura 14: Información de shape para una parada

3.3.2. Módulo de manipulación

Descripción

Mediante este módulo es posible crear un nuevo proyecto donde se permite manipular recorridos para: ya sea crearlos o eliminarlos, y para cada uno de ellos, editar sus tramos, cambiarles el sentido y agregarles una frecuencia.

Por definición, un recorrido es un conjunto de nodos viales, que están organizados mediante tramos. Un tramo es una arista cuyos extremos son dos nodos viales adyacentes. Los recorridos pueden ser: (i) circulares (ii) no circulares. Un recorrido no circular tiene nodo inicio y un nodo fin, mientras que los circulares comienzan y terminan en el mismo nodo vial. Además cada recorrido consta con un sentido y una frecuencia que representa el tiempo entre pasadas de los buses que están asociados al recorrido en cuestión.

Para crear un proyecto en el módulo manipulación se puede partir de un proyecto creado en el módulo de construcción o realizarlo de cero seleccionando las capas necesarias. Si se quiere utilizar un proyecto de construcción es necesario que previamente se haya calculado su matriz de demanda.

Los recorridos se agrupan en una estructura llamada *solución* y pueden existir varias soluciones por proyecto. Una solución generada por el módulo de manipulación podrá además ser evaluada por un algoritmo existente en el sistema dentro del módulo de algoritmos.

3.3.2.1. Cambios funcionales

El modificar la representación de las paradas agregándoles sentido impacta también en la representación de los recorridos y por lo tanto en el módulo de manipulación que es donde éstos se definen. Si bien los pasos a seguir para la creación de un proyecto de manipulación sigue respetando a la versión 2.0 de igoR-tp, se introducen cambios en la validación del recorrido y se modifican y agregan algunas funcionalidades que consideramos de gran utilidad. A continuación se detallan dichos cambios.

Eliminación de recorridos no dirigidos

El recorrido, como se mencionó anteriormente, es un conjunto de tramos que se componen por nodos viales, a su vez algunos de esos nodos son paradas que en esta nueva versión tienen un único sentido posible de arribo de los buses.

En la versión anterior del módulo de manipulación se podían definir varias clases de recorridos: circulares dirigidos, no circulares dirigidos y no circulares no dirigidos. Un recorrido no circular no dirigido deja de tener relevancia en un contexto donde las paradas tienen un único sentido. Esto sucede porque en este nuevo modelo vamos a querer que todas las paradas de un recorrido tengan el mismo sentido y que además éste coincida con el sentido del recorrido. Por lo tanto, en la nueva versión de igoR-tp no se pueden definir recorridos no dirigidos.

Al crearse un recorrido se le asigna por defecto un sentido. Luego en caso de que el sentido no sea el deseado, mediante doble clic en el nombre del recorrido se despliega una pantalla donde se puede invertir su sentido (Figura 15).

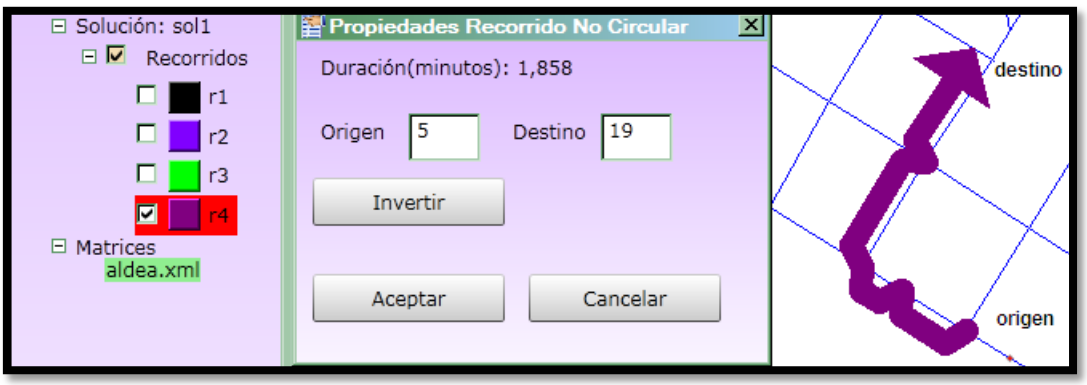

Figura 15: Invertir sentido de un recorrido

Validación de los recorridos

En la versión anterior la única validación que se hacía sobre un recorrido es que este fuera continuo. Debido a que el modelo sufrió el cambio de cómo se manejan las paradas y que ahora se cuenta con la información del sentido de las mismas, es necesario controlar que el recorrido incluya paradas en su sentido. Un recorrido con ninguna parada cuyo sentido coincida con el suyo, sería un recorrido que no tendría ninguna utilidad, pues por el mismo circularían buses que no podrían levantar o dejar pasajeros en ninguna parada.

En este punto había dos alternativas posibles: (i) permitir construir recorridos sin paradas en igoR-tp (ii) controlar que todos los recorridos en igoR-tp incluyeran al menos una parada cuyo sentido se corresponda con el del recorrido.

Se realiza la opción (II) ya que resultaba natural agregar este chequeo en el mismo lugar que se realiza la validación de continuidad, de esta manera la verificación se realiza una vez que el usuario confirma el recorrido y no se ve alterada la interfaz gráfica.

En caso de que el recorrido no incluya ninguna parada en su sentido, se mostrará un mensaje emergente indicando al usuario esta situación y sugiriéndole dos alternativas: (i) agregar paradas que efectivamente coincidan con el sentido del recorrido o (ii) invertir el sentido del recorrido.

Si bien la nueva versión de IgoR-tp es más restrictiva al momento de definir los recorridos, recordemos que se agregó la opción información de shape con la cual el usuario podrá consultar el sentido de una parada antes de agregarla al recorrido. El comportamiento esperado al definir un recorrido es el siguiente: primero el usuario identifica cuales paradas necesitará para crear un recorrido haciendo clic en las mismas y viendo su sentido, luego diseña el recorrido que pase por ellas haciendo coincidir el sentido del recorrido con el sentido de las paradas.

Además se agregó un control adicional al momento de generar un recorrido donde se verifica que el mismo no comience ni finalice en una parada. Este control se debió agregar para que todos los nodos que indican el sentido de las paradas quedaran incluidos en el recorrido. Ya que si no se imponía esta restricción podría pasar que el nodo inicial de un recorrido fuera una parada cuyo sentido estuviera en dirección a un nodo que no perteneciera al recorrido, lo que sería un problema al momento de validar el recorrido.

Carga externa de matrices de demanda

Para crear un proyecto de manipulación desde un proyecto del módulo construcción, se debe haber previamente calculado al menos uno matriz de demanda en el proyecto construcción. De esta forma al generar el proyecto manipulación las matrices previamente generadas quedan directamente asociadas al mismo. Sin embargo es muy común que se quiera modificar la matriz de demanda una vez que ya se construyó el proyecto manipulación. En la antigua versión no había forma de hacer ninguna modificación sobre las matrices de demanda.

Es por esto que se decidió agregar al módulo manipulación una nueva funcionalidad que permite agregar matrices de demanda al proyecto manipulación.

Lo que el usuario debe hacer para acceder a dicha funcionalidad es realizar un clic derecho sobre el sector de la interfaz correspondiente a las matrices. Esto desplegará un menú contextual con la

opción “Cargar Matriz” al realizar clic sobre esta opción aparecerá una ventana de búsqueda de archivos para seleccionar la matriz que se desee agregar al proyecto (ver Figura 16).

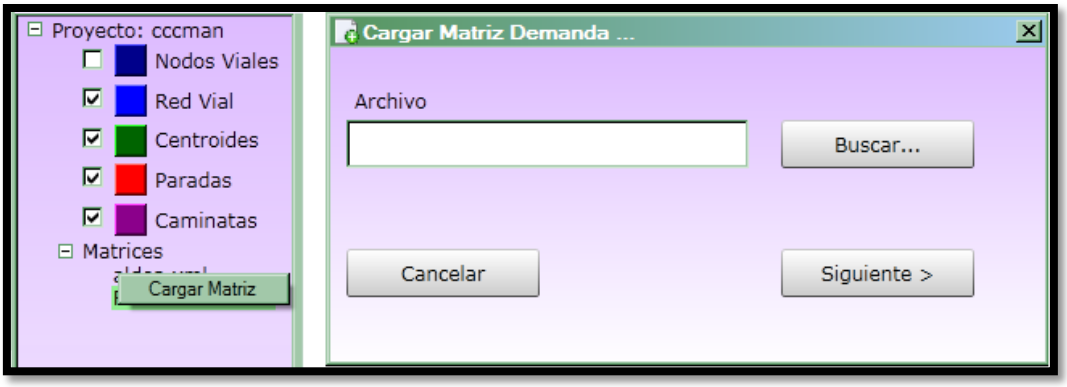

Figura 16: Cargar matriz de demanda

Selección de los parámetros necesarios para la evaluación de una solución

Cuando se deben seleccionar los parámetros necesarios de un algoritmo de evaluación, se despliega una ventana para que el usuario ingrese dichos parámetros. En el caso de que uno de los parámetros fuera un archivo, la versión anterior requería que el usuario ingresara la ruta absoluta del archivo en forma de texto, lo que no era muy amigable.

Este diálogo fue modificado de forma tal que, en caso de que el tipo del parámetro sea archivo se agrega un botón que abre un nuevo diálogo de búsqueda de archivos para que el usuario pueda seleccionar el archivo necesario desde aquí (Figura 17). Esta mejora busca acelerar el proceso de selección de archivos respecto a la versión anterior donde era necesario especificar la ruta completa.

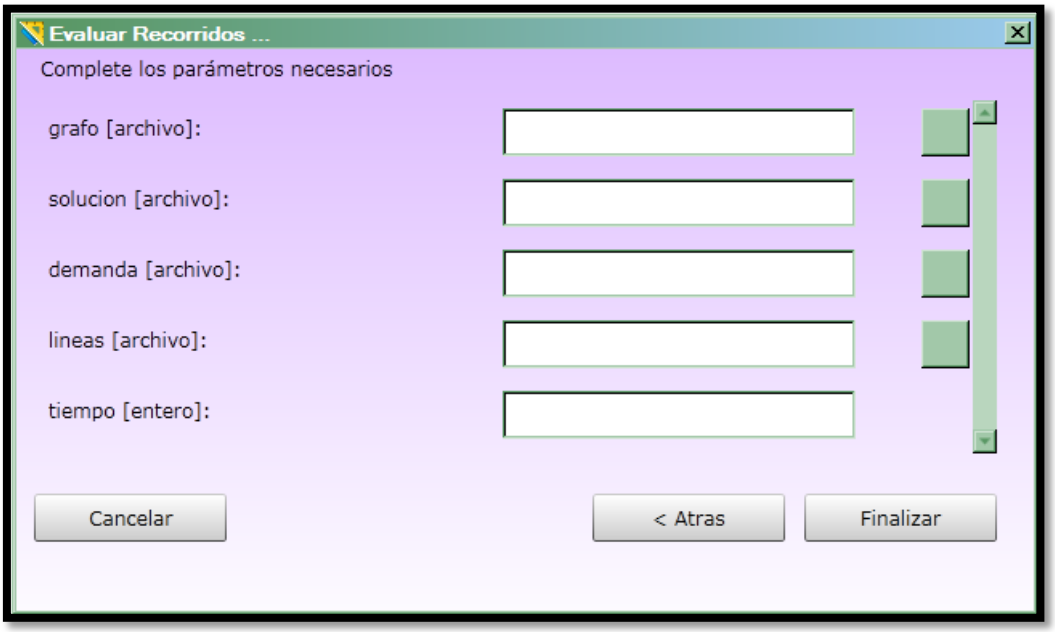

Figura 17: Selección de parámetros de tipo archivo

Cambios en los archivos de persistencia

Al momento de persistir la solución, se almacenan los recorridos en el archivo *.solucion* en formato *XML*. Este archivo describe todos los recorridos que forman parte de la solución y el conjunto de nodos viales que lo componen junto con toda la información relevante del mismo. En la *sección 3.5.3* se puede encontrar una descripción detallada de los archivos de tipo *.solucion*.

En este archivo únicamente se almacenan los nodos viales del recorrido, por lo tanto había que encontrar una forma de especificar qué nodos viales de los que componen el recorrido son paradas válidas para dicho recorrido. Se entiende por parada válida una parada que esté incluida en el recorrido y cuyo sentido coincida con el sentido del recorrido.

Como respuesta a esta necesidad se optó por agregar un atributo extra a cada nodo vial del archivo solución, que indica si es parada o no para el recorrido. Con esto se busca vincular aquellas paradas que se establecieron en el módulo de construcción, con las paradas válidas para un recorrido del módulo de manipulación.

Este cambio es necesario porque el archivo *grafo.txt* almacena las paradas del módulo del construcción pero a nivel de manipulación, una parada del módulo de construcción no necesariamente es parada para un recorrido (ya que podría no respetar su sentido), por lo tanto se trata como un simple nodo vial.

A continuación se muestra un ejemplo de los datos de un recorrido en el archivo *XML* correspondiente a una solución, se resalta en amarillo un ejemplo de nodo que es parada para el recorrido.

```
<recorrido>
      <nombre>LineaEscuela88DesdeTrapani.shp</nombre>
      <ruta>C:\Program Files (x86)\igoR-
tp\Proyectos\Rivera2Manipulacion\Soluciones\Manuales\RiveraUltima\</ruta>
      <frecuencia>30</frecuencia>
      <activo>False</activo>
      <color>
        <R>0</R>
        <G>0</G>
        <B>0</B>
      </color>
      <visible>False</visible>
      <nodos>
        <nodo esParada="0">62</nodo>
        <nodo esParada="0">394</nodo>
        <nodo esParada="0">393</nodo>
        <nodo esParada="1">61</nodo>
        <nodo esParada="0">392</nodo>
        <nodo esParada="0">102</nodo>
      </nodos>
      <circular>False</circular>
      <dirigido>True</dirigido>
    </recorrido>
```

3.3.3. Módulo de algoritmos

Descripción

Como se explicó anteriormente este módulo permite gestionar algoritmos externos a igoR-tp. Todo algoritmo que se desee emplear para evaluar una solución debe de ser dado de alta previamente en el módulo de algoritmos definiendo el ejecutable correspondiente al algoritmo, la cantidad y tipo de los parámetros de entrada que éste necesite.

Cambios funcionales

Si se quiere evaluar una solución del módulo manipulación con un algoritmo, este debe definirse previamente en el módulo de algoritmos, indicando cuántos parámetros recibe y para cada uno especificar su tipo. En la versión anterior los tipos posibles de los parámetros de entrada de un algoritmo eran: caracteres, entero, real. Entonces si el algoritmo que se quería definir necesitaba como entrada un archivo, lo que se hacía era definir ese parámetro como tipo caracteres y luego al invocarlo para la evaluación de una solución se debía ingresar la ruta del archivo como texto.

El algoritmo usado por este proyecto es la simulación que necesita como entrada cuatro archivos, se notó que era tedioso el tener que digitar la ruta de cada uno de los parámetros de entrada cada vez que se quería correr la simulación. Por este motivo se agregó el tipo "archivo" a los tipos posibles de los parámetros de entrada (Figura 18).

Figura 18: Configuración de un algoritmo con parámetros tipo archivo

Cuando se desee evaluar una solución en el módulo de manipulación utilizando un algoritmo que tenga algún parámetro de este tipo, se permite que dichos parámetros puedan ser seleccionados mediante un cuadro de diálogo de búsqueda de archivos (Figura 19).

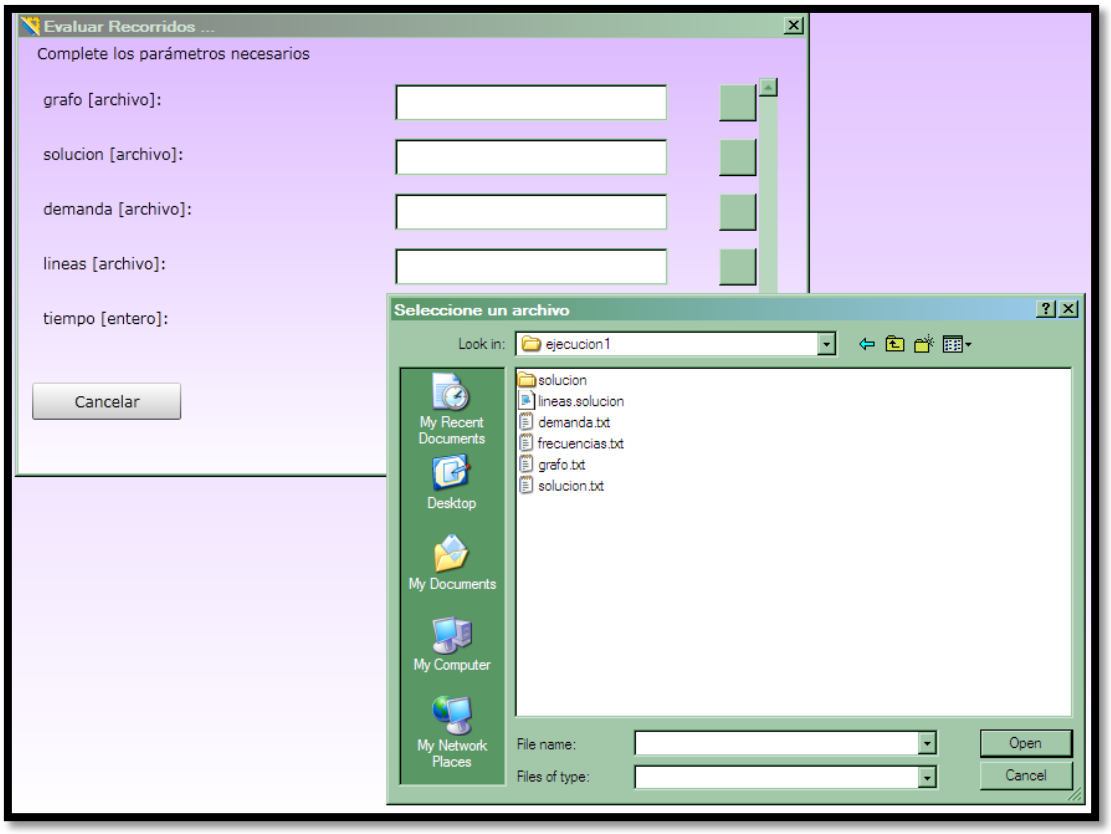

Figura 19: Selección de ubicación de un parámetro de tipo archivo

3.4. Verificación

Si bien durante todo el desarrollo de la nueva versión de igoR-tp se fueron haciendo pruebas funcionales, la verificación final del producto quedó a cargo de un docente del Departamento de Investigación Operativa, quien desarrolla tareas en el marco del proyecto CSIC relacionado con este Proyecto de Grado.

Dicha verificación constó en construir de cero el caso de prueba correspondiente al transporte público de la ciudad de Rivera. Este caso de estudio es el mismo que fuera utilizado por los dos proyectos anteriores, pero al permitir la nueva versión agregar las paradas en cualquier sector del eje de calle, se logró una ubicación de las mismas mucho más aproximada a la real. Para cada parada se agregó el sentido correspondiente según la dirección en que circulan los buses que allí paran.

Para construir el caso de la ciudad de Rivera se crearon con el módulo construcción 296 paradas, 84 zonas y se definieron 291 caminatas (Figura 20).

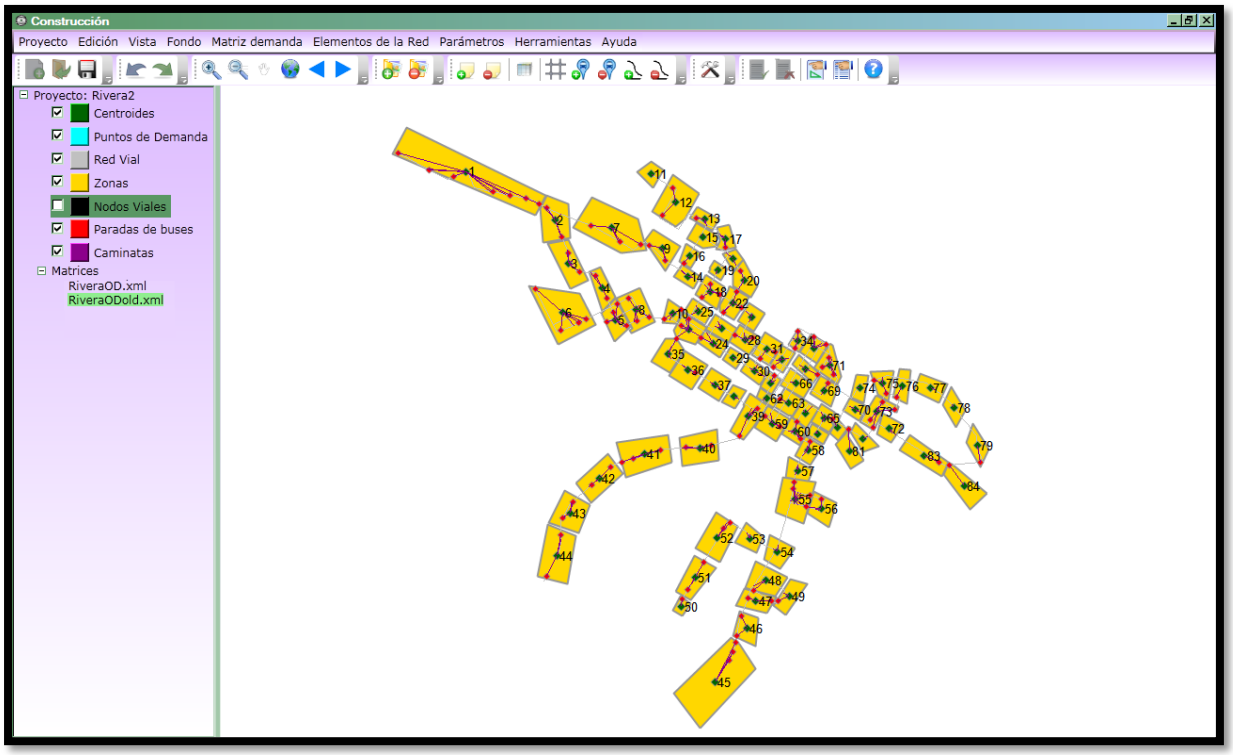

Figura 20: Caso de estudio ciudad Rivera - módulo construcción

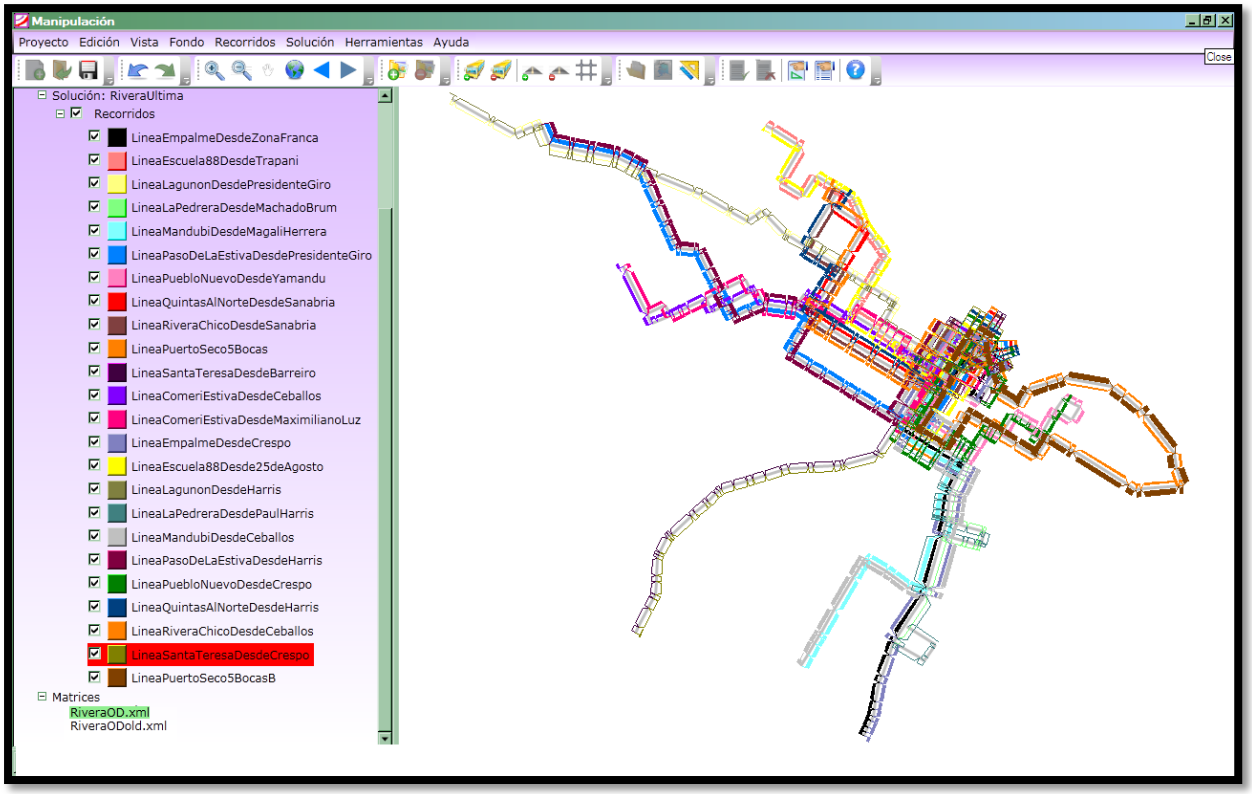

Figura 21: Caso de estudio ciudad de Rivera - módulo manipulación

Luego en el módulo de Manipulación se definieron los recorridos que serán utilizados por las líneas. En la ciudad de Rivera existen 13 líneas de transporte, de las cuales 2 son circulares por lo tanto se debieron crear 24 recorridos en el módulo de Manipulación (Figura 21). Una completa descripción del caso de la ciudad de Rivera se puede encontrar en [5].

Los errores encontrados durante el proceso de verificación fueron corregidos y de esta forma el usuario dio su consentimiento para la aceptación del producto final.

3.5.Archivos de entrada y salida

3.5.1. Persistencia de la demanda de pasajeros entre zonas

Cuando se tiene creado un proyecto en el módulo de construcción se pueden asignar al mismo una o varias matrices de demanda de pasajeros. Para agregar una matriz de demanda a un proyecto del módulo de construcción se debe contar con un archivo de tipo *mdb* (*Microsoft Access Database*). Es necesario que este archivo incluya una tabla que indique la demanda de pasajeros entre dos puntos de demanda. La tabla debe tener las siguientes columnas:

- origen: id del punto de demanda origen.
- destino: id del punto de demanda destino.
- hora: hora del día en la que se realizarán los viajes desde el origen al destino.
- cantidad: cantidad de viajes realizados a esa hora.

origen	destino	hora	cantidad
374	615	11:30:00 p.m.	52,09
588	381	10:00:00 a.m.	35,25
381	534	11:30:00 a.m.	35,25
975	381	08:10:00 a.m.	35,25
381	975	10:30:00 a.m.	35,25
589	487	07:59:59 a.m.	35,25
1072	338	08:30:00 a.m.	35,25
338	1072	10:59:59 a.m.	35,25
70	771	07:00:00 a.m.	57,43
771	70	06:29:59 p.m.	57,43
1077	69	01:00:00 p.m.	53,16
69	1077	02:39:59 p.m.	53,16

Tabla 14: Ejemplo de tabla de demanda del archivo mdb

Además se debe indicar el horizonte horario donde se quiere calcular la demanda. De esta forma igoR-tp calculará para cada zona definida en el proyecto la demanda de pasajeros hacia las demás zonas en el horizonte de tiempo indicado. Esta información es persistida en un archivo *XML*, generándose un archivo por cada matriz de demanda creada.

Los archivos *XML* correspondientes a matrices de demanda tienen el siguiente formato:

```
<matriz_demanda>
  <hora_inicio>00:01 AM</hora_inicio>
  <hora_fin>11:59 PM</hora_fin>
  <N>84</N>
  <origen>
    <id>0</id>
    <destino>
      <id>2</id>
      <demanda>0.081818</demanda>
    </destino>
    <destino>
      <id>7</id>
      <demanda>0.027273</demanda>
    </destino>
</origen>
  <cant_demanda>13.93939</cant_demanda>
</matriz_demanda>
```

Donde cada etiqueta o tag *XML* representa:

- **<hora_inicio>**: indica la hora de inicio del intervalo de tiempo considerado para calcular la demanda.
- **<hora_fin>**: indica la hora de fin del intervalo de tiempo considerado para calcular la demanda.
- **<N>**: es la cantidad de zonas definidas en el proyecto construcción.
- **<origen>**: indica el comienzo de un tag que definirá la demanda entre una zona y todas las demás. Es así que dentro de este tag se encuentra:
    - **<id>**: corresponde al id de la zona origen de la demanda.
    - **<destino>**: indica el comienzo de un tag que especificará la demanda entre la zona origen y la zona destino especificada aquí. Dentro de este tag se especifica:
        - **<id>**: corresponde al id de la zona destino de la demanda para el origen especificado.
        - **<demanda>**: el valor de la demanda entre la zona origen y la zona destino.

Nótese que solo se generan registros cuando la demanda entre dos zonas es mayor que cero.

Finalmente en cant_demanda se guarda la cantidad total de demanda generada.

Los archivos *XML* correspondientes a las matrices de demandas generadas para un proyecto construcción se guardan en el directorio *Matrices* bajo la raíz del proyecto construcción.

Luego de creada una matriz de demanda en el módulo de construcción al generarse un proyecto manipulación a partir de él, el módulo de manipulación persistirá la matriz de demanda seleccionada como activa en un archivo de texto. Este archivo se llama *demanda.txt* y se puede encontrar en el directorio raíz del proyecto manipulación. Este archivo tiene la misma información que el archivo *XML* explicado anteriormente solo que en otro formato.

Ejemplo de un archivo *demanda.txt*:

```
84
380
0 2 0.081818
0 7 0.027273
0 17 0.021212
0 22 0.068182
```

Donde la primera línea indica la cantidad de zonas definidas en el módulo construcción. La segunda línea indica la cantidad de líneas del archivo. Luego en cada línea está definida la demanda de la siguiente manera:

```
<centroide_origen1> <centroide_destino1> <demanda1>
<centroide_origen1> <centroide_destino2> <demanda2>
…
<centroide_origenN> <centroide_destinoN> <demandaN>
```

La semántica de los identificadores a partir de la tercera línea es:

- **<centroide_origen>:** corresponde al identificador del centroide origen de la demanda.
- **<centroide_destino>:** corresponde al identificador del centroide destino.
- **<demanda>:** corresponde al valor de la demanda de pasajeros para el par de centroides origen-destino. La unidad de este valor es 1/minutos.

3.5.2. Persistencia del grafo que representa la red vial

Al crearse un proyecto en el módulo de manipulación éste guarda la información de la red subyacente definida en un archivo de texto llamado *grafo.txt*. Este archivo guarda la información del grafo *G=(N, A)* que representa la red vial. Donde *N* es el conjunto de nodos y *A* es el conjunto de aristas definidas entre ellos. Para cada nodo se almacena sus coordenadas geográficas, se indica el tipo de nodo que representa y además se especifica para cada uno de sus nodos adyacentes que tipo de nodo es, qué tipo de arista los une y el costo asociado a ella.

A continuación en la Tabla 15 se presenta la estructura del archivo *grafo.txt* detallándose cada uno de sus datos.

Dato	Tipo	Descripción
Id Nodo	Entero	Identificador del nodo que dependiendo del tipo se corresponde con el id de nodo vial, parada o centroide.
Tipo Nodo	Entero	Para este dato se define el siguiente enumerado de valores: 0-nodo vial, 1-parada y 2-centroide
Coordenada x	Real	Valor de la coordenada geográfica en el eje x
Coordenada y	Real	Valor de la coordenada geográfica en el eje y
Nodos adyacentes	Información de los nodos adyacentes. Se repite los mismos datos para cada nodo adyacente.	
Id Nodo	Entero	Identificador del nodo adyacente que dependiendo del tipo se corresponde con el id de nodo vial, parada o centroide.
Tipo nodo	Entero	Para este dato se define el siguiente enumerado de valores: 0-nodo vial, 1-parada y 2-centroide
Tipo arista	Entero	Para este dato se define el siguiente enumerado de valores: 0-arista vial, 1-arista centroide – parada, 2-arista parada - parada
Costo	Real	Costo de la arista, representa el tiempo que se demora en transitarla.

Tabla 15: Detalle de los datos de grafo.txt

El formato de los datos antes descriptos es el siguiente:

```
Id Nodo : Tipo_Nodo : coord._x : coord._y <espacio> IdNodo Adyacente[1] :
Tipo adyacente[1] : Tipo_arista : costo-   …    Adyacente[n] : Tipo
adyacente[n]: Tipo_arista : costo -
```

Ejemplo:

```
133:0:526009.453579106:6581520.48616911 134:0:0:0.608-238:1:0:0.514-
```

Cada línea del archivo representa la información de un nodo, se utiliza el carácter delimitador dos puntos ":" para separar los datos del mismo y el carácter guion "-"para separar la información de cada nodo adyacente.

3.5.3. Persistencia de los recorridos en una solución.

El módulo de manipulación guarda los datos de una solución en un archivo XML cuyo nombre es igual al nombre de la solución y cuya extensión es ".solucion". En este archivo se almacenan datos generales como la configuración de frecuencias y capacidades de los buses, la ruta donde se encuentra el archivo y datos particulares de cada recorrido.

Para cada uno de los recorridos se almacena el nombre de su archivo shapefile, la ruta donde se encuentra dicho archivo, la frecuencia, el color en que se debe mostrar en el mapa, si está visible o no, si el recorrido es circular, si el recorrido es dirigido y una lista de los nodos que lo componen. Para cada nodo se especifica si se debe considerar parada para el recorrido.

Los archivos con extensión ".solucion" tienen el siguiente formato:

```
<Solucion>
  <nombre>Rivera</nombre>
  <ruta>C:\Proyectos\Rivera2Manipulacion\Soluciones\Manuales\</ruta>
  <capacidad>28</capacidad>
  <frecuencias>
    <frecuencia>20</frecuencia>
    <frecuencia>30</frecuencia>
    <frecuencia>40</frecuencia>
    <frecuencia>60</frecuencia>
  </frecuencias>
  <recorridos>
    <recorrido>
      <nombre>LineaComeriEstivaDesdeCeballos.shp</nombre>
      <ruta>C:\Rivera2Manipulacion\Soluciones\Manuales\Rivera\</ruta>
      <frecuencia>1</frecuencia>
      <activo>True</activo>
      <color>
        <R>0</R>
        <G>0</G>
        <B>0</B>
      </color>
      <visible>False</visible>
      <nodos>
        <nodo esParada="0">38</nodo>
        <nodo esParada="0">419</nodo>
        <nodo esParada="0">39</nodo>
…
    </nodos>
      <circular>False</circular>
      <dirigido>True</dirigido>
    </recorrido>
  </recorridos>
  <evaluaciones />
  <evalActiva />
</Solucion>
```

Donde cada etiqueta o tag *XML* representa:

- **<nombre>**: indica el nombre de la solución.
- **<ruta>:** indica la ruta donde se guarda este archivo.
- **<capacidad>:** indica la capacidad de los buses.
- **<frecuencias>:** indica la configuración de frecuencias.
- **<recorridos>:** bajo este tag se guarda toda la información de los recorridos, donde para cada uno se especifica:
    - **<nombre>**: nombre del archivo Shapefile asociado al recorrido.
    - **<ruta>**: directorio donde se encuentra el archivo Shapefile.
    - **<frecuencia>**: frecuencia del recorrido.
    - **<activo>**: indica si el recorrido está activo en la solución.
    - **<color>**: código RGB que especifica el color con que se debe mostrar el recorrido.
    - **<visible>**: indica si el recorrido está visible.
    - **<nodos>**: colección de los nodos que componen el recorrido, para cada nodo se especifica su identificador y si es parada o no para este recorrido.
- **<circular>:** indica si el recorrido es o no circular.
- **<dirigido>:** indica si el recorrido es o no dirigido.

4. Simulador

4.1. Introducción

En el presente capítulo se describe la herramienta de simulación utilizada para modelar distintos comportamientos de los pasajeros en un sistema TPUC. Esta herramienta, de ahora en más llamada simulador, es un componente de un modelo llamado: *modelo de simulación*.

El modelo de simulación permite abstraer sistemas de la vida real que evolucionan a través del tiempo. En el contexto de un sistema de gran tamaño como el sistema de transporte público, este modelo ofrece la ventaja de experimentar y explorar las interacciones producidas entre buses (declarados como recursos) y pasajeros (individuos) recolectando resultados relevantes. La finalidad de un modelo de simulación es producir distintos resultados que sirvan de apoyo a la toma de decisiones, y para este contexto particular, a los planificadores del TPUC. La Figura 22 indica el contexto del simulador en el ciclo del proyecto:

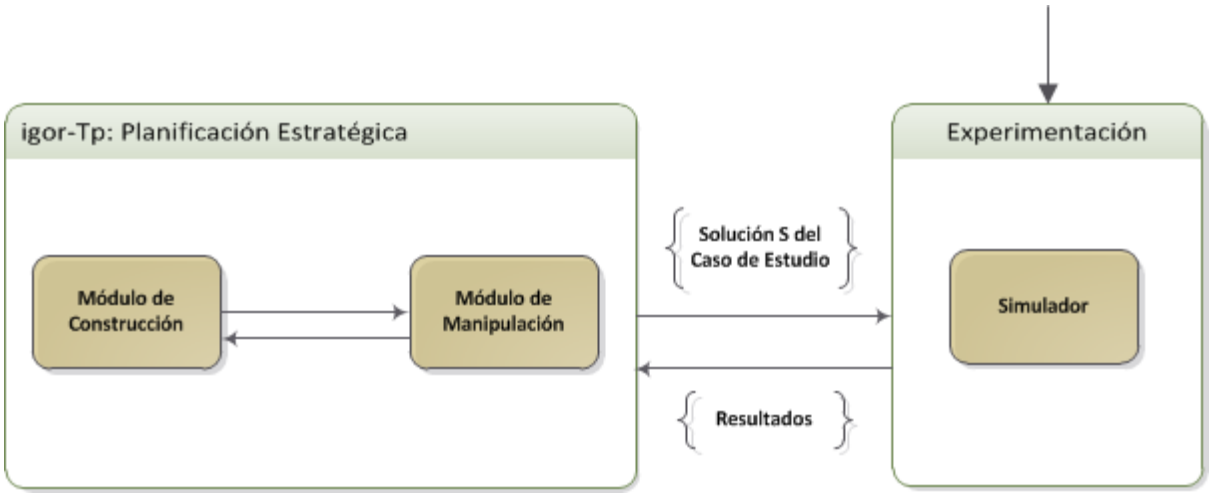

Figura 22: Ciclo de proyecto

Como se observa en la Figura 22, el simulador se abastece de un conjunto de datos provenientes de igoR-tp (ver sección 4.6: *Archivos de Entrada y salida*), realiza una serie de evaluaciones y su salida puede ser consumida nuevamente por igoR-tp para correcciones y mejoras.

La implementación de un nuevo módulo de comportamiento se realizará sobre un modelo de simulación existente diseñado por [3] para el TPUC.

4.2. TNDP y el Modelo de Simulación

Como se explicó en el capítulo 2: *Marco Teórico*, el modelo de optimización para el TNDP debe considerar en su función objetivo, los intereses de los usuarios (pasajeros) y operadores. Esto es, porque al momento de generar una solución particular del TNDP, el modelo de simulación se utilizará como un algoritmo de evaluación, participando de un ciclo de: Generación – Evaluación – Mejora. La Figura 23 ilustra la relación entre los modelos:

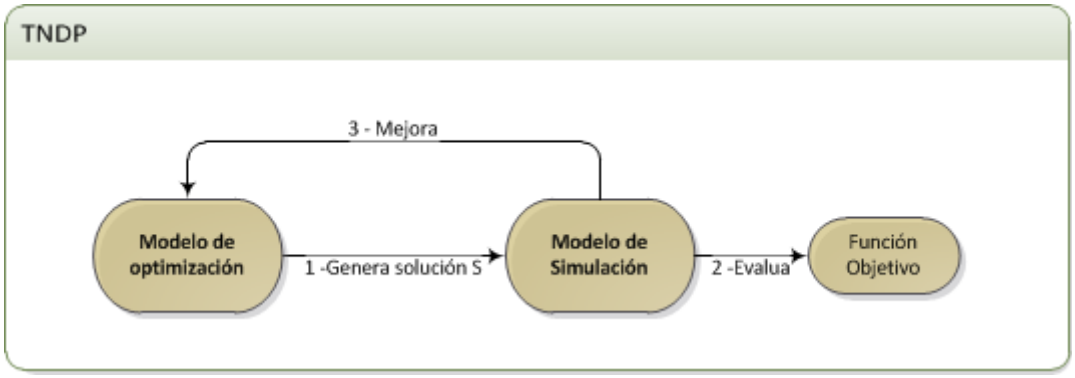

Figura 23: Ciclo de optimización de soluciones del TNDP

El modelo de optimización genera soluciones mediante un cierto algoritmo. Luego el modelo de simulación evalúa la función objetivo para cada solución S y el modelo de optimización realiza (en caso de ser necesario) ajustes sobre cada solución evaluada. Este ciclo se realiza por parte del algoritmo de optimización de recorridos y frecuencias del TNDP [6].

La evaluación de soluciones por parte del modelo de simulación a través de la distribución de la demanda de pasajeros sobre los recorridos (definiendo comportamientos) lo transforma en un *modelo de comportamiento*. La importancia de este tipo de modelos reside en que mediante la evaluación de soluciones ayuda al planificador del TPUC a la toma de decisiones. Para ello el modelo de comportamiento deberá tener en cuenta:

- Función objetivo: se define y evalúa la función objetivo del problema.
- Demanda: la cual especifica las necesidades de traslado entre distintos puntos de la ciudad. Cada pareja de centroides tiene una demanda especificada según una frecuencia de arribos.
- Algoritmo de selección de líneas: especifica una metodología para la selección de las líneas por parte de los pasajeros (lo que determina una distribución de la demanda sobre los recorridos).
- Red de Transporte: el grafo de la red vial utilizado por los servicios de transporte, compuesto por nodos y arcos. Ver capítulo 2: *Marco Teórico*.
- Solución particular del TNDP: dado por un conjunto de recorridos y frecuencias de las líneas.

4.3. Modelo de Simulación a eventos discretos

La simulación a eventos discretos, de ahora en más SED, es un tipo de simulación cuya representación se basa en una secuencia o serie de eventos que son agendados y suceden en intervalos discretos de tiempo. Para una información más detallada de la metodología SED ver *Apéndice E - sección 1*.

El desarrollo de un proceso SED consta con ciertas etapas bien definidas:

1. Análisis del problema con una detallada descripción del mismo.
2. Definición del modelo como tal en función de sus hipótesis, factores y respuestas de interés.
3. Implementación en un lenguaje de programación.
4. Diseño de plan de pruebas.
5. Análisis de resultados.
6. Verificación y validación del modelo.

Los puntos (1), (2) y (3) definen al modelo de simulación a eventos discretos, de ahora en más modelo de simulación, mientras que los puntos restantes se orientan a verificar que dicho modelo produce los resultados esperados.

De forma abstracta es posible considerar al modelo de simulación como una caja negra que se alimenta de ciertos datos de entrada retornando información para su posterior análisis y toma de decisiones.

Un modelo de simulación está estructurado de la siguiente manera [8]:

- Especificación: la cual está compuesta por una descripción completa del sistema donde se declaran objetivos, hipótesis iniciales, factores y respuestas.

  Los objetivos deben ser claros y precisos ya que a partir de su definición se obtendrá el resto de los componentes de la especificación.

  Los factores son aquellos parámetros que se obtienen del modelo (por ejemplo demanda o frecuencia) y se alteran para estudiar la sensibilidad del sistema ante dichos cambios. De esta forma, modificando los factores se pueden obtener medidas de sensibilidad del modelo de simulación ante los cambios. Las respuestas de interés son datos, estadísticas y medidas que se consideren relevantes para el análisis.

  Las hipótesis iniciales ya sean implícitas o explícitas al modelo, podrían cambiar a lo largo del proceso por lo tanto es necesario que se determinen con cuidado y que el modelo sea flexible, en lo posible, a dichos cambios.

  Los componentes internos del sistema están sujetos a acciones o actividades que son necesarios estudiar, por lo tanto la especificación también incluye información de las actividades del sistema y cómo se relacionan entre sí.

- Pseudocódigo: es la descripción en alto nivel de cómo son las interacciones entre los componentes del sistema y sus actividades. Es necesario para entender los aspectos dinámicos del sistema.
- Programa: es el código fuente de la implementación del modelo de simulación. Es deseable que sea lo más simple posible de forma tal de mantener su credibilidad y es fundamental que sea robusto a los cambios, tanto en datos de entrada como en su estructura interna.

4.4. Objetivos

A continuación se brinda una descripción de los objetivos a desarrollar en el modelo de simulación en el presente proyecto. Estos son:

- El diseño de un nuevo módulo de comportamiento para los pasajeros.
- La definición de un conjunto de indicadores que permitan validar dicho modelo.

4.4.1. Nuevo módulo de comportamiento

Hasta el momento se menciona el concepto comportamiento sin brindar una definición formal. Especificar un nuevo comportamiento en la lógica de pasajeros, implica definir una nueva "*estrategia*" para los mismos.

Según [9] una estrategia es:

> *"*Llamamos a cualquier elemento de decisión del viajero como estrategia. Una estrategia es un conjunto de reglas que, al ser aplicadas, permiten al viajero arribar a su destino. El número y el tipo de las diferentes estrategias que el viajero puede elegir dependen de la información que posea durante su viaje. Si no hay información adicional disponible durante su viaje, la estrategia simplemente define una ruta*"*.

La estrategia provee la lógica del comportamiento de los pasajeros a lo largo de la simulación por lo tanto, es importante estudiarla con cuidado. Además, es deseable que los pasajeros cuenten con varias estrategias intercambiables que permitan estudiar distintos escenarios, y obtener resultados para poder analizar y comparar.

Se diseñó un nuevo módulo de estrategia para los pasajeros donde se consideraron distintas alternativas para responder la siguiente pregunta: ¿Cómo se comportaría el sistema si se consideran trasbordos?

Existen algunos motivos por los cuales se decidió contar con un modelo de simulación que brindara el servicio de trasbordo, ellos son:

- Más opciones de viaje: los pasajeros pueden llegar a tener más combinaciones de recorridos para llegar a su destino que de otra forma no sería siempre posible. Eso hace aumentar la cantidad de buses atractivos para los pasajeros.
- Se ajusta a la realidad: la mayoría de los sistemas de transporte urbanos de pasajeros en las grandes ciudades ofrecen trasbordos para comodidad de sus usuarios.
- Cambio en la redistribución de los recorridos: las empresas de transporte, al considerar trasbordos, podrían realizar una redistribución de sus recursos que podría ser beneficiosa.
- Variación en los tiempos de espera y de viaje de los pasajeros: es interesante estudiar cómo son afectados los tiempos de espera y de viaje de los pasajeros cuando existen trasbordos en el sistema.

- ¿El tamaño de la ciudad incide en la necesidad de trasbordo? ¿Una ciudad pequeña vería afectado su transporte público por el uso de trasbordos o sería beneficioso?

En base a estas ideas se creó una estrategia de pasajeros que considera a lo sumo un trasbordo para cada pasajero en su viaje a destino. Cabe acotar que el pasajero aún seleccionando esta estrategia, podría viajar sin realizar trasbordos si decide usar un bus directo a destino.

A continuación se propone un ejemplo de trasbordo en la Figura 24:

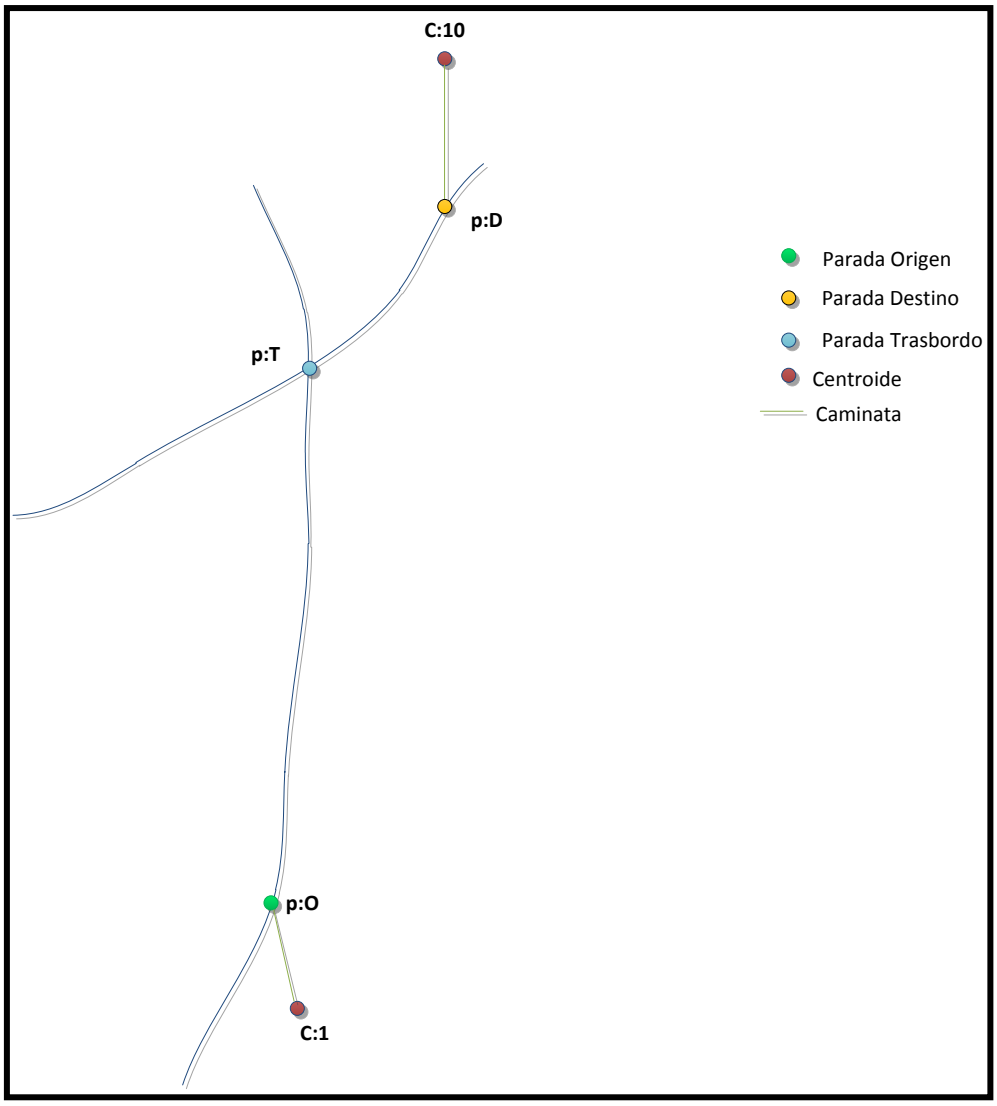

Figura 24: Ejemplo estrategia pasajero

La Figura 24 muestra un ejemplo de un caso donde un pasajero generado en el centroide c: 1 puede arribar al centroide c: 10 a través de un trasbordo por la parada p: T. Para ello es necesario contar con una línea cuyo recorrido incluya a las paradas p: O y p: T (con sentido hacia p: T), y otra línea que incluya en su recorrido a las paradas p: T y p: D (con sentido hacia p: D).

Cabe destacar que este proyecto propone un nuevo modelo de comportamiento de pasajeros y no un nuevo diseño de líneas que favorezca los trasbordos.

La descripción de la nueva estrategia se divide en dos partes:

1. Seleccionar parada origen: Al momento de crear el pasajero en una centroide determinado, se pregunta si existe una línea directa o de trasbordo que lo lleve a destino. Para ello se toman todas las posibles combinaciones de parada origen y destino (de los centroides origen y destino); y se selecciona aquella línea directa con menor tiempo de viaje.

   Si dicha línea no existe, se analizan las líneas de trasbordo utilizando una *matriz de trasbordo* previamente calculada que mantiene todos los trasbordos posibles entre líneas. Se seleccionará entonces la primera línea de trasbordo disponible que lo deje en la parada más cercana al destino. Esto le asegura al pasajero que tiene al menos un viaje (directo o de trasbordo) entre la parada origen y la parada destino, por lo tanto el pasajero recién creado viajará hacia la parada seleccionada de origen para esperar un bus.

   A continuación se describe el funcionamiento de la selección de parada origen con un ejemplo que se describe basándose en la Figura 25:

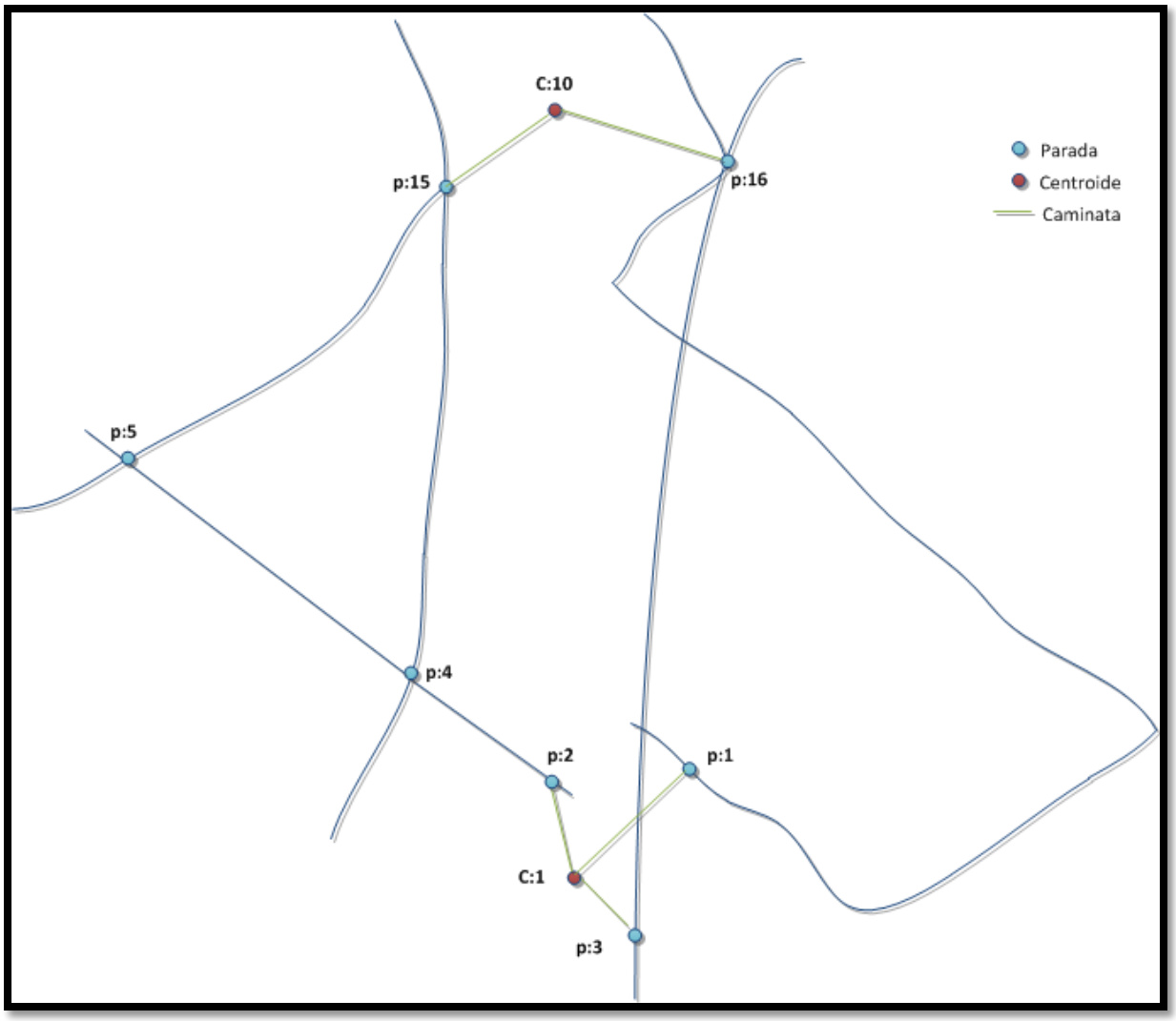

Figura 25: Funcionamiento selección parada origen

Existen pasajeros que desean ir del centroide c: 1 al centroide c: 10. El algoritmo de selección de parada iterará entre las paradas conectadas a los centroides c: 1 y c: 10 por caminatas (p:1, p:2, p:3,p:15,p:16) buscando obtener una línea. En este caso el algoritmo retornará que

la línea que pasa por parada p: 3, y finaliza en la parada p: 16, es la seleccionada ya que es la de menor tiempo de viaje versus la línea que pasa por la parada p: 1 y finaliza en la parada p: 16. Si suponemos que existen líneas directas para este ejemplo, entonces no se consideran posibles trasbordos. Por lo tanto, el pasajero caminará hacia la parada p: 3 esperando un bus (no necesariamente el de la línea candidata).

Nótese que en el caso que no exista un recorrido por p: 3 y p: 16 ni por p: 1 y p: 16, entonces los pasajeros aún podrían viajar a destino si existe un trasbordo entre líneas con recorridos que pasen por p: 2 y p: 4 y por p 4: y p: 15. De esta forma la parada p: 4 sería la parada de trasbordo.

Es importante recordar que para que los viajes sean posibles, las paradas deberán estar en el sentido del recorrido de la línea seleccionada. Por ejemplo, si los pasajeros desean ir de la parada p: 2 a la p: 4 entonces la parada p: 2 deberá tener sentido hacia la parada p: 4 y además deberá existir un recorrido en sentido de p: 2 hacia p: 4. Del mismo modo la parada p: 4 deberá tener sentido hacia la parada p: 15, al igual que el recorrido que las une.

La Figura 26 describe el diagrama de flujo relacionado con selección de la parada de origen por la estrategia.

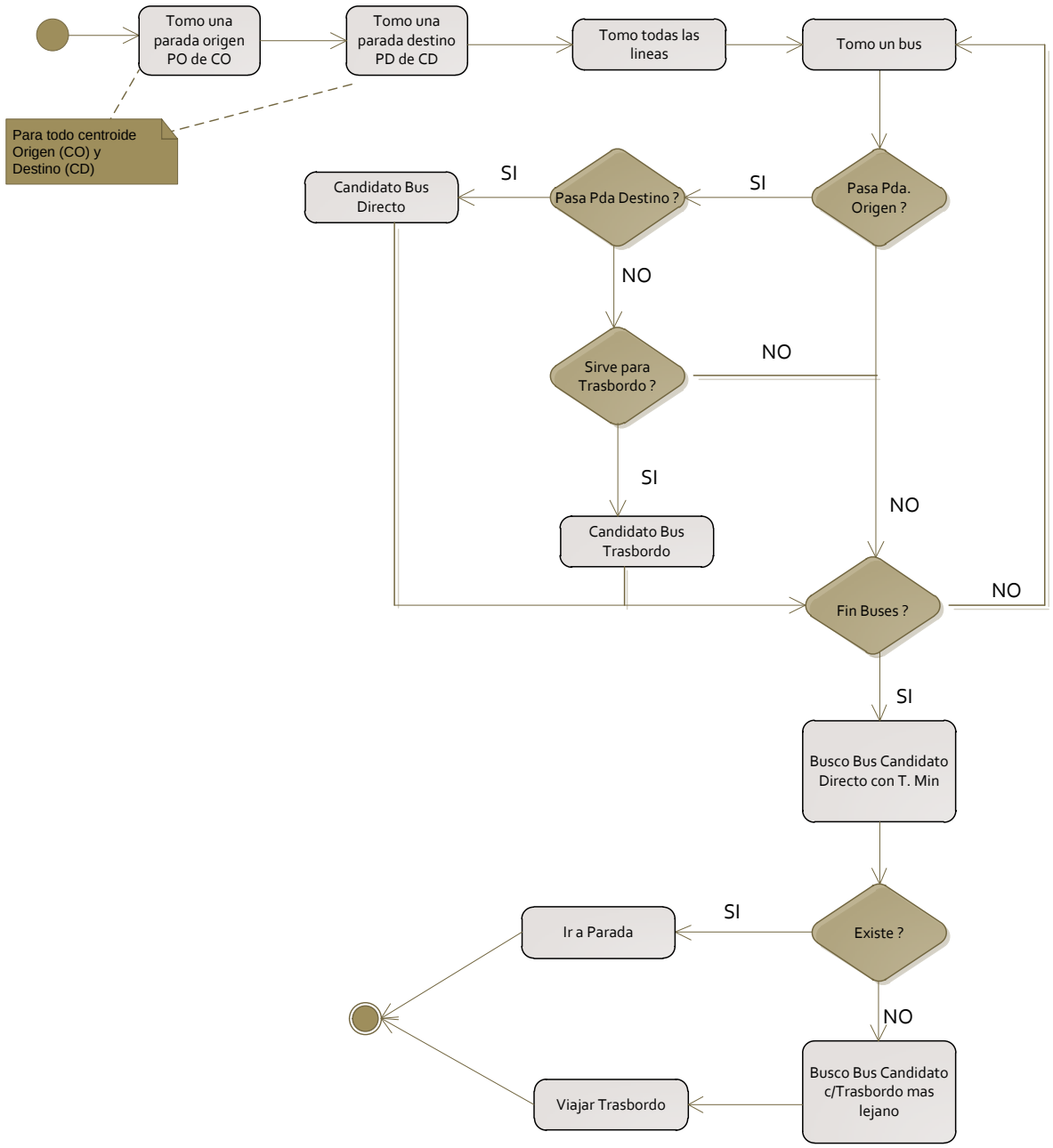

Figura 26: Diagrama de selección de parada origen

2. Seleccionar Bus: En los ejemplos del punto 1) cabe destacar que la línea seleccionada para desplazarse a la parada de origen es simplemente de referencia. Podría ocurrir que al llegar a la parada el pasajero se tome otra línea que satisfaga su destino.

   Una vez que el pasajero arriba a la parada origen, sabe que al menos un bus directo (que le asegura menor tiempo de viaje) o un bus de trasbordo pasarán por allí. De todas formas el pasajero abordará aquel que primero pase por dicha parada si es que le sirve. En caso que lleguen muchos buses a dicha parada al mismo instante, el pasajero primero seleccionará entre los directos el de menor tiempo. Si no hay directos entonces, el pasajero seleccionará entre los de trasbordo el primero que le permita hacer trasbordo y lo transporte lo más lejos posible. Para eso como en el caso (1) nuevamente se consultará la tabla de trasbordos del sistema.

Según el viaje realizado por el pasajero, hay dos alternativas:

- Si viajó mediante una línea directa: llegará a su parada destino en el tiempo dado por la suma de tiempos de los tramos que componen el recorrido a partir de la parada origen.
  Su tiempo de permanencia en el sistema está dado por el tiempo de la caminata inicial, el tiempo de espera en parada, el tiempo de viaje y el tiempo de la caminata final.
- Si viajó por línea de trasbordo, llegará a la parada intermedia (la cual es la más cercana al destino) y esperará por el primer bus de línea directa que le permita viajar a su parada destino. Su tiempo de permanencia en el sistema está dado por el tiempo de caminata inicial, dos tiempos de viaje, dos tiempos de espera en parada y el tiempo de caminata final.

La Figura 27, mediante un diagrama de flujo describe la secuencia de pasos de la nueva estrategia al momento de abordar bus:

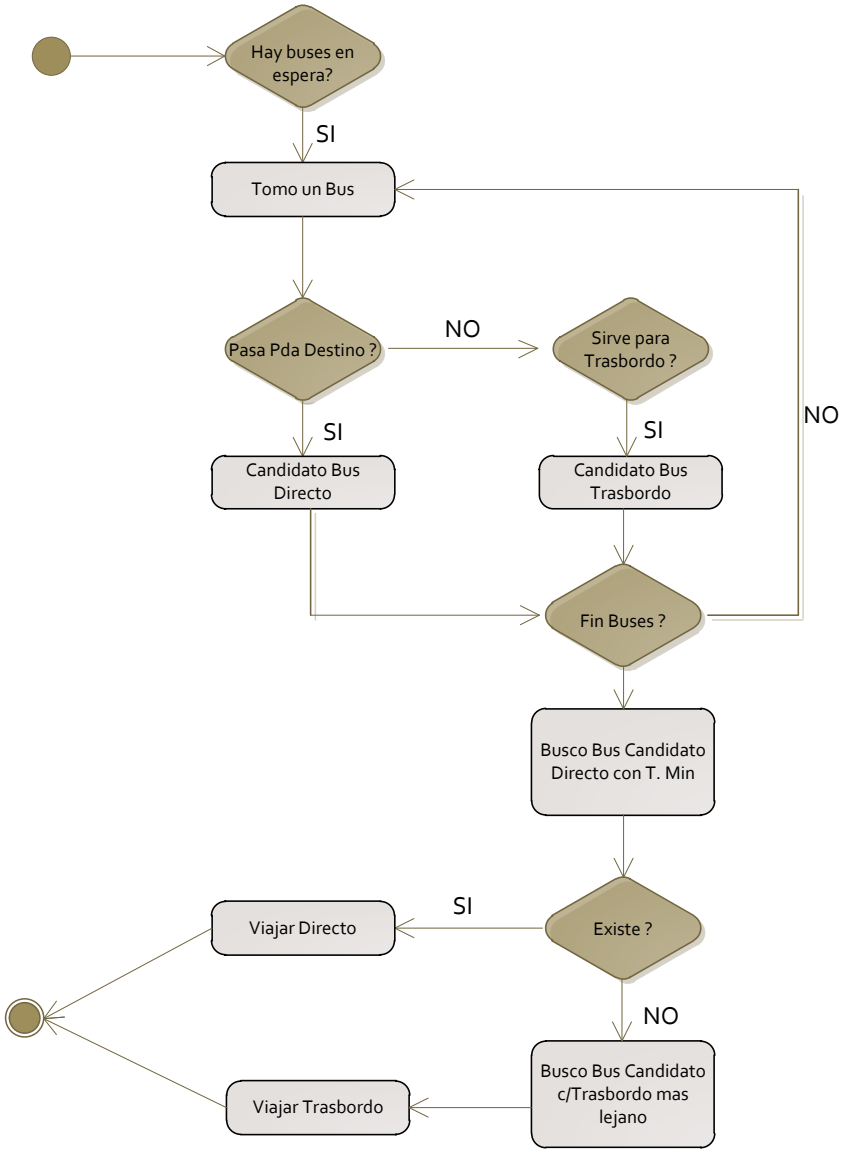

Figura 27: Diagrama de selección de bus

Nótese que en caso que el viaje sea de trasbordo el pasajero nuevamente repetirá el ciclo de espera en parada (ahora intermedia) pero abordará solamente bus directo a destino, debido a que la nueva estrategia permite a lo sumo un trasbordo.

Es importante destacar que las estrategias consideran paradas más allá del fin de recorrido circular. Si una parada se encuentra luego del fin de recorrido circular para una línea, entonces esta línea se considerará como válida para el viaje.

4.4.2. Definición de indicadores para la validación

La validación del nuevo módulo y por ende del modelo de simulación, se realizará mediante la función objetivo definida por el modelo de asignación de Baaj y Mahmassani (pág. 4 de [6]). La selección de este modelo en particular se debe a que se cuentan con resultados teóricos de la función objetivo para el caso de estudio de Rivera (mediante este modelo de asignación) y se utilizarán para la comparación con los resultados del simulador.

El modelo de Baaj y Mahmassani define dos funciones objetivo que representan los intereses de usuarios y operadores:

1. La función objetivo que representa el interés de los usuarios se define como:

$$Z1 = \sum_{i,j} d(tv + te + tt)$$

Donde *d* es la demanda de pasajeros, *tv* es el tiempo de viaje en bus, te es el tiempo de espera en parada y *tt* es el tiempo de penalización por trasbordo de los vértices *i, j*. El tiempo de penalización por trasbordo es el tiempo de espera en la parada de trasbordo. A partir de una solución *S* del TNDP los valores *tv*, *te*, y *tt* se conocerán una vez aplicado el modelo de comportamiento.

2. La función objetivo de los operadores que representa el tamaño de flota se define como:

$$Z2 = \sum_{rk} techo(fk * tk)$$

Donde *fk* es la frecuencia del recorrido *rk*, y *tk* es el tiempo que insume dicho recorrido.

El motivo de utilizar las funciones objetivo de un modelo de asignación conocido para la validación, se debe al extenso estudio realizado por distintos autores sobre este tipo de herramientas descriptivas, como por ejemplo Baaj y Mahmassani, debidamente probados y cuyos resultados se asumen confiables.

El modelo de simulación implementará las funciones objetivo y los resultados obtenidos se utilizarán como indicadores para la validación del mismo.

Descripción de la validación

El proceso de validación se realizará de la siguiente manera:

1. Se obtendrán datos de un caso de estudio con información de la red de transporte, demanda y una solución particular del TNDP.
2. Se calculan funciones objetivo Z1 y Z2 definidas por el modelo de Baaj y Mahmassani.
3. Se calcula Z1 y Z2 en el modelo de simulación.
4. Se comparan los resultados obtenidos en 2) y 3).
5. Se realiza un análisis y una validación del modelo de simulación.

Es necesario destacar que se utilizó un modelo de asignación para el cálculo del Z1 en el punto 2) debido a que no se contaba con información del tiempo de espera en paradas de los usuarios. Por tal motivo se utilizó el modelo definido en [9] para estimar dichos valores y finalmente calcular Z1.

La validación del modelo de simulación se trata en el la sección 4.7: *Validación del Simulador*.

4.5. Modificaciones del modelo SED

El presente apartado describe las modificaciones realizadas al modelo de simulación original para cumplir los objetivos descritos en el apartado anterior.

A continuación se detallan los componentes del modelo de simulación y sus respectivos cambios donde corresponde:

- Objetivos: Los objetivos del modelo de simulación se mantienen del proyecto [3]:

  "Modelar un sistema de TPUC donde se represente la interacción de pasajeros con un conjunto de líneas de ómnibus teniendo en cuenta la información de la red vial y la demanda de transporte público para un escenario dado. "

  Nuestra tarea consiste en implementar un nuevo módulo que represente una alternativa de interacción pasajeros-líneas.
- Redefinición de hipótesis iniciales:
  - Líneas de buses: las cuales están asignadas a recorridos, según sea su tipo y a una frecuencia dada. Si el tipo es circular, entonces poseen solo un recorrido asignado, mientras que las de ida y vuelta, tienen un recorrido asociado para ir y otro para regresar (podrían ser el mismo). No se hicieron modificaciones.
  - Los recorridos: son un conjunto de tramos en secuencia, por los cuales circulan los buses. A su vez, los tramos poseen dos nodos viales en sus extremos, algunos de ellos son paradas donde los pasajeros abordan y bajan de los buses. Es necesario recordar que el cambio introducido por las paradas, indica que todos los buses consideraran aquellas paradas que declaren en sus recorridos. Lo que significa que un recorrido puede considerar un nodo vial como parada, mientras que otro recorrido no. Esto difiere significativamente de la versión anterior, donde cada parada era una parada para todos los recorridos que la incluyeran.
  - Los buses: los cuales pertenecen a sus respectivas líneas y utilizan los recorridos asociados a cada una para realizar sus trayectos. Cada bus tiene una capacidad máxima

de pasajeros sentados y parados, y una única puerta por donde suben y bajan pasajeros. Además, al finalizar el recorrido deberá esperar un tiempo dado por la frecuencia de la línea para determinar si debe salir nuevamente o esperar hasta que se cumpla dicho tiempo. En caso que el bus no cumpla con la frecuencia impuesta por la línea, entonces el sistema generará otro nuevo bus para mantener dicha frecuencia. No se hicieron modificaciones.

  - Los pasajeros: son quienes realizan los viajes según especifica la demanda, utilizando los buses del sistema. Se generan en los centroides origen, realizan una caminata a la parada origen y seleccionan mediante una estrategia cómo realizar el viaje. Una vez que arriban a su parada destino, caminan hacia el centroide destino y son removidos del sistema. La implementación de un nuevo módulo de comportamiento incide directamente en cómo realizan los pasajeros la selección de las líneas para realizar su viaje.

- Los datos de entrada del modelo SED son un conjunto de archivos con la siguiente información:
  - Red vial o grafo con información espacial de nodos viales, centroides, puntos de demanda, caminatas y aristas. La nueva disposición de las paradas hace que esta información presente cambios y el simulador deberá adaptarse a ellos. Este archivo y sus cambios se detalla en la sección 3.5: *Archivos de entrada y salida*.
  - Matriz de demanda de pasajeros en el sistema. No tiene cambios. Este archivo se detalla en la sección 3.5: *Archivos de entrada y salida.*
  - Información sobre los distintos recorridos de las líneas de buses. Presenta cambios en los recorridos que tienen la información de las paradas. Este archivo y sus cambios se detalla en la sección 3.5: *Archivos de entrada y salida.*
  - Información de las frecuencias de las líneas del sistema. No presenta cambios. Este archivo se detalla en la sección 4.6: *Archivos de Entrada y Salida*.
- Los datos de salida que brinda el modelo SED, están dados por:
  - Tiempo de viaje de los pasajeros en el sistema.
  - Tiempo de espera de pasajeros en parada.
  - Utilización de buses del sistema.
  - Cantidad de pasajeros transportados.
  - Porcentaje de uso de cada línea.
  - Tamaño de flota: Objetivo de los operadores.

  Todos estos datos de salida se detallan en la sección 4.6: *Archivos de Entrada y Salida*.
- Los factores del modelo están dados por la demanda de pasajeros y la solución S del modelo de optimización, definida como un conjunto de recorridos y frecuencias, ver el capítulo 2: *Marco Teórico*. Mientras que las respuestas de interés son: la función objetivo de los usuarios y la función objetivo de los operadores.

El diseño del modelo de simulación se obtuvo del proyecto de grado **[3]** y solamente se debe incorporar el nuevo módulo de estrategia. La implementación del modelo de simulación ha sido desarrollada utilizando el lenguaje de programación C++ **[10]** y la biblioteca para SED EOSimulator **[11]**.

La

Figura 28 muestra el diagrama de clases del modelo de simulación original obtenida de **[3]**:

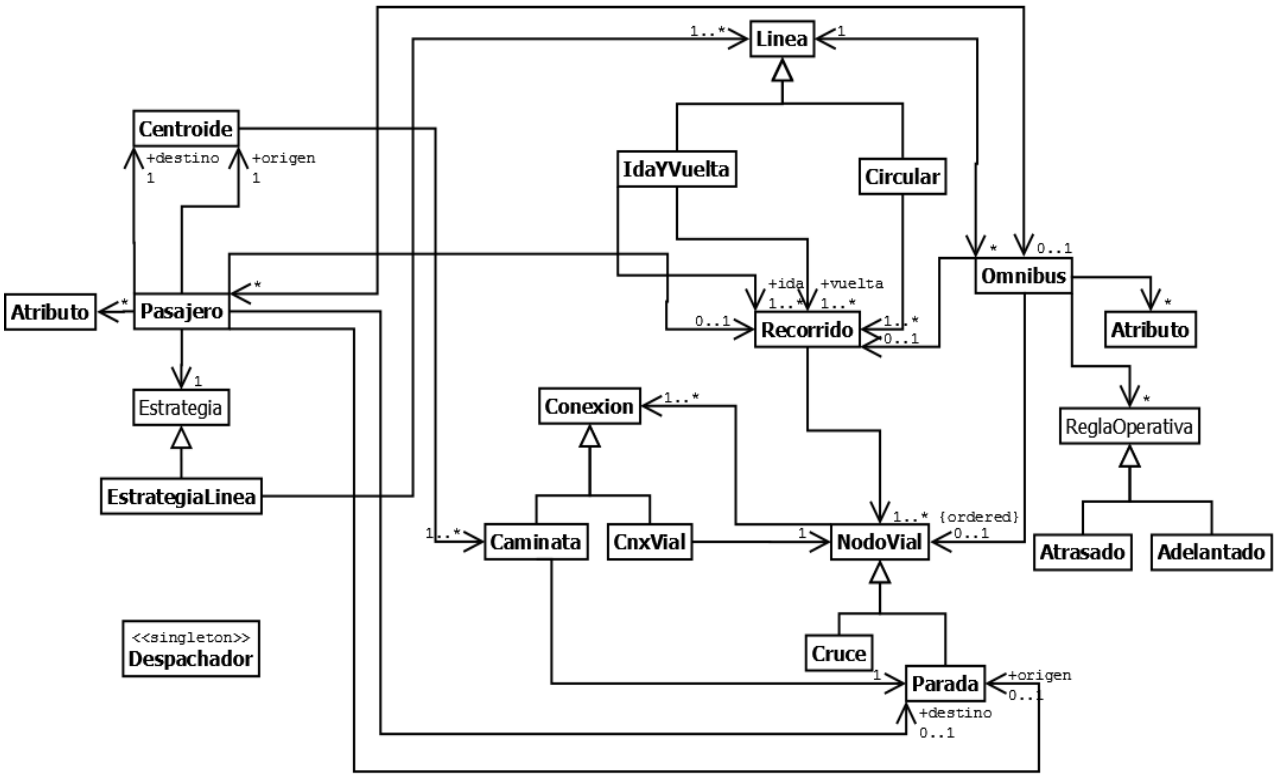

Figura 28 : Diagrama de clases del simulador *PublicTransport*

A partir del conjunto de clases utilizado para representar el modelo de simulación, se observa en la figura cómo se desarrolló cada concepto del sistema de transporte de pasajeros.

4.5.1. Elementos de modelado

A continuación se describen los elementos más relevantes que integran el modelo de simulación.

- Líneas y Buses

  Existen dos tipos de líneas en el sistema, circular e ida y vuelta. Las líneas poseen sus recorridos asociados: si es circular entonces tendrá un único recorrido mientras que si es de ida y vuelta, tendrá dos recorridos asociados.

  Cada recorrido consta de un conjunto de nodos viales que lo conforman, donde cada uno puede ser un cruce o una parada. A su vez, cada nodo vial es parte de una conexión (arista) que puede ser una caminata o una conexión vial (tramo). Por lo tanto, el tiempo que demore cada bus de la línea en realizar el recorrido, está dado por la suma de los tiempos de sus conexiones viales.

  Si la línea es de ida y vuelta, al llegar al último nodo vial de su recorrido cambiará el sentido: si es de ida, pasará al de vuelta y viceversa. El tiempo de cambio de sentido dependerá de la

frecuencia de la línea. Si la línea es circular, entonces el último nodo vial de su recorrido coincide con el primero.

Las líneas tienen una colección de buses que realizan el/los recorrido/s que esta tiene asignada. Por lo tanto, el/los recorridos también serán accesibles desde los buses. Además, cada bus tiene una referencia al último nodo vial visitado.

Para mantener un control de la totalidad de las líneas del sistema, se cuenta con una clase llamada despachador. Para ello basado en la información de las frecuencias de cada línea, esta clase es la encargada de proveer buses a las líneas de forma tal de respetar la frecuencia asignada. Debido a que no hay restricción de buses en el sistema, el despachador siempre contará con un bus disponible para asignar a una línea. Para las líneas de ida y vuelta el despachador asegurará que al momento de cambiar el recorrido entre la ida y la vuelta, se respete un tiempo de "penalización" configurable.

- Pasajeros

Los pasajeros se modelan como objetos o entidades que tienen un periodo de vida en el sistema. Para realizar el viaje a destino, necesitan de una estrategia de selección de línea. El sistema cuenta con el patrón de diseño *Estrategia* [12] de forma de poder manejar a través de una interface uniforme, distintos tipos de comportamientos de los pasajeros. De esta forma se pueden implementar distintos tipos de estrategias, agregando nuevos módulos donde cada uno defina un nuevo comportamiento. La Figura 29, muestra la estructura del patrón aplicada al modelo.

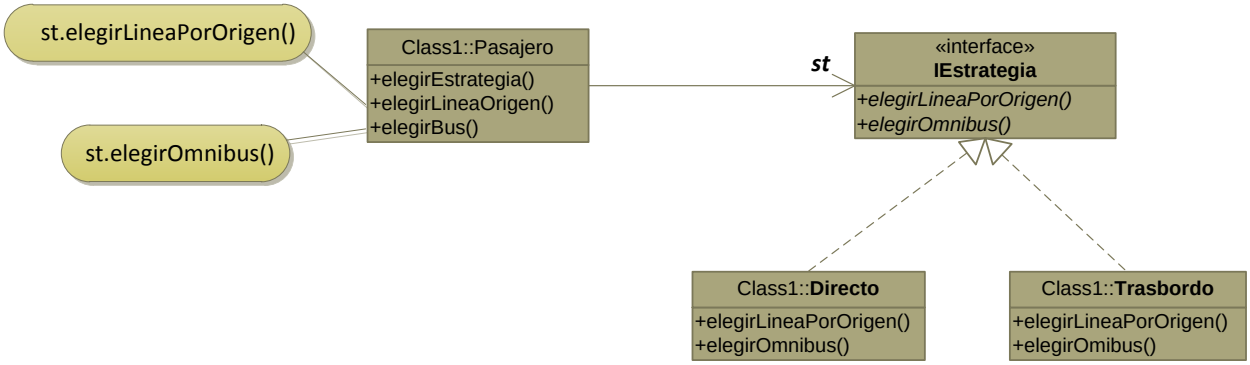

Figura 29: Patrón *Estrategia* aplicado al contexto

Las clases involucradas al aplicar el patrón *Estrategia* al modelo son:

- Pasajero: esta clase es la responsable de seleccionar su propia estrategia concreta y referenciarla cuando sea necesario. Para seleccionar su propia estrategia invocará el método *elegirEstrategia()*.
- IEstrategia: interfaz que define el comportamiento de las estrategias concretas. Toda operación declarada en ella, deberá ser implementada por cada estrategia. Fundamentalmente define dos operaciones:
  - *elegirLineaPorOrigen(p: Pasajero)*: Se aplica en la primera parte de la estrategia: "Seleccionar parada origen", cuando el pasajero está en el centroide origen (ver pág. 54).

- *elegirOmnibus(l: listaBuses, p: Pasajero)*: Se aplica en la segunda parte de la estrategia: “Seleccionar bus”, cuando el pasajero ya está en la parada origen (ver pág. 56).

- Directo, Trasbordo: estrategias concretas donde cada una encapsulará un comportamiento diferente para la lógica de los pasajeros.

El simulador ya provee una estrategia concreta por defecto la cual es similar a la nueva estrategia pero con la salvedad que no permite viajes mediante trasbordos, solamente directos. Nuestra estrategia no solo permite viajes por trasbordos sino que también viajes directos.

4.5.2. Distribuciones

Para modelar la aleatoriedad de arribo de pasajeros, tiempo de viaje de buses por tramos (tiempo entre nodos viales) y las puertas de descenso de pasajeros de los buses se utilizan distribuciones de probabilidad.

Es natural suponer que los arribos de individuos al sistema son independientes entre sí, aleatorios y pueden ocurrir en cualquier momento de la simulación. Por lo tanto, dicho comportamiento es representado por el simulador mediante una distribución exponencial negativa. De esta forma se describe el tiempo que transcurre entre ocurrencias aleatorias, y particularmente en este caso, que son arribos de pasajeros.

El tiempo de arribos de pasajeros entonces, se modelará con una distribución exponencial negativa con media igual a la demanda entre centroide origen y destino proporcionada en los datos de entrada del simulador.

El tiempo de viaje de los buses entre dos nodos viales no es el mismo para todos los casos. Si bien existe un tiempo de viaje fijo que es un dato de entrada, en la realidad existen distintos tipos de demora que pueden hacer que el tiempo varíe de dicho valor fijo. Aquellas actividades donde la variación es aleatoria alrededor de un valor medio, son usualmente aproximadas por la distribución Normal. Esto implica entonces que los tiempos insumidos por los buses son el resultado de una distribución Normal cuya media es igual al tiempo entre nodos viales (dato de entrada) y su desviación estándar se mantiene en 1/5 de la media. Si hay valores negativos entonces el resultado es cero.

Para el descenso de pasajeros del bus, estos decidirán en base a una distribución uniforme sobre el total de las puertas del mismo. Cada bus mantiene una distribución uniforme donde cada pasajero determina por cual puerta desciende con igual probabilidad.

4.5.3. Diagrama de actividad y eventos del sistema

El diagrama de actividades de un modelo de simulación a eventos discretos es una representación que indica el flujo o ciclo de vida de las entidades. En este caso el cambio introducido al momento de agregar una nueva estrategia introduce una leve modificación en el diagrama de actividades del proyecto anterior. Basándonos en dicho diagrama, modificamos el mismo con los cambios necesarios introducidos en el modelo debido a los trasbordos.

El diagrama mostrado en la Figura 30 describe el ciclo de vida de las entidades bus y pasajero en el sistema y sus interacciones con otras entidades. El cambio realizado del diagrama original se puede visualizar a través de la transición en color rojo.

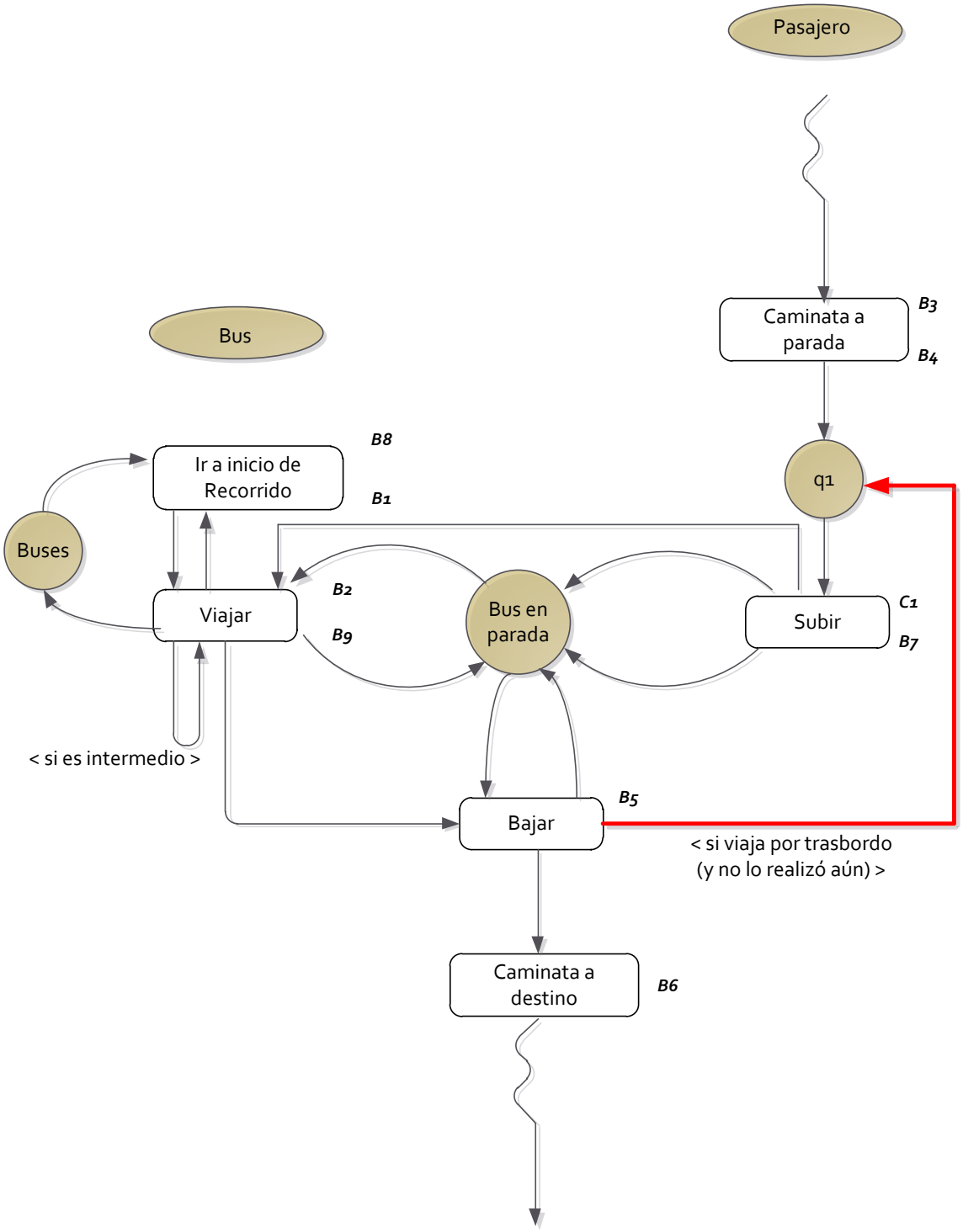

Figura 30: Diagrama de actividades

Los pasajeros en el sistema son creados en cada centroide y luego se dirigen (estrategia mediante) a su parada de origen. Allí los pasajeros son encolados en la cola de la parada: q1, esperando por abordar un bus. Al momento de arribar un bus en parada que le sirva al pasajero

(nuevamente lo decidirá su estrategia) este se desencola de q1, sube al bus en parada y realiza su viaje hacia la parada destino o a la parada intermedia. Una vez llegado a la parada deseada, el pasajero baja y se dirige al centroide destino o se dirige a la parada intermedia de trasbordo para nuevamente ser encolado en otra q1 (línea roja) y esperar el bus directo.

Los buses al momento de ser creados proceden a inicializar su recorrido, dependiendo del tipo de línea. Luego, comienzan el viaje a través de los nodos viales que componen el recorrido ya sea circular o ida y vuelta. Al arribar a una parada de su recorrido, bajan y suben los pasajeros para luego retomar el viaje y este comportamiento se repite para todas sus paradas. Una vez que el bus arriba al fin del recorrido dado por su último nodo vial, entonces el bus esperará para comenzar nuevamente su viaje según lo indique la frecuencia de la línea.

Los eventos utilizados para modelar el sistema están indicados en el diagrama de actividades. Ellos son los encargados de (a través de su lógica) realizar todas las interacciones en la simulación. La Tabla 16 describe cada evento que utiliza el sistema:

Evento	Tipo	Acción
*{C1}***Subida Pasajero Bus**	Condicionado	Gestionar para todas las paradas, la subida de los pasajeros a los buses.
*{B1}***Comienzo Recorrido Bus**	Fijo	Encargado de la lógica de avance del bus por los nodos viales de su recorrido.
*{B2}***Arribo Bus a Nodo**	Fijo	Refleja el momento en que bajan los pasajeros del bus (en caso que sea su parada destino).
*{B3}***Comienzo Viaje Pasajero**	Fijo	Encapsula el comienzo de viaje del pasajero.
{B4} **Arribo Pasajero a Parada**	Fijo	Encola el pasajero en parada.
{B5} **Baja Pasajero Bus**	Fijo	Procesa el momento en que el pasajero baja en una parada.
{B6} **Arribo Pasajero Centroide**	Fijo	Procesa el momento en que el pasajero llega a su centroide final.
{B7} **Fin Subida Pasajeros**	Fijo	Indica que el pasajero ya subió al bus.
{B8} **Frecuencia Bus**	Fijo	Controla que los buses viajen por el sistema de acuerdo a su frecuencia establecida.

Tabla 16: Descripción de eventos

En el *Apéndice E - sección 2* se proporciona una mejor descripción de la lógica de cada evento a través de diagramas.

4.5.4. Trasbordos

Se mantiene una estructura de trasbordos para que el sistema compruebe rápidamente si existe trasbordo de líneas dado un par de paradas. Para obtener las líneas desde una parada origen O y una destino D si no existe una línea directa se consulta esta estructura de trasbordo. El sistema deberá determinar aquellas paradas T, de forma tal que:

1. Existe un recorrido desde O hacia cualquier parada T.
2. Para el subconjunto de T que satisface 1) existe otro recorrido que las comunica con D.

Se debe contemplar la restricción de orden para que el ordinal de O sea menor que el de T dentro del recorrido y a su vez el ordinal de T sea menor que el de D. Por lo tanto, la estructura almacenará para la pareja (O, D) una lista de parejas (id recorrido, parada de trasbordo T). De esta forma se puede saber todas las líneas que pasan por T (pasando por O previamente) y que pasan por D.

El siguiente pseudocódigo explica la creación de dicha estructura:

1. Para cada pareja de paradas (O, D) obtener el conjunto de líneas *Ls* tal que las líneas pasan por O, pero no pasan por D.
2. Para cada línea li de *Ls*, según sea ida - vuelta, o circular; se obtienen las paradas *Ps* (li) = [siguiente (O), resto (recorrido)] de su recorrido.
3. Para cada parada de p de *Ps*, se consulta si existe alguna línea que pase por p y llegue a D.
4. En caso afirmativo de la condición 3) entonces se almacena en la estructura de trasbordos en la posición dada por (O, D) = <recorrido de li, parada de trasbordo p>.
5. ¿Fin? No: volver a 1.

Para mejor comprensión sobre el funcionamiento se describe un ejemplo del uso de dicha matriz. Se supone que se desea ir de la parada 3 a la parada 13 y no existe un recorrido directo que las comunique. Aun así, existen dos paradas de trasbordo que sí permiten el viaje: la número 2 y la número 8. La Figura 31 muestra una estructura auxiliar que mapea índice de matriz con identificador de parada:

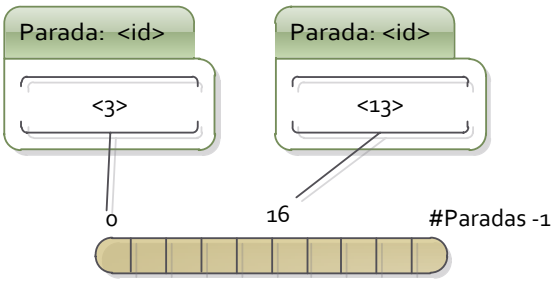

Figura 31: Estructura auxiliar de trasbordos

Para este ejemplo, la parada 3 estará en el índice 0 y la parada 13 en el índice 16. A continuación se describe cómo se organiza la matriz de trasbordos mediante la Figura 32:

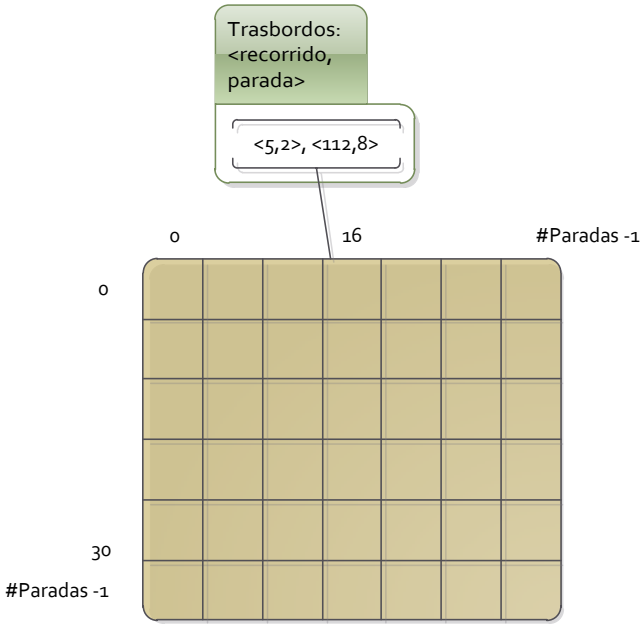

Figura 32: Matriz de trasbordos

En la posición [0,16] de la matriz de trasbordo, hay una lista con dos posibles candidatos a trasbordo: la línea con recorrido número 5 con parada número 2; y la línea con recorrido número 112 con parada número 8. Por lo tanto, el pasajero sabrá que: desde la parada 3 existen dos recorridos que lo comunican con a las paradas 2 y 8, y desde ahí, puede alcanzar la parada 13 en viaje directo. La Figura 33 utiliza la información alojada en la matriz para el ejemplo:

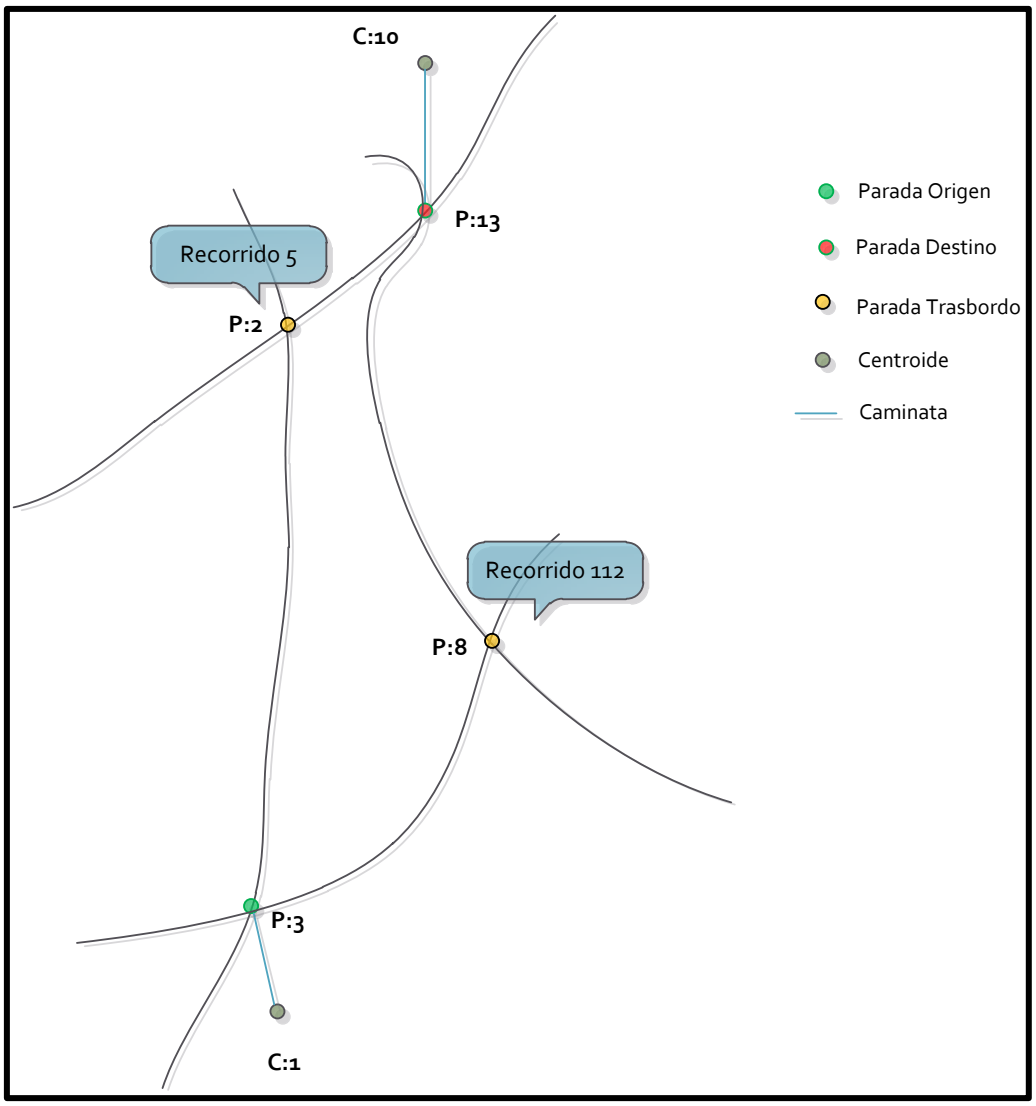

Figura 33: Ejemplo de uso de trasbordo

4.5.5. Salida Gráfica

La salida gráfica es una herramienta muy importante en varios aspectos. Principalmente para que el usuario comprenda el funcionamiento del modelo y por ende del sistema; y para realizar verificaciones de forma gráfica sin utilizar un alud importante de datos. En ciertas situaciones es más rápido visualizar un error en la simulación a nivel gráfico que en los reportes o histogramas. En estos tipos de sistemas, contar con una adecuada herramienta de experimentación es muy útil.

Se decidió que la salida gráfica en *SDL* [13] del modelo de simulación, incluida en *EOSimulator* [11], sea configurable a partir de la invocación al simulador. Por lo tanto, la misma puede ser deshabilitada y la información de la simulación se guardará en un archivo log en disco para que un módulo externo (con su propia salida gráfica) la interprete. El principal motivo es para que la simulación pueda ser utilizada “desacoplado de la vista”, por lo tanto si no se hace validación visual, el tiempo de ejecución se reduce notablemente.

4.6. Archivos de Entrada y Salida

El simulador cuenta con un conjunto de parámetros de entrada que son necesarios para realizar la simulación. Son una serie de archivos de texto, que proveen toda la información sobre el modelo que se desea simular; estos archivos pueden provenir del módulo de manipulación de igoR-tp o pueden ser generados independientemente por el usuario. Se deberán ubicar en la ruta: */bin/entrada/* del directorio del simulador.

4.6.1. Entradas

Como se especificó en la sección 4.2: *TNDP y el Modelo de Simulación,* el simulador necesita entrada de datos para su ejecución.

El conjunto de archivos de entrada es:

- Red vial: La descripción de la misma se encuentra en el archivo *grafo.txt*. El simulador utilizará dicha información para generar el conjunto de clases y estructuras para la representación de dicha Red. Esto incluye a puntos de demanda, centroides, nodos viales, paradas, caminatas y calles. El formato de este archivo puede verse en la sección 3.5.2.
- Demanda: La información de la demanda de pasajeros en un horizonte de tiempo dado puede obtenerse del archivo *demanda.txt*. El simulador generará pasajeros entre los centroides -que indique el archivo- acorde a la frecuencia de arribos especificada. La descripción detallada de este archivo puede verse en la sección 3.5.1.
- Recorridos: El conjunto de recorridos por los cuales circularán las líneas es una entrada al simulador. El simulador construirá el conjunto de recorridos de la Red de Vial a partir del archivo lineas.solucion. La descripción detallada de este archivo puede verse en la sección 3.5.3.
- Líneas: Es necesario contar con un mapeo de las líneas y los recorridos por los cuales circulan. El archivo *lineas.txt* define para cada línea su tipo, su/s recorrido/s y su frecuencia.

  Cada fila del archivo, que se asocia a una línea, tiene la siguiente estructura:

```
Tipo   id_recorrido      espera            frecuencia
```

La Tabla 17 describe cada concepto de la fila del archivo:

Concepto	Descripción
Tipo	Indica el tipo del recorrido, 0 si es ida y vuelta y 1 si es circular. Si es ida y vuelta entonces el atributo id_recorrido está compuesto por un valor para el recorrido de ida y otro para el recorrido de vuelta.
Id_recorrido	Indica el número de recorrido asignado a la línea. Se corresponde con la posición del recorrido en el archivo lineas.solucion, comenzando en 1.
Espera	Tiempo en segundos que demora una línea de ida y vuelta en cambiar de recorrido de “ida” a “vuelta”.
Frecuencia	Tiempo en segundos que representa la salida de un nuevo bus de la línea.

Tabla 17: Descripción conceptos de lineas.txt

4.6.2. Salidas

El conjunto de datos de salida para el análisis de resultados se dividen en dos grupos, reportes e histogramas. Ambos se ubican en subdirectorios dentro de la carpeta */bin* del simulador.

Carpeta Reportes

Dentro de este grupo, podemos encontrar distintos archivos con información específica para cada acción considerada importante. De esta forma se facilita la verificación a través de la información que producen los reportes. Los archivos de texto son:

- Incidencias
  El archivo Incidencias.txt contiene aquellas situaciones anómalas que pueden impactar negativamente en la simulación. El archivo presenta el siguiente formato:

  Si no existen líneas que puedan transportar pasajeros entre dos centroides el archivo tendrá información de este tipo al comienzo:

  ```
  ATENCION: New Pasajero FAIL: [CentroideO=81, CentroideD=65]
  ```

  Esta advertencia indica que se generó un pasajero acorde a la información provista por la demanda, pero éste no pudo seleccionar ninguna línea. Por lo tanto, se creó un pasajero "zombie" que no tiene destino y sobrecarga el simulador. Es un claro ejemplo de que la información provista por el modelo no es correcta.

  Si un pasajero espera un tiempo excesivo en parada, se registra una incidencia similar a la siguiente:

  ```
  2138.94 ATENCION: tiempo de espera en parada excesivo  ID_PASAJERO:
  255 ORIG: 32 DEST: 17 TIEMPO ESPERA: 1166.55 LINEA: 8
  ```

  Se indica el tiempo de simulación que ocurrió el incidente, junto a los datos relevantes para su depuración.

- Validación
  El archivo Validación.txt contiene todos los viajes realizados por todos los pasajeros del sistema. Es muy útil para hacer seguimientos, controlando trayectos y tiempos. Un ejemplo es el siguiente:

  ```
  ********************************************
  ** DIRECTO
  ** Pasajero ID = 360
  ********************************************
  ** Tiempo Espera Parada Origen :    150.847
  ** Linea ID para Tramo1 :           11
  ** Tiempo Viaje Tramo1 :            771.074
  **
  ** Total :                          921.921
  ********************************************
  ** Total TiempoViajeSinCaminatas :  921.921
  ********************************************
  ```

Como se puede observar, para un pasajero dado del sistema, se posee toda la información de su viaje y los tiempos de espera en parada. Si el pasajero hizo viaje mediante trasbordo entonces aparecerán dos bloques de información con el mismo id de pasajero pero con distintos datos.

- Reportes
  Como explica el documento del proyecto de grado [3], existen dos tipos de archivos de reporte que se brindan al operador. Un reporte en cualquier momento de la simulación al presionar la tecla R y otro reporte al final de la simulación. Estos reportes realizan un sumario con información al final de la simulación (para cada iteración) sobre tiempos de viaje totales, pasajeros generados, pasajeros transportados, cantidad de buses generados y una descripción sobre cada línea junto con su estado.

Carpeta Histogramas

Los histogramas son una representación visual de una distribución a lo largo de la simulación. Son fundamentales para estudiar el comportamiento de los factores. En particular, del proyecto anterior, se mantuvo los histogramas provistos llamados “histograma de observación”. Dicho histograma cuenta la cantidad de veces que un valor dado fue registrado por una variable.

En el simulador, se proveen los siguientes histogramas:

- tiempos que los pasajeros permanecían esperando en las paradas para subir a un bus.
- tiempos que esperan los buses en la parada mientras suben y bajan pasajeros.

Log de simulación

Todos los cambios de estado de las entidades durante una corrida de simulación se registran en un archivo *XML*. Este archivo fue creado con la finalidad de poder reconstruir por parte de una aplicación externa la animación de las entidades de dicha corrida.

Una descripción detallada de este archivo se puede encontrar en el *Apéndice F.*

4.7. Validación del simulador

Llegado a este punto se dispone de un modelo de simulación modificado el cual necesita ser verificado y validado para determinar que los objetivos descritos en la sección 1.2 se han cumplido. Como se menciona anteriormente el nuevo modelo de simulación dispone de una nueva estrategia de pasajeros con trasbordos y además implementa nuevos medidores para los indicadores Z1 y Z2.

Para verificar el modelo de simulación es necesario comprobar que la nueva estrategia de trasbordos permite que todos los pasajeros generados en una simulación arriben a su destino.

Para validar el modelo de simulación existen distintas técnicas que estudian los efectos producidos en la conducta del modelo cuando se testean distintas hipótesis sobre el mismo. Una de ellas, se basa en ejecutar la simulación alterando los valores de sus variables de decisión o factores, de forma de comprender y analizar la sensibilidad del modelo ante los cambios.

Dicha técnica es llamada análisis de factores (sensibilidad) y debe ser realizada con una correcta planificación y cuidado. El análisis de sensibilidad es un proceso necesario al momento de validar un modelo, ya que busca asegurar que el mismo mantiene un nivel razonable de la calidad de sus resultados cuando se alteran factores e hipótesis [8].

Otra ventaja que presenta la técnica de análisis de factores, es que permite simplificar el modelo, ya que si este no experimenta cambios significativos de respuesta al alterar sus factores, se puede considerar omitir el recurso o la actividad asociada al cambio [8].

Debido a que en la mayoría de los casos es imposible cubrir todas las posibles combinaciones de factores, dicho análisis debe ser cuidadosamente planificado y la cantidad de factores a analizar debe ser reducida (la complejidad del análisis crece de manera exponencial con la cantidad de factores).

A continuación se detallará un caso de estudio pequeño llamado *Aldea*, este caso de estudio es lo suficientemente complejo para probar la nueva estrategia de trasbordos y lo suficientemente simple para calcular los indicadores Z1 y Z2 en forma teórica. Dichos valores teóricos pueden ser comparados con los valores obtenidos con las nuevas implementaciones de los medidores de indicadores Z1 y Z2 en el modelo de simulación.

Los factores elegidos para llevar a cabo el análisis, las pruebas, las hipótesis de pruebas y resultados esperados se expondrán en un plan de pruebas en un capítulo 4.7.2: *Plan de Pruebas*.

Por último es de interés del usuario de este proyecto experimentar con el modelo de simulación del caso de estudio de la ciudad de Rivera y contrastar los valores de los indicadores Z1 y Z2 con los valores calculados en [5].

4.7.1. Caso de estudio

Se construyó un caso de estudio para la validación del modelo de simulación con el objetivo de estudiar el comportamiento de los pasajeros al utilizar una nueva estrategia con posibilidad de viajes mediante un trasbordo.

El modelo se diseñó con tamaño reducido para analizar en detalle como impacta el cambio de estrategia a nivel de tiempos de viaje y cómo evoluciona la cantidad de buses en el sistema. La ventaja del tamaño del caso de estudio, permite no perder efectividad ya que se hace un seguimiento riguroso de cada entidad pasajero que circula por el mismo de una forma más simple. Otra ventaja es que la recolección de datos estadísticos es menor y de esta forma se evita un alud de resultados.

Como regla inicial se adoptó que el modelo fuera lo más simple posible y que produjera resultados verificables. De esta forma, se obtuvieron distintos tipo de información de forma teórica como distancias, tiempos de espera, tiempos de viaje y trasbordo; y se compararon contra la información empírica que nos brindó el simulador en sus salidas. El caso de estudio para la validación se observa en la Figura 34.

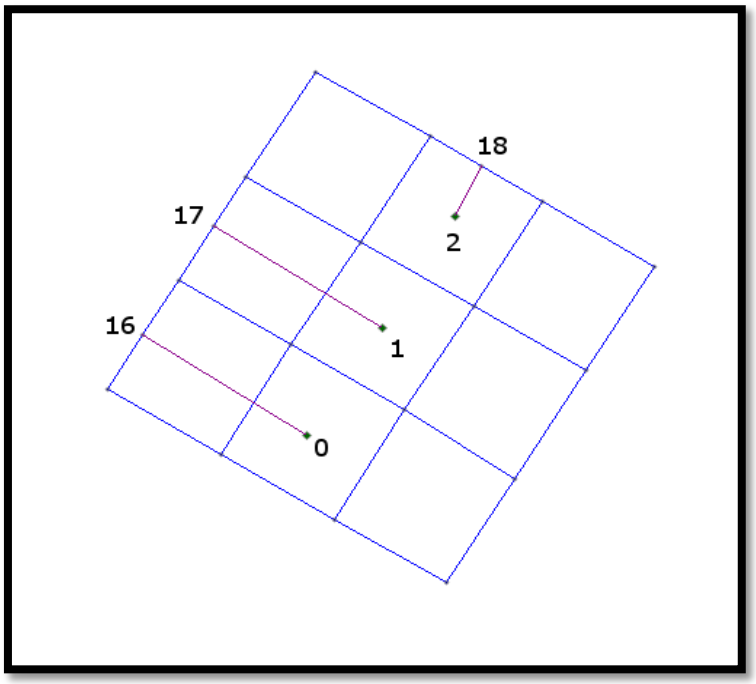

Figura 34: Modelo reducido de 9 manzanas y 3 paradas

Se construyó una ciudad que consta de tres zonas cuyos centroides se enumeran como 0, 1 y 2 respectivamente. A su vez la ciudad cuenta con tres paradas conectadas cada una por una caminata con su respectivo centroide. Las paradas se enumeran como 16, 17 y 18.

El objetivo en las siguientes etapas es analizar el comportamiento del sistema cuando los pasajeros viajan desde la parada 16 a la 18 utilizando viajes directos o de trasbordo (mediante la parada 17).

Sobre este modelo de ciudad se definen una serie de líneas de transporte que permiten viajar desde la parada 16 hacia la parada 18 utilizando opcionalmente la parada 17 como parada de trasbordo. Las tres líneas, una circular y dos de ida y vuelta con las siguientes frecuencias son:

- Línea 1 - circular con un intervalo de 540 segundos.
- Línea 2 - ida y vuelta con un intervalo de 188 segundos.
- Línea 3 - ida y vuelta con un intervalo de 450 segundos.

A los efectos de cálculos de tiempo de viaje de la simulación solo se utilizaron las idas en los recorridos de ida y vuelta. Las vueltas no fueron consideradas debido a que no hay pasajeros que realicen su viaje a través de un recorrido de vuelta. Los recorridos de las tres líneas se observan en las Figura 35, Figura 36 y Figura 37:

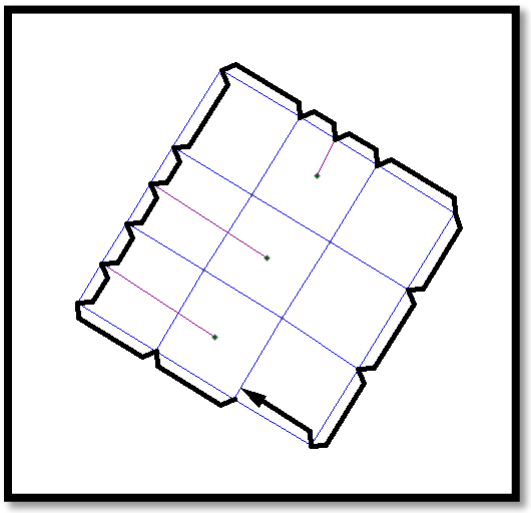

Figura 35: Recorrido circular de Línea 1

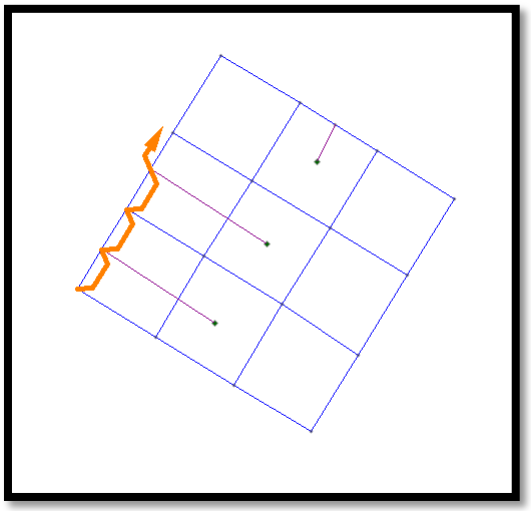

Figura 36: Recorrido ida de Línea 2

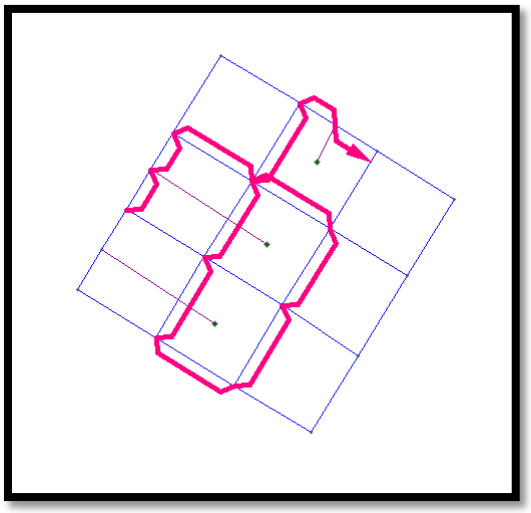

Figura 37: Recorrido de ida de Línea 3

La demanda de pasajeros es definida únicamente entre los centroides 0 y 2 con una ventana horaria de 6 horas y un total de 100 pasajeros transportados en dicha ventana. El valor de la demanda es de 0,277777 arribos por minuto (0,00463 arribos por segundo).

La velocidad de los buses es constante de 10 kilómetros por hora. Junto a las distancias dadas en la geometría del ejemplo, se calculan los costos (en segundos) que insumen los viajes entre paradas para cada línea los cuales se detallarán mas adelante.

Cálculo de los indicadores Z1 y Z2

El artículo: *Optimal Strategies: A new assignment, model for transit networks* de Spiess y Florian [9], propone un nuevo modelo de asignación para un conjunto de líneas fijas, donde el pasajero puede viajar al destino en el menor tiempo esperado. En el caso que el tiempo de espera en una parada depende únicamente de una combinación de frecuencias se traduce como un problema de programación lineal.

Se analizó el caso de estudio utilizando algunos de los conceptos manejados por dicho modelo de optimización sobre todo para el análisis de las rutas de transporte público y cálculos de tiempos de espera de los pasajeros en las paradas. Es necesario aclarar que para los cálculos no se buscó el camino óptimo sobre el caso de estudio de forma de reducir los tiempos de viaje de los pasajeros, como es la propuesta de [9]. Por lo tanto, no se obtuvieron mediciones teóricas rigurosas para la comparación de los datos sino que se consideraron como cotas superiores a los valores óptimos debidos a que son heurísticas.

Primero se definió el concepto de red de transporte *T*, la cual es un conjunto de líneas de tránsito donde cada una está compuesta de una secuencia de nodos donde los pasajeros llegan y salen. También se definió el concepto de caminata, en las cuales cada una conecta dos nodos. Los costos asociados con cada caminata o tramo de una línea de transporte público eran tiempos conocidos. La red de transporte se describe en la Figura 38:

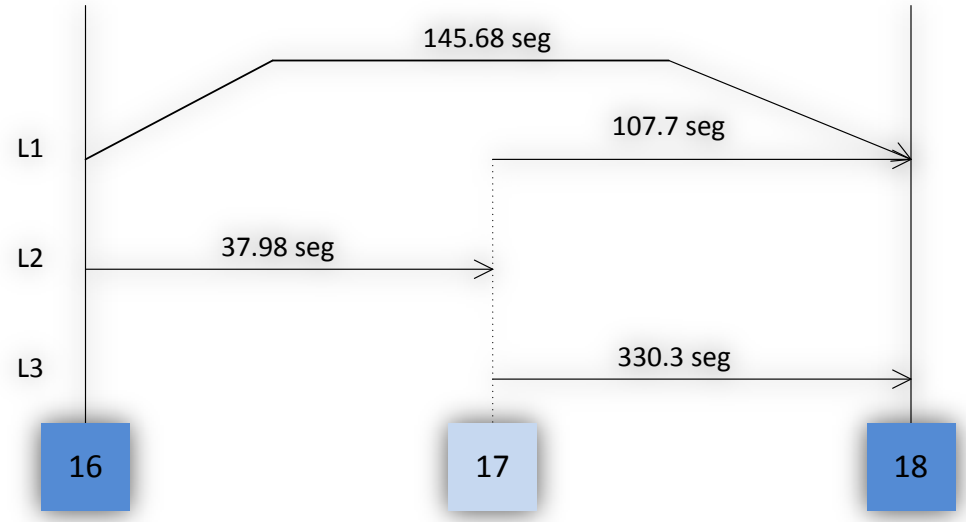

Figura 38: Red de tránsito del modelo

De la red se observan las combinaciones de líneas entre las paradas y los tiempos que insumen los recorridos entre éstas por cada línea obtenidos de la geometría del caso de estudio. De esta red se puede apreciar el caso de trasbordo donde un pasajero puede viajar por la línea 2 hasta la parada 17 y luego optar por la línea 1 o la línea 3 para llegar a la parada 18.

Debido a que las formulaciones en las que se basa de Spiess y Florian [9] para la red de transporte no identifica explícitamente las líneas, sino que utiliza el concepto de arista o link, la terminología asociada a las líneas es dejada de lado (como esperar por un vehículo determinado de una línea). El modelo de asignación descrito en [9] define seis componentes de un viaje, en forma de aristas, dentro de una red vial:

1. Caminata de centroide origen hasta parada.
2. Espera en parada.
3. Viaje en bus.
4. Descenso de bus.
5. Caminata entre paradas.
6. Caminata de parada hasta centroide destino.

Nuestro modelo consideró los puntos 2, 3 y 4 para la representación del grafo de la red de transporte. Por lo tanto, dichos puntos fueron los únicos tenidos en cuenta para la realización de grafo asociado y eran las aristas del grafo. El grafo minimizado de nuestro modelo inicial, se describe en la Figura 39.

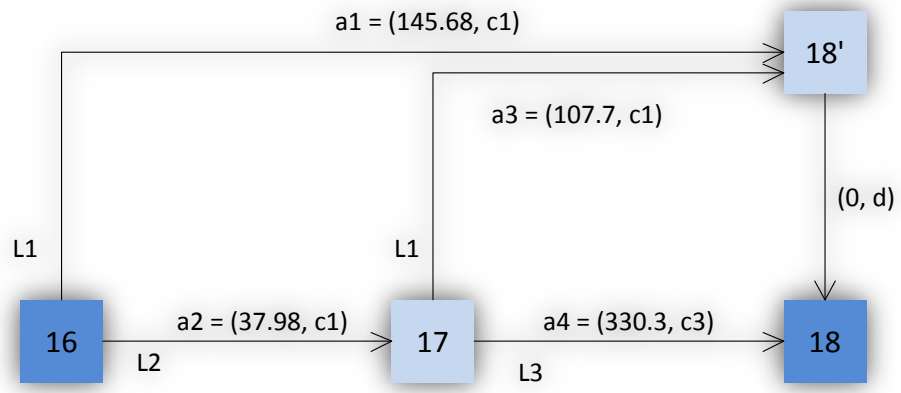

Figura 39: Grafo de aristas minimizado.

La relación entre tramo del modelo y arista del grafo minimizado se describe en la Tabla 18.

Línea	Costo 16-17	Costo 17-18	Costo 16-18
1	0,00	a3 = 107,70	a1 = 145,68
2	a2 = 37,98	0,00	0,00
3	0,00	a4 = 330,30	0,00

Tabla 18: Tiempo de viaje por arista.

La Tabla 18 indica, bajo el nombre de costo, el tiempo de viaje por tramo en segundos que le insume a cada línea.

Una vez que se cuenta con el costo de los tramos, y en base a la frecuencia de las líneas es posible obtener el tiempo de espera de los pasajeros en una parada como:

$$te(pda) = \alpha / \sum_{linea} f$$

Donde el tiempo de espera en una parada particular, es el resultado de la división de un factor *alfa = ½* (suponiendo un tiempo constante de arribo de *1/frec*. para cada línea sobre la parada) sobre la suma de frecuencias de las líneas que pasan por esa parada. A continuación en la Tabla 19 se describen los tiempos de espera de los pasajeros por parada.

Parada	Tiempo de Espera
16	69,73
17	122,73

Tabla 19: Tiempo de espera en parada.

Estos valores obtenidos en la Tabla 19, se utilizarán para calcular el *Z1* teórico del modelo de asignación del caso de estudio. Ver sección 4.7.2: *Plan de Pruebas-Comparativa Z1*.

Es importante destacar que este modelo utiliza las siguientes hipótesis para el cálculo de los tiempos de espera de pasajeros en parada, *Z1 y Z2*:

- El tiempo de subida y bajada de los pasajeros se ignora para el cálculo de Z1 (*tt=0*) pero para el cálculo de Z2 si es considerado (*tt>=0*).

- No se consideran los tiempos de las caminatas a las paradas origen y destino.

Como ya se especificó, los períodos de las líneas de buses del caso de estudio son 540s, 188s y 450s para la línea 1, 2 y 3 respectivamente, por lo tanto las frecuencias de las líneas de buses se describen en la Tabla 20.

Línea	Frecuencia = (1/período)
1	0,00185
2	0,00532
3	0,00222

Tabla 20: Frecuencias de líneas de buses.

En base a los resultados obtenidos en la Tabla 18, Tabla 19 y Tabla 20, se está en condiciones de calcular los valores *Z1* y *Z2* teóricos. Recordamos las fórmulas:

$$Z1 = \sum_{i,j} d(tv + te + tt)$$

$$Z2 = \sum_{rk} techo(fk * tk)$$

Los valores obtenidos se describen en la Tabla 21Tabla 20.

Indicador	Valor teórico
Z1	2,26
Tiempo de Espera (aplicando frecuencia)	0,9
Z2	3

Tabla 21: Valores teóricos de referencia.

Los valores expuestos en la Tabla 21 serán utilizados de aquí en más para validar el simulador y serán comparados con los resultados empíricos del mismo.

4.7.2. Plan de pruebas

La naturaleza estocástica de los datos que genera el modelo de simulación condiciona los tipos de test que se pueden realizar sobre el mismo. Las pruebas recomendadas en la bibliografía para la validación de este tipo de software son del tipo de análisis de factores [8].

Una vez finalizada la elaboración del caso de estudio *Aldea*, se establece un plan de pruebas con el fin de verificar y validar el modelo de simulación de este proyecto. La presente sección deja registrado qué es lo que se pretende verificar y validar, y también las pruebas a realizar las cuales son elegidas arbitrariamente por los encargados de este proyecto.

Objetivos

Los objetivos del presente plan de pruebas son:

- Verificación del nuevo módulo de estrategia de trasbordo para pasajeros.
- Verificación de las implementaciones de los nuevos indicadores Z1 y Z2.
- Validación del nuevo modelo de simulación.

Ámbito de la aplicación

Los datos de entrada se encuentran alojados en la carpeta *entrada* al mismo nivel de directorio que el ejecutable del simulador. La misma cuenta con los archivos que describen el caso de estudio de la *Aldea* descrito anteriormente.

Los datos de salida se recolectan en las carpetas *reportes* e *histogramas*, en el mismo nivel de directorio que el ejecutable del simulador. Cada ejecución del simulador genera un archivo de histograma y cuatro archivos de reporte. Todos los archivos generados tienen el sufijo "*_NN"* donde *NN* es el número de corrida.

La descripción (nombre y formato) de los archivos de entrada y salida se encuentra en la sección 4.6.

Pruebas

Prueba 1	
Descripción	Los pasajeros del sistema parten de una parada origen y llegan a una parada de destino. Probablemente algunos decidan hacer trasbordos y otros solamente viajar directo. La finalidad por lo tanto fue comprobar que los viajes efectivamente se realizaron.
Objetivo	Verificar que todos los pasajeros generados en el centroide 0 del caso de estudio se transportan al centroide 2 utilizando la nueva estrategia de trasbordos.
Metodología	Se realizan 100 *corridas independientes* [3] con distintas semillas y se contabilizan y promedian los pasajeros generados y los pasajeros arribados en cada corrida. La ejecución de cada corrida tiene una duración de 6 horas de tiempo de simulación.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt**, recoge totales de la simulación (en este caso interesa pasajeros transportados y generados) **.\reportes\Validacion_NN.txt**, monitoriza la actividad y el ciclo de vida de cada pasajero durante la simulación.
Criterios de falla	La prueba se considera no exitosa si se detectan pasajeros que esperan indefinidamente en las paradas 16 o 17 (paradas de salida e intermedia), pues la geometría y los recorridos garantizan que siempre existen conexiones entre la parada 16 y 18.
Criterios de aprobación	La media de pasajeros transportados es superior al 90% del total de pasajeros generados.

Prueba 2	
Descripción	La prueba 1 verifica que los pasajeros son transportados entre dos puntos. Esta prueba verifica que la cantidad de pasajeros que no son transportados se mantiene acotada a lo largo de todas las iteraciones.
Objetivo	Verificar que la cantidad de pasajeros que no arriban a destino en cada corrida se mantiene acotada.
Metodología	Se realizan 100 corridas independientes con distintas semillas y se contabilizan y promedian la diferencia de pasajeros generados y arribados. Se calcula un intervalo de confianza para dicho valor al 95%. La ejecución de cada corrida tiene una duración de 6 horas.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt**

[3] En general es aconsejable realizar varias corridas independientes para tomar varias muestras como respuestas tanto para calcular la media como la varianza (y la desviación estándar) [8].

Criterios de falla	La prueba se considera no exitosa si se detecta que al menos 10% de las iteraciones presentan más de 10 pasajeros que no arriban al centroide destino.
Criterios de aprobación	Que la media de la diferencia de pasajeros generados y transportados no supere la cantidad de 10 pasajeros.

Prueba 3	
Descripción	Este análisis tiene la finalidad de verificar la correctitud de cambio de estrategia de pasajeros del simulador.
Objetivo	Comprobar que la estrategia de viajes directos (original) no presenta viajes de trasbordo incluso cuando dispone de líneas para tal fin.
Metodología	1. Se configura el simulador para correr 100 corridas independientes con la estrategia original. 2. Se registran las líneas utilizadas por los pasajeros al viajar.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt**, recoge totales de la simulación (en este caso interesa pasajeros transportados y generados).
Criterios de falla	Al menos un pasajero viaja utilizando las líneas 2 y 3 que son de ida y vuelta y sirven para trasbordo.
Criterios de aprobación	Todos los pasajeros sin excepción utilizan la línea directa.

Prueba 4	
Descripción	Este análisis pretende determinar qué líneas son las preferidas por los pasajeros, al momento de variar la frecuencia de una de ellas respecto a las demás. Se estudia cómo incide la frecuencia de un bus para que los pasajeros decidan hacer el viaje directo o tomar el trasbordo. Debido a que la línea circular es la que posee viaje directo o de primer trasbordo, se elige la misma como candidata para variar su frecuencia.
Objetivo	Analizar la preferencia de los pasajeros por las líneas ida y vuelta si la frecuencia de la única línea circular que cubre todas las paradas se aumenta un 20%.
Metodología	1. Se realizan 100 corridas independientes con las frecuencias base de las líneas originales utilizando la estrategia de trasbordo. 2. Se realizan 100 corridas independientes mas utilizando trasbordos y se incrementa la frecuencia de la línea circular del caso base en un 20% 3. Se contabilizan los pasajeros transportados por cada línea.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**

Salida	**.\reportes\Validacion_NN.txt**
Criterios de falla	Se contabilizan menos viajes directos que en el caso base.
Criterios de aprobación	Se contabilizan más viajes por la línea circular y menos viajes por las líneas restantes respecto al caso base.

Prueba 5	
Descripción	Este análisis pretende determinar qué líneas son las preferidas por los pasajeros, al momento de cambiar la frecuencia de una de ellas respecto a las demás. Se estudia cómo incide la frecuencia de un bus para que los pasajeros decidan hacer el viaje directo o tomar el trasbordo. Debido a que la línea circular es la que posee viaje directo o de primer trasbordo, se elige la misma como candidata para variar su frecuencia.
Objetivo	Analizar la preferencia de los pasajeros por las líneas ida y vuelta si la frecuencia de la única línea circular que cubre todas las paradas se decrementa un 20%.
Metodología	1. Se realizan 100 corridas independientes con las frecuencias base de las líneas originales utilizando la estrategia de trasbordo. 2. Se realizan 100 corridas independientes mas utilizando trasbordos y se decrementa la frecuencia de la línea circular del caso base en un 20%. 3. Se contabilizan los pasajeros transportados por cada línea.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\Validacion_NN.txt**
Criterios de falla	Se contabilizan más viajes directos que en el caso base.
Criterios de aprobación	Se contabilizan menos viajes por la línea circular y más viajes por las líneas restantes respecto al caso base.

Prueba 6	
Descripción	La red de transporte aunque es sencilla está diseñada para favorecer la estrategia de trasbordos. Es de interés realizar dos series de ejecuciones con los mismos datos de entrada pero con distintas estrategias (una directa y otra con trasbordos), para estudiar el comportamiento de las mismas. Es de esperar que el indicador tiempo de espera promedio de un pasajero para la serie de trasbordos sean menor que el de la serie con estrategia directa. El comportamiento de Z1 y Z2 es incierto a priori, pero resulta interesante incluirlo en este test.
Objetivo	Verificar empíricamente que el modelo de red vial del caso de estudio *Aldea* tiene menores tiempos de espera promedio (utilizado siempre las mismas entradas) utilizando la estrategia de trasbordo en lugar de la estrategia directa.
Metodología	1. Se realizan 100 corridas independientes con estrategia de trasbordo y se registra tiempo de espera promedio de pasajeros, Z1

	y Z2. 2. Se realizan 100 corridas independientes con estrategia directa y se registra tiempo de espera promedio de pasajeros, Z1 y Z2.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt** (campo Z1 y Z2) **.\histogramas\EsperaPasajeros_NN.txt** (campo mean)
Criterios de falla	Contabilizar al menos un 10% de casos en los que no se cumpla que el tiempo de espera promedio con trasbordo es menor que con estrategia directa.
Criterios de aprobación	Que el tiempo de espera promedio de los pasajeros que utilizan trasbordo es menor que el tiempo de espera promedio de los pasajeros con estrategia directa.

Prueba 7	
Descripción	Hasta el momento todo los test realizados se han enfocado en la verificación de la validez de ciertos aspectos que se consideran indispensables por parte del simulador. La presente prueba se enfoca en la validación del simulador. Para ello se contrastan valores teóricos con valores empíricos.
Objetivo	Validar el simulador comparando el valor teórico calculado para Z1 contra los valores empíricos obtenidos de la simulación.
Metodología	1. Se realizan 100 corridas independientes con estrategia de trasbordo y se registran los valores de Z1 de cada corrida. 2. Se elabora un intervalo de confianza al 95% suponiendo distribución estándar para las 100 observaciones recolectadas. 3. Se compara el valor de Z1 teórico con el intervalo de confianza calculado en el paso anterior.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt**
Criterios de falla	Que el valor teórico de Z1 no pertenezca al intervalo de confianza y quede muy alejado del mismo. Para formalizar el término “muy alejado”, se considera que un valor de Z1 teórico que dista de la media del intervalo más de dos veces la distancia de la media al extremo del mismo.
Criterios de aprobación	Que el valor teórico de Z1 pertenezca al intervalo de confianza calculado con los valores empíricos.

Prueba 8	
Descripción	Este test es una copia del test anterior, la única diferencia es en el indicador que se registra a lo largo de las pruebas. En este caso el indicador a

	observar es Z2. Una vez más se procede a contrastar valores teóricos con valores empíricos.
Objetivo	Validar el simulador comparando el valor teórico calculado para Z2 contra los valores empíricos obtenidos de la simulación.
Metodología	1. Se realizan 100 corridas independientes con estrategia de trasbordo y se registran los valores de Z2 de cada corrida. 2. Como el resultado de Z2 siempre es un entero, se obtendrá un único valor promedio para las 100 observaciones recolectadas. 3. Se compara el valor de Z2 teórico con el valor medio obtenido calculado en el paso anterior.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt**
Criterios de falla	Que el valor teórico de Z2 sea menor en más de 2 unidades al valor empírico del simulador.
Criterios de aprobación	Que el valor teórico de Z2 sea menor hasta 2 unidades del valor empírico del simulador.

Prueba 9	
Descripción	Este test es similar al test anterior, la única diferencia es en el indicador que se registra a lo largo de las pruebas. En este caso el indicador a observar es tiempo de espera en parada (*te*). Una vez más se procede a contrastar valores teóricos con valores empíricos.
Objetivo	Validar el simulador comparando el valor teórico calculado para *te* contra los valores empíricos obtenidos de la simulación.
Metodología	1. Se realizan 100 corridas independientes con estrategia de trasbordo y se registran los valores de *te* de cada corrida. 2. Se elabora un intervalo de confianza al 95% suponiendo distribución estándar para las 100 observaciones recolectadas. 3. Se compara el valor de *te* teórico con el intervalo de confianza calculado en el paso anterior.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt**
Criterios de falla	Que el valor teórico de *te* no pertenezca al intervalo de confianza y quede muy alejado del mismo. Para formalizar el termino “muy alejado”, se considera que un valor de *te* teórico que dista de la media del intervalo mas de dos veces la distancia de la media al extremo del mismo.
Criterios de aprobación	Que el valor teórico de *te* pertenezca al intervalo de confianza calculado con los valores empíricos.

Prueba 10	
Descripción	El presente test y los restantes tienen la finalidad de validar el simulador haciendo uso de la técnica de análisis de factores o sensibilidad.
Objetivo	Validar el simulador verificando empíricamente que los indicadores *tiempo medio de espera de pasajero*, Z1 y Z2 varían al ser incrementada la frecuencia de buses del sistema.
Metodología	1. Se realizan 100 corridas independientes con estrategia de trasbordo (caso base). 2. Registrar para cada corrida, tiempos medios de espera de pasajeros, Z1 y Z2. 3. Incrementar un 20% las frecuencias originales de las líneas de buses del sistema (editar *lineas.txt*). 4. Se realizan otras 100 corridas independientes con estrategia de trasbordo. 5. Registrar para cada corrida, tiempos medios de espera de pasajeros, Z1 y Z2. 6. Comparar los indicadores registrados en 2 con los registrados en 5.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt** (campo Z1 y Z2) **.\histogramas\EsperaPasajeros_NN.txt** (campo mean)
Criterios de falla	Si para alguno de los siguientes casos se detecta que: 1) 15% de los tiempos de espera (con frecuencia incrementada) decrecen. Esto equivale a decir que se detecten 15% de iteraciones con tiempos medios de espera de pasajeros (para frecuencias incrementadas) las cuales sean superiores a las que tienen frecuencias originales. 2) Ídem para Z1. 3) Ídem para Z2.
Criterios de aprobación	Z1 y tiempo de espera de pasajeros son menores o iguales que el caso base para más del 85% de las iteraciones. Z2 es mayor o igual que el caso base para más del 85% de las iteraciones.

Prueba 11	
Descripción	Este test es una variante del test 9 y forma parte del conjunto de pruebas del análisis de factores. En este caso las frecuencias originales de líneas de buses se disminuyen un 20%.
Objetivo	Validar el simulador verificando empíricamente que los indicadores *tiempo medio de espera de pasajero*, Z1 y Z2 varían al ser decrementada la frecuencia de buses del sistema.
Metodología	1. Se realizan 100 corridas independientes con estrategia de trasbordo (caso base). 2. Registrar para cada corrida, tiempos medios de espera de pasajeros, Z1 y Z2. 3. Disminuir todas las frecuencias originales de líneas de buses un

	20% (editar *lineas.txt*). 4. Se realizan otras 100 corridas independientes con estrategia de trasbordo. 5. Registrar para cada corrida, tiempos medios de espera de pasajeros, Z1 y Z2. 6. Comparar los indicadores registrados en 2 con los registrados en 5.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt** **.\entrada\lineas.solucion** **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt** (campo Z1 y Z2) **.\histogramas\EsperaPasajeros_NN.txt** (campo mean)
Criterios de falla	Si para alguno de los siguientes casos se detecta que: 1) 15% de los casos de tiempos de espera (con frecuencia incrementada) aumentan. Esto equivale a decir que se detecten 15% de iteraciones con tiempos medios de espera de pasajeros para frecuencias disminuidas al 20% las cuales sean inferiores a las que tienen frecuencias originales. 2) Ídem para Z1. 3) Ídem para Z2.
Criterios de aprobación	Z1 y tiempo de espera de pasajeros son mayores o iguales que el caso base para más del 85% de las iteraciones. Z2 es menor o igual que el caso base para más del 85% de las iteraciones.

Prueba 12	
Descripción	Este test completa los test del análisis de factores. En este caso el factor a analizar es la demanda, interesa investigar el valor de los indicadores (tiempo de espera, Z1 y Z2) cuando el sistema es sobrecargado. Para ello se realiza un incremento de un 10% de la demanda y se conservan las frecuencias de líneas de buses originales. Al considerarse tiempos de subida/bajada de pasajeros a los buses es interesante observar como incide la capacidad máxima del bus en los tiempos de espera.
Objetivo	Observar el comportamiento del sistema aumentando la carga de trabajo. A priori se espera que al incrementar la demanda el valor de Z1 (que incluye la demanda) se vea incrementado.
Metodología	1. Se realizan 100 corridas independientes con estrategia de trasbordo (caso base). 2. Registrar para cada corrida, tiempos medios de espera de pasajeros, Z1 y Z2. 3. Aumentar la demanda 10% (editar *demanda.txt*). 4. Se realizan otras 100 corridas independientes con estrategia de trasbordo. 5. Registrar para cada corrida, tiempos medios de espera de pasajeros, Z1 y Z2. 6. Comparar los indicadores registrados en 2 con los registrados en 5.
Entrada	Archivos del caso de prueba Aldea: **.\entrada\demanda.txt** **.\entrada\grafo.txt**

	.\entrada\lineas.solucion **.\entrada\lineas.txt**
Salida	**.\reportes\ReporteFinal_NN.txt** (campo Z1 y Z2) **.\histogramas\EsperaPasajeros_NN.txt** (campo mean)
Criterios de falla	Si para alguno de los siguientes casos se detecta que: 1) 15% de los tiempos de espera (con frecuencia incrementada) decrecen. Esto equivale a decir que se detecten 15% de iteraciones con tiempos medios de espera de pasajeros (para frecuencias incrementadas) las cuales sean superiores a las que tienen frecuencias originales. 2) Ídem para Z1. 3) Ídem para Z2.
Criterios de aprobación	Z1 y tiempo de espera de pasajeros son mayores o iguales que el caso base para más del 85% de las iteraciones. Z2 es mayor o igual que el caso base para más del 85% de las iteraciones.

4.7.3. Ejecución del plan de pruebas

La ejecución de pruebas se puede dividir en dos partes, la primera secuencia de pruebas está compuesta de pruebas de verificación y la segunda parte permite efectuar la validación del modelo.

Todos los resultados de las pruebas realizadas con el modelo de simulación utilizaron la siguiente configuración de componentes físicos y programas:

Componente	Descripción
CPU	*Intel Core 2 Duo* , 2.53Ghz
Placa base	*Asus* E7200
Memoria RAM	4GB DDR2
Sistema Operativo	*Windows 7 Ultimate* 64bits
PublicTransport.exe	*Programa Simulador*
Entorno de desarrollo	*Visual Studio 2010*
Tiempo de ejecución de cada prueba.	*Aprox. 1 minuto*

Tabla 22: Equipo utilizado para la ejecución de pruebas

El análisis estadístico se basó en análisis de factores e intervalos de confianza normales sobre las medidas de interés para la mayoría de los casos. Salvo que se indique lo contrario se realizaron 100 corridas independientes para los estudios.

Los pasajeros se generaron en el simulador utilizando una distribución exponencial negativa cuyo parámetro era igual al inverso de la demanda en segundos, para lo cual se calculó un tiempo entre arribos de 216 segundos.

Prueba 1: transporte de pasajeros

Se añadió código al simulador para que contabilizara los pasajeros generados (desde el centroide 0) y los pasajeros arribados (hacia el centroide 2) y se generó una salida con estos datos. En la Tabla 23 se observa el estudio estadístico con los valores que se obtuvieron para cada variable.

Pasajeros Generados	Pasajeros Transportados
101,8	96,9

Tabla 23: Estudio cantidad de pasajeros

La media de pasajeros generados es de 101,8 mientras que la de pasajeros transportados es de 96,9 o sea, un 95% de los pasajeros se transportan y llegan a destino.

El 5% de los pasajeros que no llegan a destino son los que una vez finalizado el tiempo de simulación, aún viajaban entre paradas, o caminaban desde centroide a parada o viceversa.

De esta forma se satisface el criterio de aprobación de la prueba porque los pasajeros efectivamente viajan en el sistema con la nueva estrategia de trasbordo.

Prueba 2: cota de pasajeros no transportados

Se realiza el análisis del comportamiento de la diferencia de pasajeros a lo largo de la simulación. Es de interés analizar si la diferencia se mantiene acotada a lo largo de las iteraciones.

Los datos obtenidos tienen una media de 4,9 pasajeros en un intervalo normal de [4,5; 5,3]. La cantidad de pasajeros que no son trasportados en promedio satisface el criterio de aprobación.

Se detectó que en 2 iteraciones sobre 100, más de 10 pasajeros (12 en el peor caso) no llegaron a destino lo que representa un 2% de las iteraciones.

De esta forma se satisface el criterio de aprobación de la prueba porque se verifica que la diferencia de pasajeros generados y arribados se mantiene acotada a lo largo de las iteraciones.

Prueba 3: Análisis de uso de cada línea con estrategia directa

Se verificó que la lógica de la estrategia directa no fuera afectada por la incorporación de una nueva estrategia. La Figura 40 muestra el resultado de la experimentación.

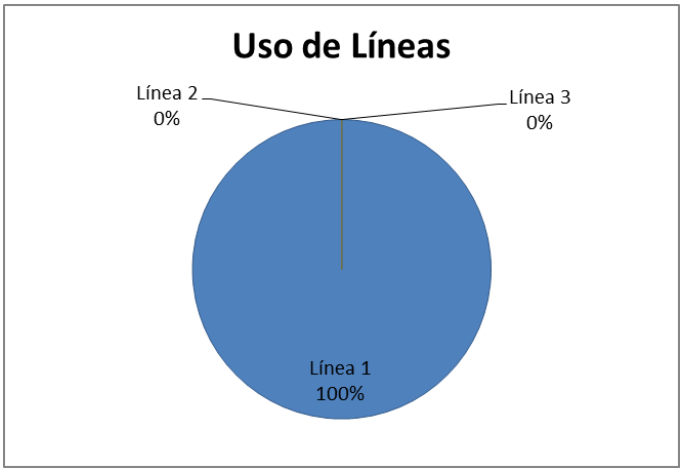

Figura 40: Uso de líneas, estrategia directa

Como era de esperar, al tener únicamente disponible la estrategia directa, la línea 1 (línea directa) es la preferida por la totalidad de los pasajeros del sistema. Por lo tanto, se satisface el criterio de aprobación de la prueba.

Prueba 4: Análisis de uso de cada línea con estrategia de trasbordo

Este es el caso por defecto con el cual se utilizó como caso base para las comparaciones. Las pruebas siguientes se verificaron contra este ejemplo.

Las frecuencias de las líneas son las que se definieron en la descripción del caso de estudio, la demanda de pasajeros es la que se especificó de 100 personas cada 6 horas. Como se observa en la Figura 41, el 70% de los pasajeros del sistema utiliza trasbordo frente al resto que decide ir directamente vía la línea 1 circular.

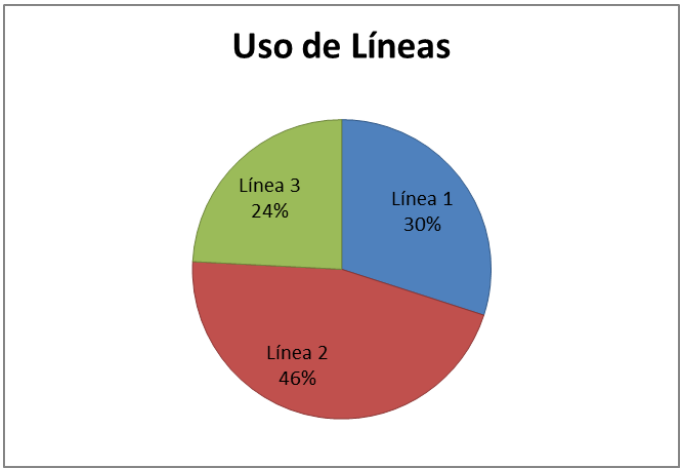

Figura 41: Caso base: porcentaje de uso de cada línea

Se incrementó un 20% la frecuencia de la línea 1, el resultado del análisis se observa en la Figura 42.

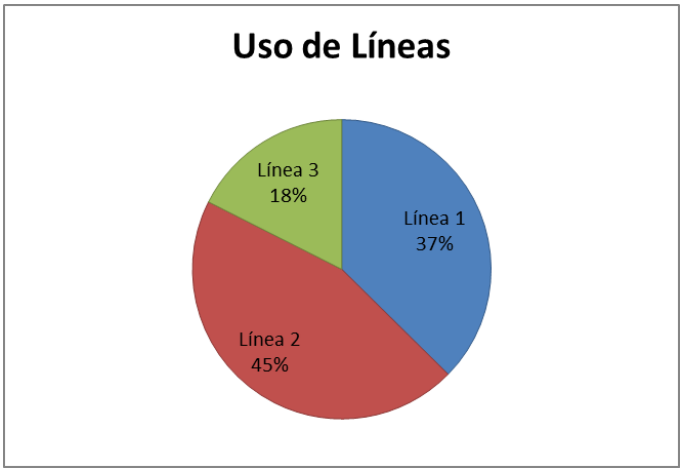

Figura 42: Frecuencia línea 1 incrementada 20%

El incremento en la frecuencia, implica que más pasajeros realicen viajes directos por la línea 1 y no por viajes de trasbordo a través de la línea 3, la cual desciende notoriamente su uso. Por lo tanto,

la pérdida de 6% de viajes por la línea 3 la absorbe la línea 1, lo cual es razonable ya que más pasajeros viajan de forma directa.

En total un 63% de los pasajeros prefieren realizar el trasbordo frente al 37% que desea viajar directamente. De esta forma se observa que un incremento en la línea directa es proporcional a la cantidad de viajes directos, como era esperable. De esta forma se satisface el criterio de aprobación de la prueba.

Prueba 5: Análisis de uso de cada línea con estrategia de trasbordo

En este caso nuevamente se presenta el análisis inicial contra el cual se realizará la comparación y se observa en la Figura 43.

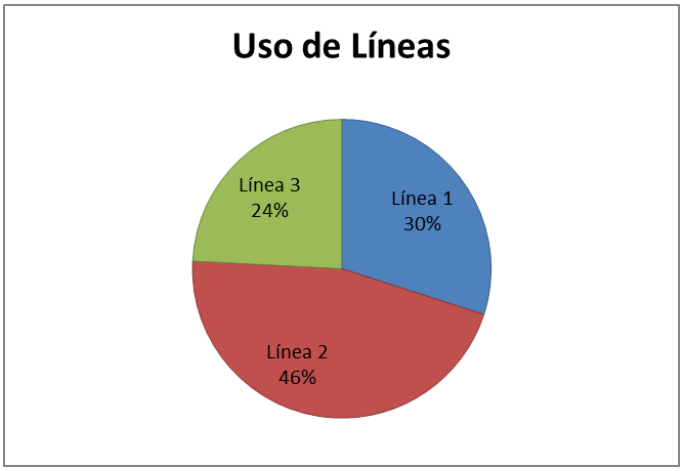

Figura 43: Caso base: porcentaje de uso de cada línea

Se decrementó un 20% la frecuencia de la línea 1, el resultado del análisis se observa en Figura 44.

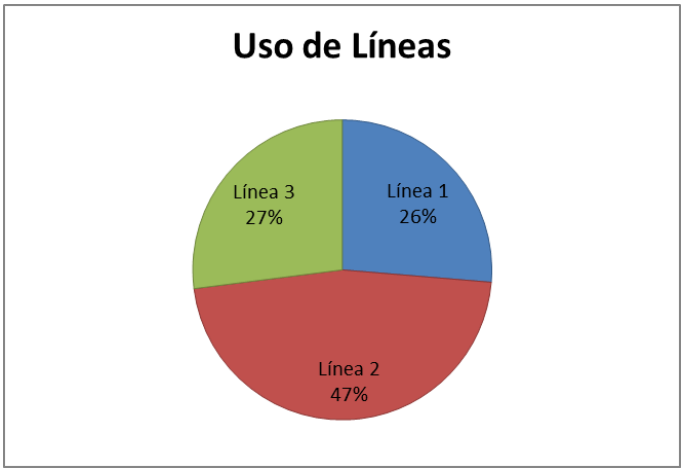

Figura 44: Frecuencia línea 1 reducida 20%

El decremento en la frecuencia de la línea 1, implica que menos pasajeros la prefieran para viaje directo (demora más) y más pasajeros realicen viajes de trasbordo. Por eso las líneas 2 y 3, ven incrementado su uso y por ello se deduce que la pérdida de pasajeros que sufre la línea 1, la absorben las líneas 2 y 3.

En total un 74% de los pasajeros prefieren realizar el trasbordo frente al 26% que desea viajar directamente. Al igual que el caso anterior las frecuencias de las líneas inciden directamente en el porcentaje de uso de las mismas por parte de los pasajeros, como era esperable. De esta forma se satisface el criterio de aprobación.

Prueba 6: Comparativa de estrategias

Se realizaron las pruebas con ambas estrategias y el resultado del análisis se observa en Tabla 24.

Indicador	Estrategia Directa	Estrategia Trasbordo
Z1	1,88	2,05
Tiempo de espera	1,20	0,93
Z2	4	4

Tabla 24: Comparativa de indicadores según la estrategia de pasajeros

De los resultados se obtienen las siguientes conclusiones:

- Tiempos de Espera: Si bien los pasajeros que realizan trasbordo tienen dos esperas en parada, una en la parada inicial y otra en la parada de trasbordo, en el análisis global la media del tiempo de espera de trasbordos es menor que en el caso sin trasbordo.

  Los pasajeros que utilizan solo la estrategia directa esperan únicamente por la línea 1, y la frecuencia de la misma incide directamente sobre sus tiempos de espera. La diferencia a favor de la estrategia de trasbordo está dada por la frecuencia de la línea 2 la cual es mayor (pasa más veces). Al tomar la línea 2 los pasajeros trasbordarán en la parada 17 y esperarán por la línea 3, la cual también tiene mayor frecuencia que la línea 1, por lo tanto en promedio, los pasajeros de trasbordo esperan menos globalmente. La Figura 45 muestra la comparativa de los tiempos de espera para ambas estrategias.

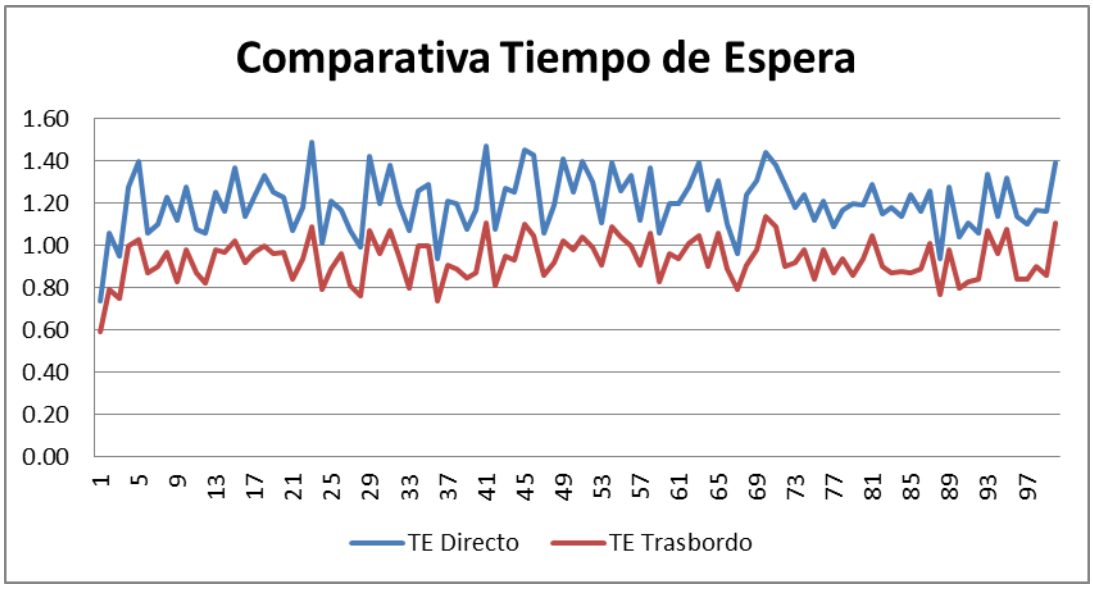

Figura 45: Tiempos de espera en parada (estrategia directa y trasbordo)

- Z1: El Z1 promedio de la estrategia directa es menor que el Z1 promedio de la estrategia de trasbordo. Si bien para los trasbordos hay menores tiempos de espera en paradas, el motivo por el cual Z1 es mayor que en el caso directo se debe a los tiempos de viaje a bordo. Los pasajeros que realizan trasbordo tienen dos viajes, uno hacia la parada de trasbordo y otro hacia la parada de destino. Los pasajeros que realizan un único viaje utilizan la línea 1, por lo tanto su tiempo de viaje depende del tiempo de recorrido de dicha línea. La Figura 46 muestra la comparativa de Z1 en ambas estrategias.

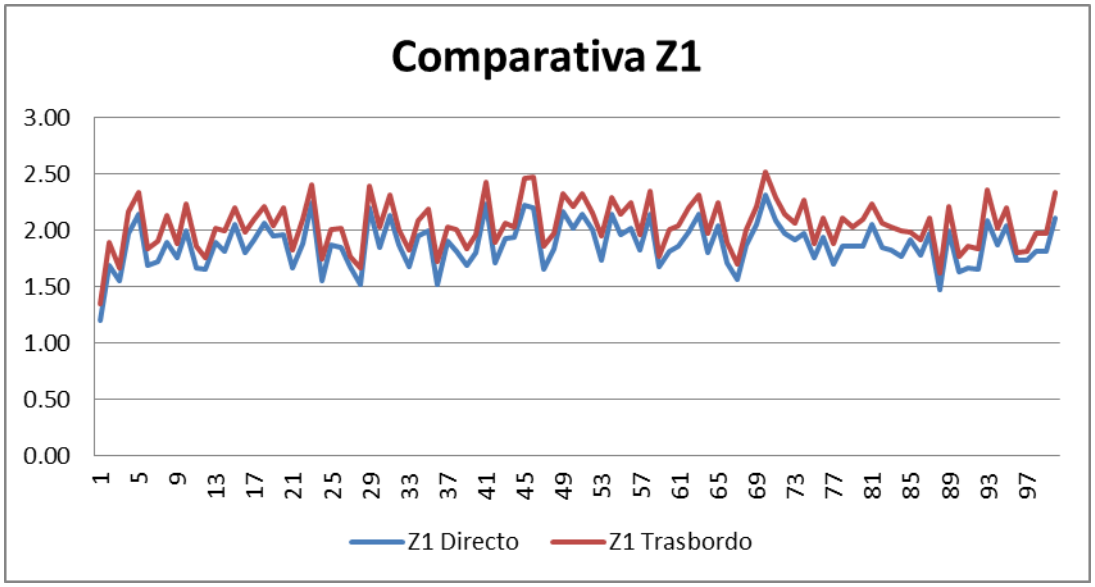

Figura 46: Tiempos de viaje de pasajeros Z1 (estrategia directa y trasbordo)

- Z2: Ambas estrategias aseguran un tamaño de flota de 4 buses. Debido a que la demanda de pasajeros es la misma para ambas pruebas y la frecuencia de los buses también; entonces el sistema puede mantenerse estable con la misma cantidad de buses para ambos casos. Otra interpretación es la siguiente: el sistema satisface la demanda de los pasajeros con la misma cantidad de buses para ambas estrategias. La Figura 47 muestra la comparativa de Z2 para ambas estrategias.

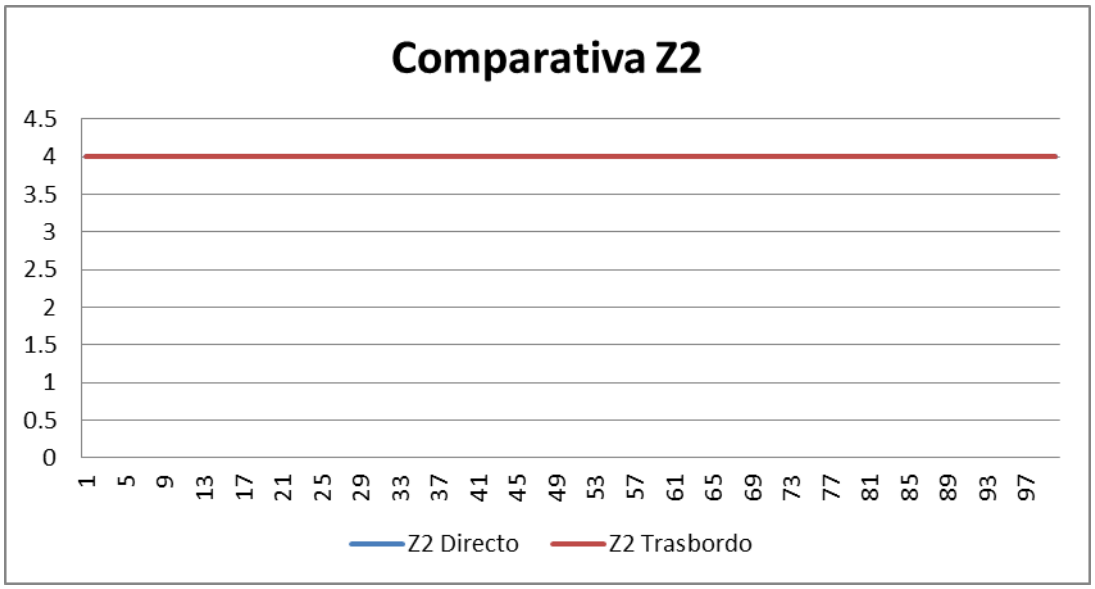

Figura 47: Tamaños de flota Z2 (estrategia directa y trasbordo)

Debido a que el diseño del caso de estudio (y de la solución en particular) favorece viajes por trasbordo, los pasajeros efectivamente esperan menos tiempo en parada y se satisface el criterio de aprobación de la prueba.

Prueba 7: Comparativa Z1 empírico y teórico

Se compararon los resultados de Z1 teórico calculado de la función objetivo del modelo de Baaj y Mahmassani versus Z1 obtenido del modelo de simulación para el caso de estudio. El Z1 teórico que se obtuvo para el caso de estudio se observa en la Tabla 25.

Indicador	Valor Teórico	Modelo de Simulación
Z1	2,26 (tabla 4)	2,05

Tabla 25: Indicador Z1 teórico y empírico

Con un 95% de probabilidad se determinó que la media pertenece al intervalo [2,01; 2,09].

Como se observa, el Z1 teórico no pertenece al intervalo de confianza del intervalo calculado, lo cual hace que el criterio de aprobación falle. De todas formas los valores medios son muy similares ya que varían ambos en los dígitos decimales. Los posibles motivos de esta mínima diferencia podrían deberse a:

- El cálculo del Z1 teórico se realiza mediante fórmulas obtenidas de los modelos de asignación: Baaj y Mahmassani y de Florian y Spiess pero de forma manual y no mediante los algoritmos propuestos por los modelos de optimización. Por lo tanto, es un valor teórico no óptimo sino aproximado.
- Para el cálculo teórico del tiempo de espera de cada parada, utilizando Florian y Spiess se utiliza un factor alfa que supone una aproximación de un tiempo constante de arribo de buses a las paradas. En nuestro caso utilizamos alfa = ½ (valor comúnmente usado según los autores). Mientras que en el simulador, el tiempo de arribo de cada bus al siguiente nodo vial está dado por una distribución normal.

Prueba 8: Comparativa Z2 empírico y teórico

Del mismo modo que para Z1, para el tamaño de flota se compararon los resultados de Z2 teórico versus Z2 del modelo de simulación y los resultados se observan en la Tabla 26.

Indicador	Valor Teórico	Modelo de Simulación
Z2	3 (tabla 4)	4

Tabla 26: Indicador Z2 teórico y empírico

Como se puede observar Z2 teórico es menor que el Z2 experimental con una diferencia de un bus. Por lo tanto, se intuye que a mismas frecuencias la causa de la diferencia está en los tiempos de recorrido.

El Z2 teórico se calcula con valores fijos de tiempos de recorrido por tramo provistos por el modelo de red. En el caso del simulador, el tiempo de recorrido es la sumatoria del tiempo de cada tramo. El tiempo de viaje por cada tramo es resultado de una distribución normal con media igual al tiempo del tramo y una desviación de 1/5 del tiempo del tramo que genera una demora. Es posible que para alguna línea en particular, sus tiempos de viaje por tramo sean mayores al tiempo medio y generen esa unidad extra.

Debido a que la diferencia entre el valor teórico de Z2 es menor en una unidad al valor empírico, el criterio de aprobación se satisface para esta prueba.

Prueba 9: Comparativa Tiempo espera en parada empírico y teórico

Para el tiempo de espera en parada *te*, se compararon los resultados del valor teórico versus el del modelo de simulación. Los resultados se observan en la Tabla 27.

Indicador	Valor Teórico	Modelo de Simulación
te	0,90 (tabla 4)	0,93

Tabla 27: Indicador te teórico y empírico.

Con un 95% de probabilidad se determinó que la media pertenece al intervalo [0,91; 0,95].

Al igual que en el caso del Z1, el criterio de aprobación falla porque el valor teórico no pertenece al intervalo de confianza calculado (aunque los valores medios son similares). Los posibles motivos de esta diferencia son los mismos expuestos para el caso del Z1.

Prueba 10: Análisis de Sensibilidad: Caso 1

Se incrementó un 20% la frecuencia de todas las líneas y el resultado del análisis se observa en la Tabla 28.

Indicador	Caso Original	Caso 1 (Frec. +20%)
Z1	2,05	1,82
Tiempo de espera	0,93	0,71
Z2	4	7

Tabla 28: Indicadores relevados para frecuencias originales y frecuencias +20%

De los resultados se obtienen las siguientes conclusiones:

- Tiempos de Espera: El incremento de un 20% en las frecuencias minimiza los tiempos de espera. Esto es razonable debido a que si los buses de las líneas realizan sus viajes a una mayor frecuencia, los pasajeros esperaran (en promedio) menos tiempo para tomar los buses. La Figura 48 muestra la comparativa de los tiempos de espera.

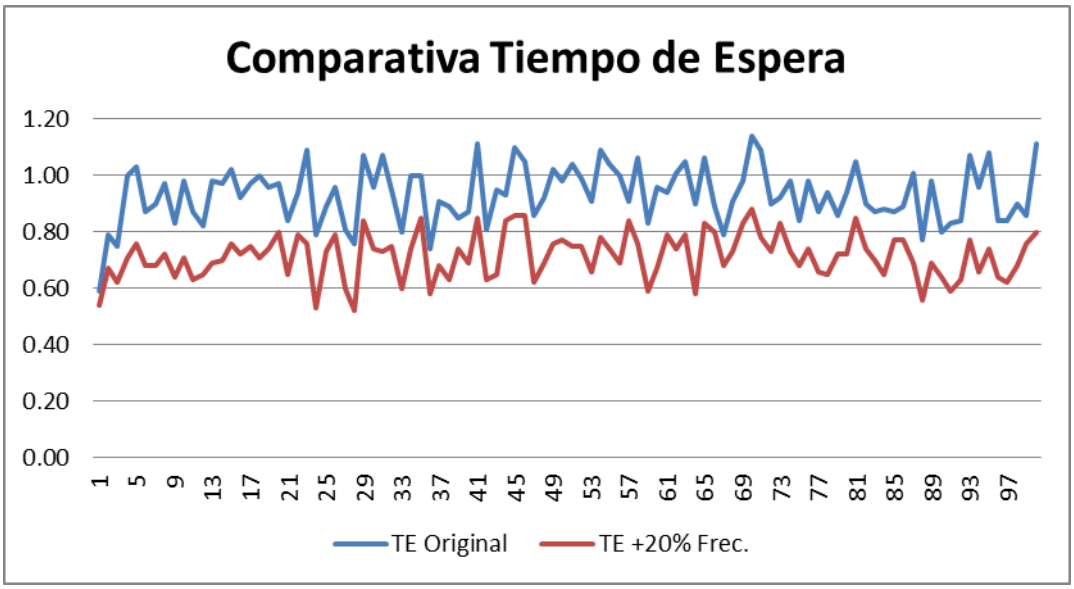

Figura 48: Tiempos de espera en parada para frecuencias originales y frecuencias +20%

El 100% de los tiempos de espera con la frecuencia incrementada disminuyeron respecto al valor del caso base, por lo tanto se satisface criterio de aprobación para tiempos de espera.

- Z1: Al igual que los tiempos de espera, el Z1 también disminuye al aumentar la frecuencia. Debido a que en promedio el tiempo total de los viajes se mantiene y la demanda no varía, entonces la reducción de Z1 está directamente relacionada con la reducción de los tiempos de espera. La Figura 49 muestra la comparativa para los Z1.

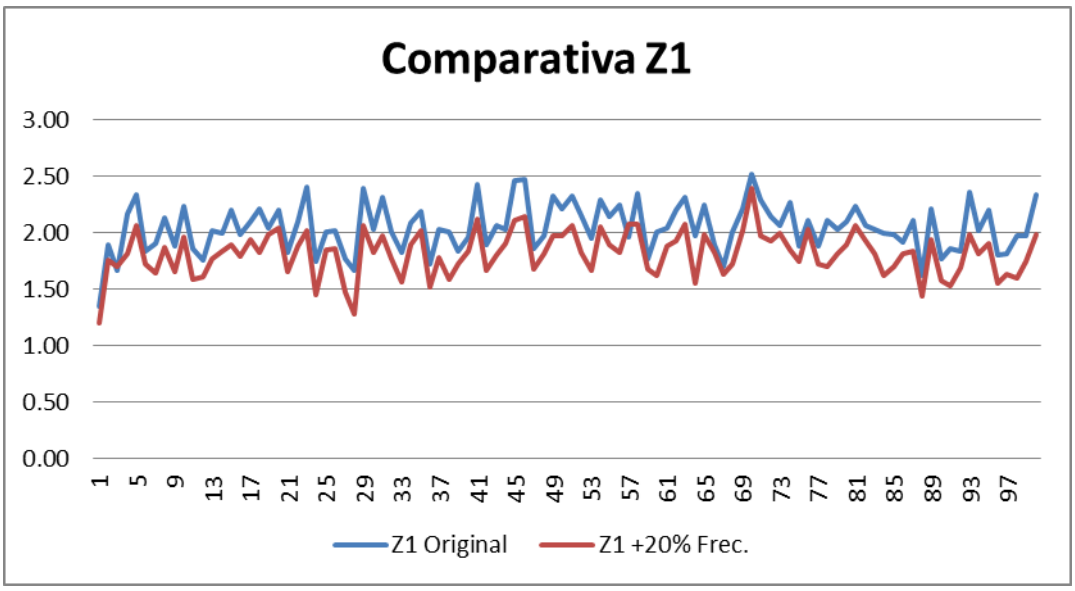

Figura 49: Valores de Z1 parada para frecuencias originales y frecuencias +20%

El 98% de los valores de Z1 con la frecuencia incrementada disminuyeron respecto al valor del caso base, por lo tanto se satisface criterio de aprobación para Z1.

- Z2: El Z2 sube al momento de incrementar la frecuencia debido a que se generan más buses en el sistema por unidad de tiempo, por lo tanto el tamaño de la flota crece. La Figura 50 muestra la comparativa para los Z2.

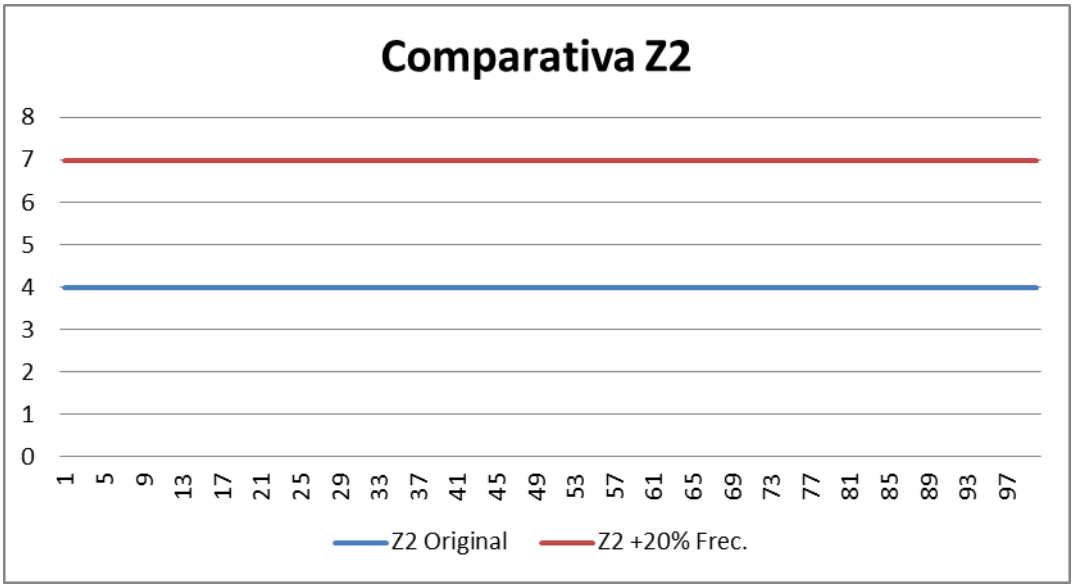

Figura 50: Valores de Z2 parada para frecuencias originales y frecuencias +20%

El 100% de los valores de Z2 con la frecuencia incrementada aumentaron respecto al valor del caso base, por lo tanto se satisface criterio de aprobación para Z2.

Debido a que los 3 indicadores satisfacen sus criterios de aprobación particulares, se satisface el criterio global de aprobación de la prueba.

Prueba 11: Análisis de Sensibilidad: Caso 2

Se decrementó un 20% la frecuencia de todas las líneas y el resultado del análisis se observa en la Tabla 29.

Indicadores	Caso Original	Caso 2 (Frec. -20%)
Z1	2,05	2,19
Tiempo de espera	0,93	1,12
Z2	4	3

Tabla 29: Indicadores relevados para frecuencias originales y frecuencias -20%

De los resultados se obtienen las siguientes conclusiones:

- Tiempos de Espera: Se observa que aumentan los tiempos de espera como resultado de decrementar la frecuencia. Esto es razonable debido a que si los buses de las líneas realizan sus viajes a una menor frecuencia, se generan menos buses por unidad de tiempo y los pasajeros esperaran (en promedio) más tiempo para abordarlos. La Figura 51 muestra la comparativa de los tiempos de espera.

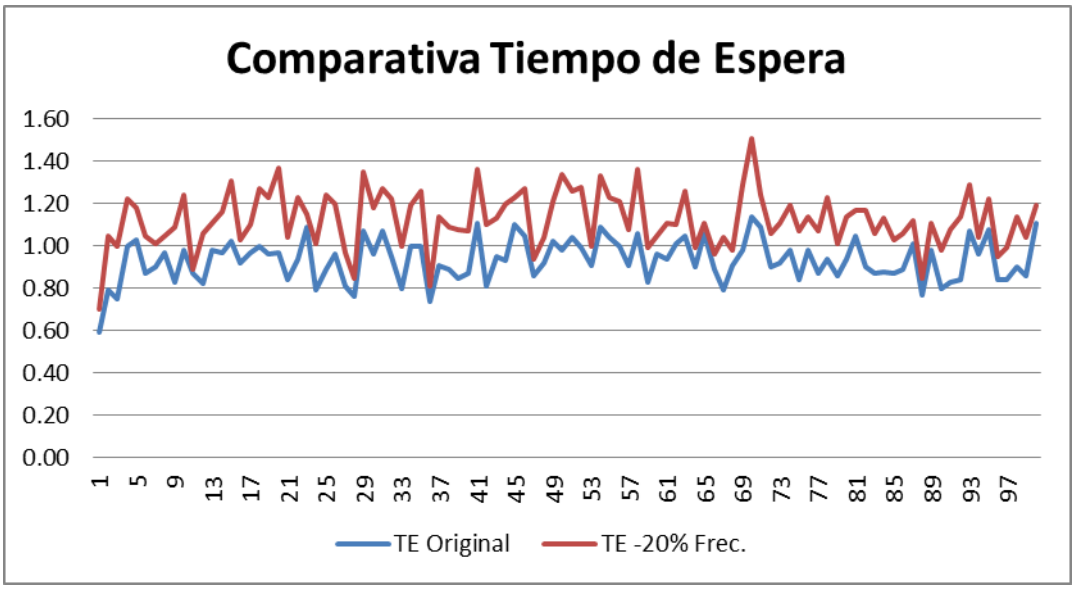

Figura 51: Tiempos de espera en parada para frecuencias originales y frecuencias -20%

El 100% de los tiempos de espera con la frecuencia reducida aumentaron respecto al valor del caso base, por lo tanto se satisface el criterio de aprobación para tiempos de espera.

- Z1: Al igual que los tiempos de espera, el Z1 aumenta al incrementar la frecuencia.
  Debido a que en promedio el tiempo total de los viajes se mantiene y la demanda no varía, entonces el incremento de Z1 está directamente relacionado con el incremento de los tiempos de espera. La Figura 52 muestra la comparativa de Z1.

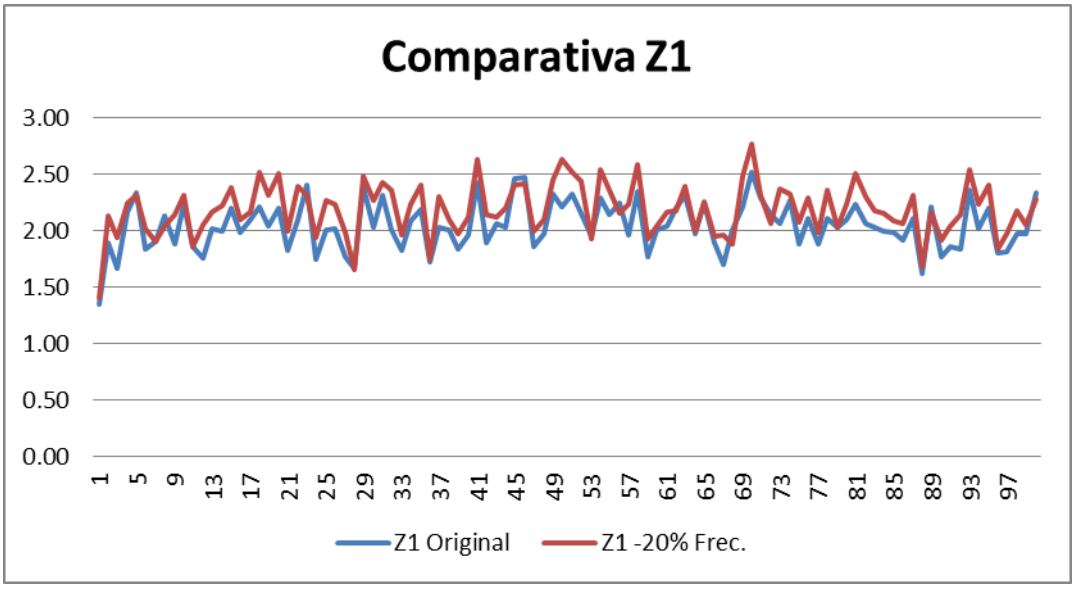

Figura 52: Valores de Z1 parada para frecuencias originales y frecuencias -20%

El 87% de los valores de Z1 con la frecuencia reducida aumentaron respecto al valor del caso base, por lo tanto se satisface criterio de aprobación para Z1.

- Z2: El Z2 se reduce al momento de decrementar la frecuencia debido a que, se generan menos buses en el sistema por unidad de tiempo, por lo tanto el tamaño de la flota decrece. La Figura 53 muestra la comparativa de Z2.

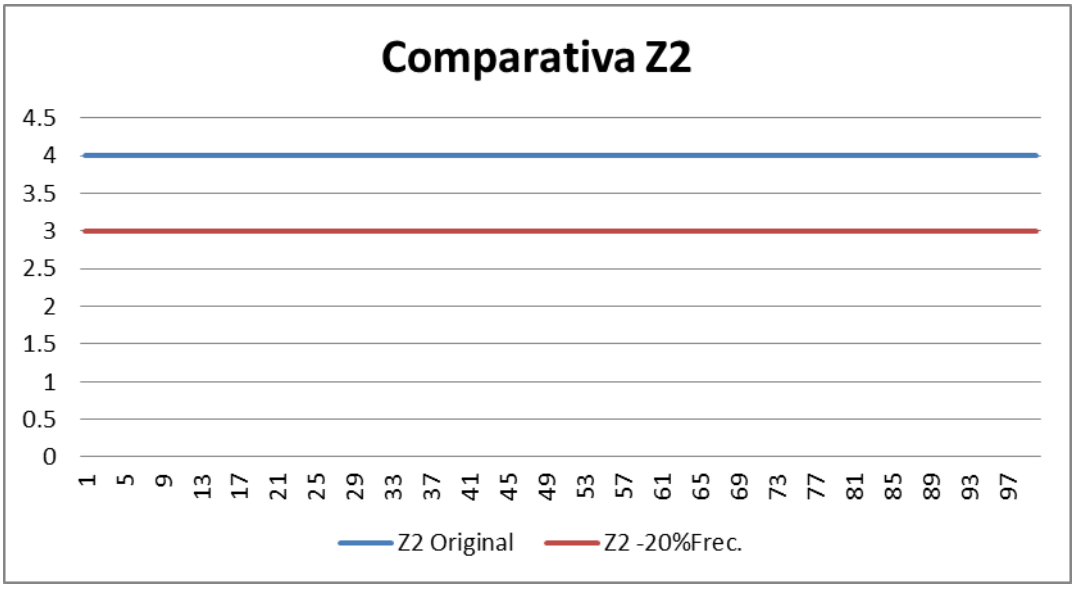

Figura 53: Valores de Z2 parada para frecuencias originales y frecuencias -20%

El 100% de los valores de Z2 con la frecuencia reducida disminuyeron respecto al valor del caso base, por lo tanto se satisface criterio de aprobación para Z2.

Debido a que los 3 indicadores satisfacen sus criterios de aprobación particulares, se satisface el criterio global de aprobación de la prueba.

Prueba 12: Análisis de Sensibilidad: Caso 3

Se incrementó de un 10% de la demanda de pasajeros y el resultado del análisis se observa en la Tabla 30.

Indicador	Caso Original	Caso 3 (Demanda +10%)
Z1	2,05	2,28
Tiempo de espera	0,93	1,04
Z2	4	4

Tabla 30: Indicadores relevados para demanda original y demanda +10%

De los resultados se obtienen las siguientes conclusiones:

- Tiempos de espera: el leve aumento de los tiempos de espera al incrementar la demanda, posiblemente se deba a que algunos pasajeros intenten abordar buses que tengan colmada su capacidad, lo cual los obligará a esperar el siguiente bus afectando negativamente sus tiempos de espera en parada. La Figura 54 muestra la comparativa de tiempos de espera.

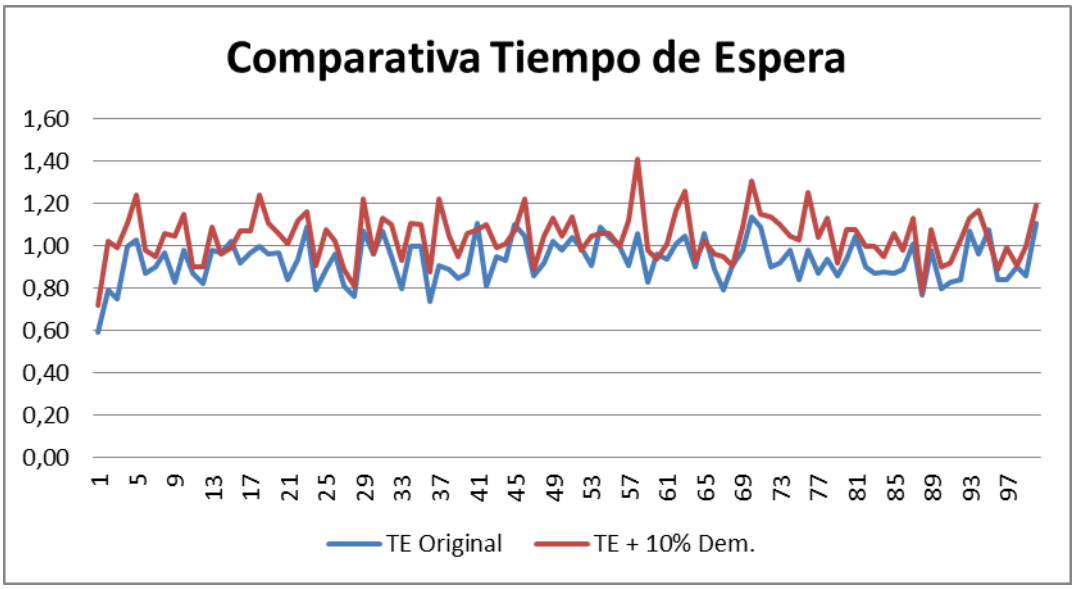

Figura 54: Tiempos de espera en parada para demanda original y demanda +10%

El 91% de los tiempos de espera con la frecuencia reducida aumentaron respecto al valor del caso base, por lo tanto se satisface criterio de aprobación para tiempos de espera.

- Z1: De manera simular al tiempo de espera, el Z1 también aumenta. El incremento en la demanda produce que existan más pasajeros esperando en parada y más cantidad de viajes. Por lo tanto, crecen en promedio los tiempos de espera y de viaje respectivamente, lo cual obliga a incrementar el Z1. La Figura 55 muestra la comparativa de Z1.

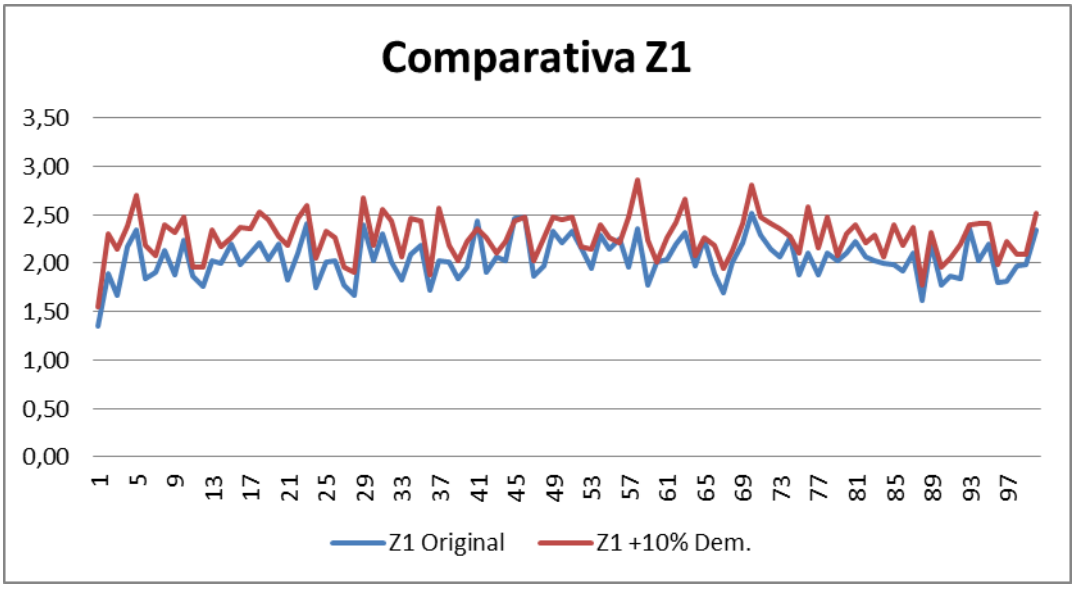

Figura 55: Valores de Z1 para demanda original y demanda +10%

El 97% de los valores de Z1 con la frecuencia reducida aumentaron respecto al valor del caso base, por lo tanto se satisface criterio de aprobación para Z1.

- Z2: El tamaño de flota permanece inalterado en ambos casos. Esto se interpreta como que no es necesario generar más buses por parte del sistema ya que el aumento de la demanda de pasajeros no es significativo. Además, los tiempos de subida y bajada de pasajeros no retrasan suficientemente los buses para que el sistema genere nuevos buses que satisfagan la demanda. Para que esto ocurriera la demanda debería ser mayor a un 10%. La Figura 56 muestra la comparativa de Z2.

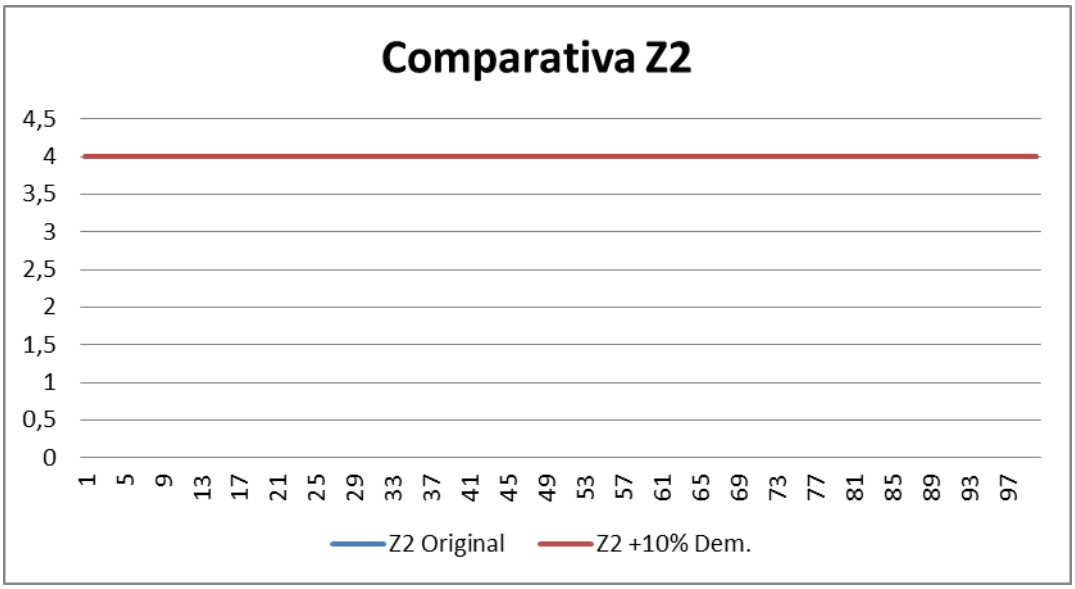

Figura 56: Valores de Z2 para demanda original y demanda +10%

El 100% de los valores de Z2 con la frecuencia reducida se mantuvieron iguales respecto al valor del caso base, por lo tanto se satisface criterio de aprobación para Z2.

El incremento de 10% de la demanda no sobrecarga al sistema y los indicadores no se "disparan" por lo tanto, el sistema es capaz de funcionar de manera similar aun con una demanda mayor

Debido a que los 3 indicadores satisfacen sus criterios de aprobación particulares, se satisface el criterio global de aprobación de la prueba.

Conclusiones luego de ejecutar el plan de pruebas

Respaldados por lo resultados obtenidos en el plan de pruebas, podemos afirmar que los indicadores (Z1, Z2 y tiempo espera en parada) con la nueva estrategia se comportan de forma esperada.

Que la nueva estrategia se comporta de forma esperada quiere decir:

- Que garantiza el transporte de pasajeros (no quedan pasajeros esperando indefinidamente en una parada). Visualmente se realizó el seguimiento de varios pasajeros, los cuales comenzaban el viaje de color rojo, luego hacían trasbordo y llegaban a la parada 17 (en color amarillo) y luego tomaban un bus para finalmente ir al centroide que se conecta con la parada final, numero 18. Esta verificación se corroboró además analizando los datos recolectados en los reportes descritos en el Plan de Pruebas.

- Que el sistema es sensible a la variación de frecuencia de buses y demanda como se observa en las pruebas de análisis de sensibilidad.
- La nueva estrategia presenta ventajas a la hora de reducir el tiempo de espera en parada de los pasajeros, pero en contra partida se obtienen tiempos de viaje mayores, afectando negativamente al Z1 (frente al Z1 de la estrategia de viaje directo). En cuanto al Z2 ninguna estrategia parece ser mejor que otra.

Se concluye que ninguna estrategia del modelo de simulación expuesto es mejor que otra, ambas sirven para evaluar soluciones del TNDP de la Aldea. Pero la elección de la estrategia a utilizar está sujeta a la habilidad o experiencia de los planificadores y al contexto particular de aplicación, ya que debe seleccionarse aquella que mejor se ajuste al comportamiento de los usuarios del sistema real que pretende modelar.

En base al conjunto de resultados obtenidos sobre las pruebas realizadas, el simulador (junto a su nueva estrategia) fue validado por el usuario.

4.7.4. Caso de estudio de Rivera

El sistema de transporte público utilizando datos de la ciudad de Rivera es de especial interés en este trabajo. Por lo tanto, se generó una solución del TNDP y la matriz demanda de pasajeros para el caso de estudio de la ciudad. Se obtuvo además la configuración de las 13 líneas que utilizan esos recorridos, con la particularidad de que 11 de ellas son de ida-vuelta y las 2 restantes son circulares. Una completa descripción de las líneas y sus recorridos se puede observar en la especificación del caso de estudio de Rivera en [5].

De [5] se obtuvieron los resultados óptimos de la evaluación de las funciones objetivo del modelo de asignación de Baaj y Mahmassani para el caso de estudio de la ciudad de Rivera.

Se utilizó una ventana horaria para la simulación de 24hs y se efectuaron 50 iteraciones. Se determinaron medias estadísticas e intervalos de confianza para Z1, Z2 y tiempos de espera utilizando ambas estrategias. El tiempo en promedio de la ejecución de la simulación fue de aproximadamente 5 horas con la misma configuración descrita en 4.7.3: *Ejecución del plan de pruebas.* Los resultados se muestran en la Tabla 31.

Indicador	Modelo de Baaj y Mahmassani	Modelo de Simulación (Estrategia Directa)	Modelo de Simulación (Estrategia Trasbordo)
Z1	406	408,5	461,9
Tiempo de espera	174	173,3	202,5
Z2	33	36	36

Tabla 31: Resultados de las funciones objetivos de ambos modelos para Rivera

A partir de los resultados teóricos obtenidos en [5] (columna Modelo de Baaj y Mahmassani de la Tabla 31) y las salidas del simulador, se pueden hacer las siguientes observaciones:

- El Z1 teórico es menor que los Z1 obtenidos por el modelo de simulación en ambos casos. Este resultado es de esperar ya que los modelos de simulación utilizan estrategias que probablemente consideren menos alternativas que las consideradas por el modelo utilizado en [5].

El valor medio del Z1 del simulador con estrategia de trasbordo está "más alejado" del valor teórico debido a que este último fue obtenido mediante un modelo que no considera trasbordos. Recordemos que en el modelo de simulación cada viaje de trasbordo implica dos viajes y dos esperas incrementando el valor de Z1. Eso implica que mientras el Z1 medio del modelo de asignación acumula tiempos cero para trasbordos, el del simulador (con estrategia trasbordo) acumula tiempos mayores a 0 por cada viaje de trasbordo.

El valor del Z1 medio del simulador con estrategia directa es levemente superior al valor medio del Z1 teórico. Se utilizó un intervalo de confianza normal para la media del Z1 del simulador y se obtuvo que el intervalo de confianza al 95% es [405,8; 413,1]. Como se observa el valor medio de Z1 teórico cae dentro del intervalo de confianza.

- El valor medio del Z2 es idéntico en ambas estrategias para el modelo de simulación y difiere en apenas 3 unidades del teórico. Una causa de esta pequeña diferencia podría ser que en el modelo de Baaj y Mahmassani, las duraciones de los recorridos se calculan por el tiempo de los tramos que los componen y sus costos son constantes para cualquier recorrido. En el simulador se utiliza una distribución normal con media igual al costo del tramo y desviación igual a 1/5 del mismo. Por ello el costo de viaje por cada tramo es resultado de una distribución y no es constante.
- Respecto a los tiempos de espera, se observa una situación muy similar que con Z1. El valor medio del simulador con estrategia de trasbordo, es bastante "lejano" del tiempo obtenido con el modelo de simulación. Mientras que el valor medio del tiempo de espera del simulador con la estrategia directa es prácticamente el mismo valor teórico de referencia. Este resultado en particular era esperado ya que, si los Z1 de ambos modelos eran muy similares, lo mismo debería ocurrir con los tiempos de espera, y efectivamente sucede.

  Al igual que para Z1, el tiempo de espera teórico se encuentra dentro del intervalo de confianza [172,42; 176,05] elaborado con los tiempos de espera promedio arrojados por el simulador.

Conclusiones del caso de estudio de Rivera

De los resultados obtenidos se desprenden varias conclusiones:

1. En base a la evidencia empírica para la ciudad de Rivera, podemos afirmar que los promedios de los indicadores implementados en el modelo de simulación (en particular Z1 y tiempo de espera en parada) utilizando estrategia directa tienen un 95% de probabilidad de contener a los valores teóricos de referencia.
2. Que el modelo de simulación utilizado es una herramienta alternativa válida para evaluar soluciones del TNDP para un caso de estudio dado.
3. La estrategia directa es más adecuada para la solución proporcionada de la ciudad de Rivera ya que se obtienen mejores valores para las funciones objetivo que utilizando la estrategia de trasbordo.

Por lo tanto, podemos concluir que el modelo de simulación es una herramienta adecuada para evaluar distintas soluciones del TNDP para el caso de estudio de la ciudad de Rivera.

4.8. Conclusión

Se puede concluir que el nuevo modelo de simulación cumple los objetivos del proyecto. Por un lado se extendió el mismo con un nuevo módulo de comportamiento de pasajeros en forma exitosa y por otro lado se verificó utilizando casos de estudio elaborados (ver sección 4.7.2 *Plan de pruebas*) y un caso real (ver sección 4.7.4 *Caso de estudio de Rivera*) cuyos resultados fueron validados por el usuario.

Como conclusión final podemos afirmar que el modelo de simulación propuesto es una herramienta válida para evaluar soluciones del TNDP utilizando los indicadores Z1, Z2 y tiempo de espera en parada, la misma puede ser utilizada como herramienta de apoyo a la toma de decisiones en la planificación de transporte urbano colectivo.

5. Herramienta de visualización del Simulador

5.1. Introducción

Como se menciona en el capítulo 4, el simulador *PublicTransport* implementado y descrito en [3] incluye una salida visual utilizando la biblioteca multimedia SDL [13]. Esta salida visual se probó con éxito en el caso de la ciudad de Rivera, sin embargo presenta algunas limitaciones funcionales.

Dichas limitaciones funcionales motivan al usuario a solicitar la búsqueda de alternativas que permitan superar estas limitaciones en el marco de este nuevo proyecto. Las alternativas posibles son, añadir las funcionalidades que faltan a la salida visual ya existente o implementar una nueva salida visual que incorpore todas las funcionalidades de la salida visual original y que además incluya las funcionalidades faltantes.

Las limitaciones funcionales de la salida visual original del simulador *PublicTransport* son, la falta de zoom sobre el área que se despliegan los mapas y se visualiza la simulación, y la falta de respuesta inmediata por parte del simulador a las acciones del teclado que realiza el usuario.

Se plantea entonces como tercer componente de este proyecto de grado, modificar la interfaz del simulador existente o implementar una nueva salida visual independiente del simulador, de modo que se permita al usuario un manejo más amigable de la visualización (que reaccione inmediatamente a las acciones del usuario y que las acciones se puedan realizar mediante una interfaz gráfica de usuario).

Las inspecciones preliminares del código del simulador original nos revelan que la implementación del mismo es monolítica [3], existe acoplamiento entre el código del simulador (principalmente en la clase que encapsula el modelo) y la interfaz de usuario. El mismo dificulta el mantenimiento de la salida visual o la extensión de nuevas funcionalidades de interfaz de usuario (UI de aquí en más).

Es importante destacar que este acoplamiento es debido en parte a la herramienta EOSimulator, ya que para implementar la salida visual original es necesario incluir y programar con primitivas de la biblioteca SDL, siendo responsabilidad del programador la implementación de la entrada desde el teclado y el dibujo en pantalla de los modelos de red vial.

La implementación actual de la UI del simulador *PublicTransport* tiene un problema de desempeño. Cuando se trabaja con una simulación compleja o simplemente el número de entidades manejadas por el simulador aumenta al avanzar el tiempo, la velocidad de la animación disminuye y se pierde realismo de la animación. Esto sucede porque el simulador y la UI comparten el mismo proceso.

Otro detalle que revela la inspección del código de la salida visual del simulador *PublicTransport* es que no trabaja con coordenadas geográficas [14], cuando las entradas de la ciudad sobre la que se efectúa la simulación generalmente vienen dadas en coordenadas geográficas. Actualmente solo es capaz de dibujar en pantalla el caso de la ciudad de Rivera (págs. 54, 55 de [3]). Para trabajar con otra ciudad es necesario modificar el código del simulador para que este transforme las coordenadas geográficas de la nueva ciudad al área de dibujo del simulador.

Se decide entonces, implementar una aplicación de visualización (***Herramienta de Visualización*** de aquí en más) en un módulo completamente independiente del simulador y además desarrollar un mecanismo de comunicación entre el simulador y la nueva herramienta de visualización, de modo que el desempeño del simulador no afecte a la velocidad de la visualización. También se decide hacer un relevamiento de tecnologías de salida gráfica para seleccionar la más adecuada a las necesidades de este proyecto.

Un aspecto a tener en cuenta es que la nueva aplicación *herramienta de visualización* no es una aplicación aislada. Es necesario ubicarla en el contexto de una gran caso de uso que involucra a la aplicación *igoR-tp* para desarrollar un modelo de red vial, al simulador *PublicTransport* que realiza una o varias simulaciones sobre el modelo de red vial y las persiste en algún formato próximo a determinar y por último, a la herramienta de visualización que reconstruye la información persistida por el simulador y la presenta al usuario en forma visual. La Figura 57 indica el contexto de la herramienta de visualización en el marco del proyecto:

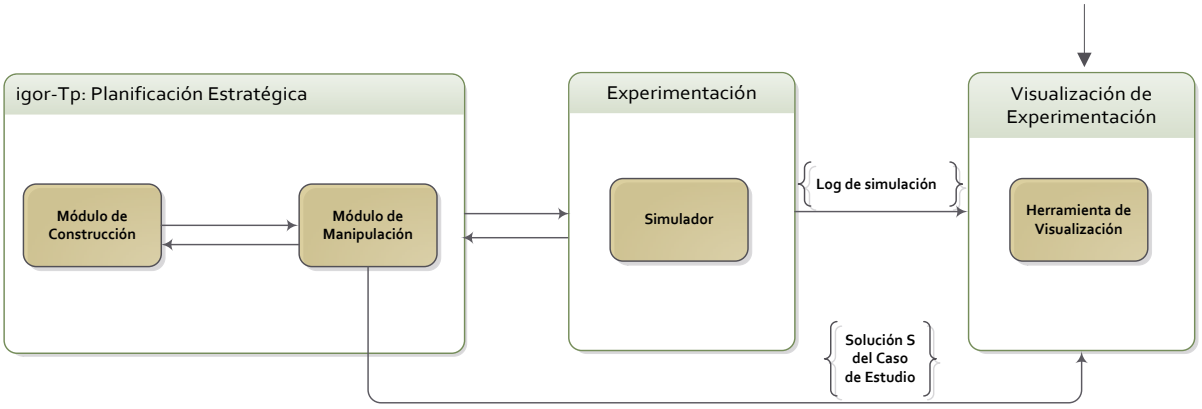

Figura 57: Contexto de la herramienta de visualización

Como se observa en la Figura 57, la herramienta de visualización tiene como entradas de datos la salida del simulador *PublicTransport* y una solución particular de recorridos construida en el módulo de manipulación de igoR-tp. Es importante destacar que la herramienta de visualización no verifica si la entrada desde *igoR-tp* está relacionada con la entrada desde el simulador, es responsabilidad del usuario proporcionar las entradas adecuadas a la herramienta.

Estas entradas de datos se abordarán en secciones siguientes y en particular existe un apéndice (*Apéndice C*) dedicado a la preparación de datos de entrada de la herramienta de visualización.

5.2. Requerimientos iniciales

Los requerimientos funcionales y no funcionales iniciales se obtienen del documento de propuesta de proyecto de grado y fueron refinados luego de celebrar reuniones quincenales con el tutor del proyecto, quien a su vez cumple el rol de usuario.

Los requerimientos relevados luego de la primera reunión con el usuario sumados a los del documento de propuesta de proyecto son los siguientes:

5.2.1. Requerimientos funcionales

La nueva salida visual del simulador debe:

- Contar con funcionalidades de zoom.
- Permitir visualización selectiva de los distintos componentes de un sistema de transporte público. Se entiende por visualización selectiva a la capacidad del software de mostrar u ocultar componentes visuales mediante acciones concretas en la interfaz de usuario.
- Ser capaz de leer y ejecutar logs generados por el simulador.
- Permitir la carga de archivos shapefiles extra para usar de fondo.

5.2.2. Requerimientos no funcionales

- El software de visualización debe trabajar con la tecnología más nueva posible para el despliegue de gráficos en pantalla.
- No hay restricciones con respecto a la tecnología con la que se implemente la nueva herramienta de visualización.
- Las tecnologías para implementar el nuevo visor deben ser de libre distribución y sin costo.

Estos requerimientos no son concretos, pero se consideran como punto de partida, se plantea la necesidad de realizar tareas de relevamiento de tecnologías de visualización para volver a evaluar con el usuario. Para lograr determinar correctamente los requerimientos se utiliza la técnica de ***Prototipado rápido*** (cap. 4.5 en [15]). Esta técnica sugiere realizar prototipos descartables con las distintas tecnologías de visualización relevadas, luego con dichos prototipos y asistencia del usuario se refinan los requerimientos y se descartan tecnologías.

Una vez seleccionado el prototipo que mejor se ajuste a las necesidades del usuario se continúa con un ciclo iterativo y evolutivo sobre el mismo, relevando nuevos requerimientos, validándolos por parte del usuario y ajustando los artefactos del análisis y diseño del mismo. Este ciclo se detiene cuando el usuario considera que el prototipo lo satisface o cuando el equipo de desarrollo detecta que el tiempo dedicado a esta tarea excede el tiempo de proyecto.

Las siguientes secciones del presente capítulo exponen cómo se realiza la selección de la tecnología de visualización, los requerimientos funcionales finales como casos de uso [16], los requerimientos no funcionales en forma descriptiva y algunos artefactos tradicionales del análisis y diseño orientado a objetos [17] para poder comprender fácilmente la aplicación final y su implementación.

5.3.Selección de tecnología

Alcanzado este punto se plantea: ¿Qué tecnología de salida gráfica seleccionar para la nueva *herramienta de visualización*: bibliotecas de dibujo como en el simulador original o bibliotecas de manipulación de mapas?

Los estudiantes involucrados en este proyecto cuentan con experiencia previa en el desarrollo con ambos grupos de bibliotecas, dicha experiencia se logró con los cursos *Introducción a la Computación Gráfica* [18] e *Introducción a los Sistemas de Información Geográfica* [19].

En base a la experiencia previa, se descartan las bibliotecas de dibujo *DirectX* [20], *OpenGL* [21] o *SDL* (biblioteca usada en el proyecto [1]) ya que no trabajan en forma nativa con coordenadas geográficas y no ofrecen funcionalidades para carga y manipulación de archivos de mapas.

Por lo tanto, se decide investigar las distintas alternativas de bibliotecas para manipulación de mapas para representar los modelos de prueba con coordenadas geográficas, dicho estudio en forma más profunda se puede consultar en el *Apéndice D.*

Las tecnologías relevadas en el *Apéndice D* para trabajar con mapas fueron:

- *MapWindows*
- *ESRI ArcGIS Silverlight*
- *DotSpatial*
- *BingMaps*
- *GoogleMaps*
- *OpenStreetMap*

Un aspecto a tener en cuenta es que no se conoce el sistema de coordenadas geográficas de los datos de prueba, en particular para el caso de la ciudad de Rivera. Esto implica que las coordenadas de los datos de prueba y la simulación no tienen por qué coincidir con las coordenadas de la ciudad de Rivera con las que trabaja *GoogleMaps*, *BingMaps* y *OpenStreetMaps* (de hecho no lo hacen). Otro inconveniente de estos controles es que, es necesario estar conectado a internet para acceder a los datos de sus mapas y a sus funcionalidades.

Se decide entonces relegar el trabajo con *GoogleMaps*, *BingMaps* y *OpenStreetMaps* para analizar *MapWindows*, *ArcGis API for Silverlight* y *DotSpatial.*

Considerando el requerimiento no funcional de trabajar con la tecnología más nueva posible, se descarta *MapWindows* por utilizar tecnología *ActiveX*. En su lugar se considera *DotSpatial* (es un proyecto desarrollado y mantenido por la misma comunidad de *MapWindows*) que ofrece las funcionalidades necesarias de carga de mapas y zoom y está implementado utilizando el framework *.Net 4.0* el cual es más reciente que *ActiveX*.

Finalmente se seleccionan dos tecnologías, estas son: *ESRI ArcGIS Silverlight* [22] y los componentes para *WinForms DotSpatial* [23]. Con cada una de ellas se implementan prototipos sin funcionalidades complejas, solamente lo necesario para cargar los archivos en formato Shapefile con el caso de estudio de Rivera y de esta forma elegir junto con el usuario la más adecuada para sus necesidades.

5.3.1. Prototipo 1 - ESRI ArcGIS Silverlight

Este primer prototipo (ver Figura 58) se desarrolló completamente en *Silverlight* [24], aprovechando el conocimiento disponible que se adquirió en el curso *Introducción a los Sistemas de Información Geográfica* [19]. El desarrollo con *Silverlight* tiene la ventaja de ser independiente de la plataforma [25], es decir las aplicaciones que hacen uso de esta tecnología puede ejecutar en distintos navegadores de distintos sistemas operativos, también es posible desarrollar aplicaciones sin necesidad de un navegador, que es este caso.

Con respecto al trabajo con mapas, esta biblioteca ofrece controles para desplegar mapas, además ofrece controles flotantes (con animaciones y transparencia para que no interfieren con el mapa) con funcionalidades de zoom y movimientos de mapa, también realiza suavizado de elementos geográficos. Tiene como inconveniente que para poder cargar datos geográficos y realizar operaciones sobre los mismos se debe contar con acceso a internet y además disponer de un servidor *ArcGIS* [26] que ofrezca los datos del caso de prueba en forma de servicios web.

Se buscó una manera alternativa, que no dependa de una conexión a internet y que no requiera un servidor *ArcGIS* (que es comercial) con los datos de Rivera, para que la aplicación permita la carga de los archivos Shapefile del caso de estudio de Rivera desde disco. La solución hallada es la biblioteca *CatFoods* para lectura de archivos Shapefile [27], con esta biblioteca se pueden cargar elementos geográficos como gráficos (puntos, polilíneas y polígonos) y ejecutar la aplicación Silverlight como si fuera de escritorio.

Este prototipo se descarta en común acuerdo entre el usuario y los desarrolladores, debido a que la solucion hallada con *CatFoods* tiene baja performance para dibujar los elementos geográficos y no permite animaciones fluidas. La causa de la baja performance es que los componentes de la biblioteca están desarrollados para mostrar mapas y datos enviados mediante servicios web desde un servidor *ArcGIS* y no para ser usados como una biblioteca de dibujo.

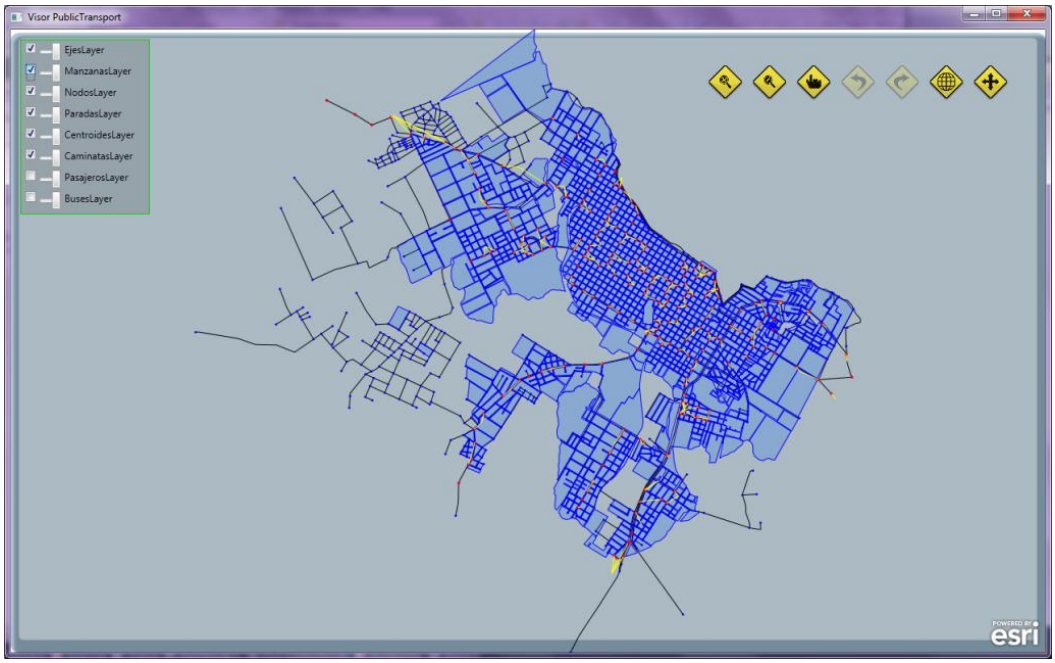

Figura 58: Prototipo Silverlight/ESRI

5.3.2. Prototipo 2 – DotSpatial/WinForms

Este prototipo alternativo no es tan moderno en lo que refiere a tecnologías de UI como en el prototipo anterior ya que la tecnología de presentación *WinForms* está un escalón por debajo del binomio *Windows Presentation Foundation/Silverlight* [28]. Para la elaboración de este prototipo se utiliza tecnología *.Net* [29], en particular el lenguaje C# [30] y la biblioteca *DotSpatial* [23].

La implementación se realiza desde cero con un mínimo esfuerzo, ya que los componentes *DotSpatial* permiten utilizar el diseñador de *Visual Studio* [31] y arrastrando solamente 3 componentes sobre un formulario vacío, se obtiene una pantalla completamente funcional como la de la Figura 59.

Estos componentes son libres bajo licencia GNU LGPL [32], ofrecen una barra superior para controlar los mapas, con zoom, herramienta de movimiento de mapa, historial de zooms, centrado automático y herramienta para medidas. Ofrecen un área (área izquierda de la Figura 59) donde es posible ver las capas cargadas en el mapa (área derecha de la Figura 59) y realizar operaciones sobre ellas.

Sobre este prototipo, además se implementó una animación de elementos gráficos de tipo punto sobre la capa de la red vial para simular el movimiento de Buses en el sistema. La animación resultó fluida, y los controles provistos por estos componentes satisfacen parcialmente los requerimientos funcionales (requerimientos 1,2 y 4) y no funcionales (requerimiento 1).

En este punto se cuenta con una solución parcial del problema, resta analizar, diseñar e implementar la lógica de este prototipo, es decir una aplicación capaz de leer y reconstruir la simulación dada por un log generado por el simulador *PublicTransport* y diseñar un mecanismo de comunicación con este prototipo (Figura 59).

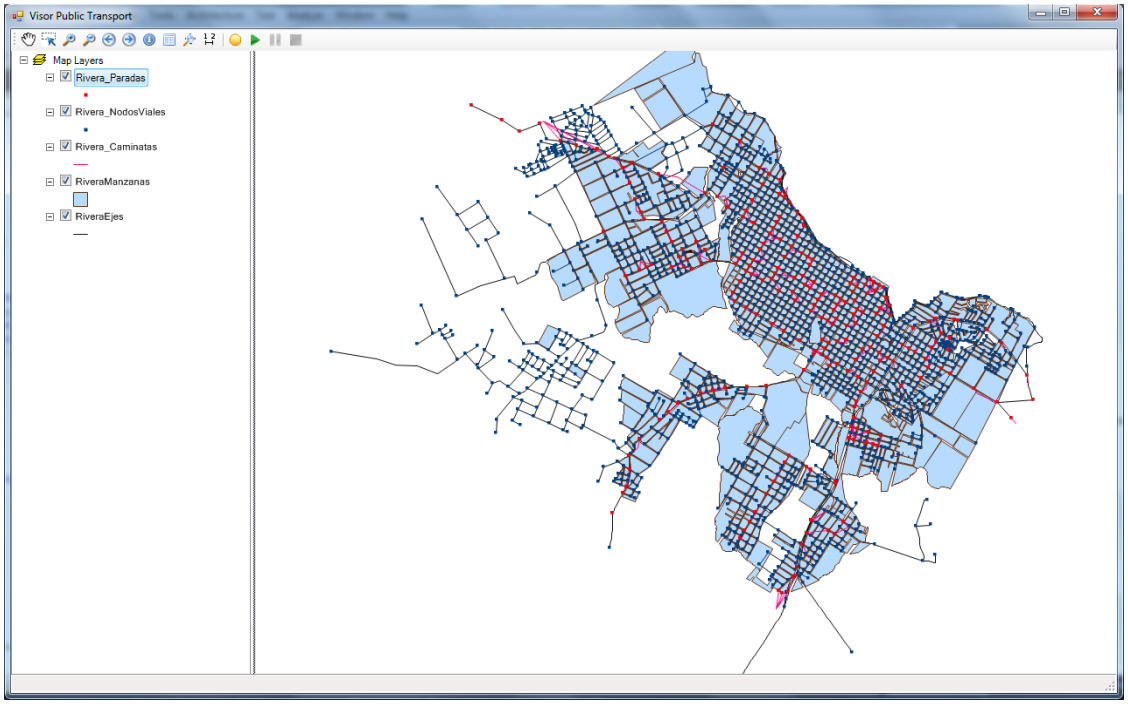

Figura 59: Prototipo WinForms/DotSpatial

5.4.Análisis

El presente capítulo sintetiza el resultado final del proceso de análisis. A continuación se exponen los requerimientos funcionales finales como casos de uso (diagramas UML y forma narrativa en lenguaje natural), los requerimientos no funcionales en forma narrativa y un modelo de dominio con su glosario para familiarizar al lector los conceptos más relevantes de este desarrollo.

Todos los requerimientos funcionales y no funcionales documentados en este capítulo fueron validados por el usuario.

5.4.1. Requerimientos funcionales

Los requerimientos funcionales documentados en este capítulo son el resultado final del proceso de desarrollo iterativo llevado a cabo con el usuario. Dichos requerimientos son un refinamiento de los requerimientos iniciales propuestos en la sección 5.2: *Requerimientos iniciales*.

Cabe señalar que estos requerimientos funcionales también están condicionados por la tecnología seleccionada (en este caso *DotSpatial*). A manera de ejemplo, la tecnología permite fácilmente añadir más información visual en forma de capas sobre un mapa, por lo que a pedido del usuario se añadió el caso de uso “Cargar capa de fondo”, por otro lado se perdieron funcionalidades que estaban presentes en la salida original *SDL* del simulador *PublicTransport*.

Las funcionalidades perdidas son “Ocultar línea” y “Mostrar línea” [3]. El motivo por el que se removieron dichas funcionalidades fue para ganar fluidez en la animación de la simulación reconstruida, puesto que usar una capa para todas la líneas da mejores resultados (la aplicación responde mas rápido y la animación es mas fluida) que usar una capa por cada línea.

A lo largo del proceso de desarrollo se detectó que las operaciones requeridas por parte de la nueva herramienta se pueden categorizar en dos grupos, por un lado operaciones sobre el modelo de red vial y por otro lado operaciones sobre la simulación.

5.4.1.1. Casos de uso: Diagramas

La Figura 60 muestra el diagrama de casos de uso con las operaciones relevadas sobre el mapa:

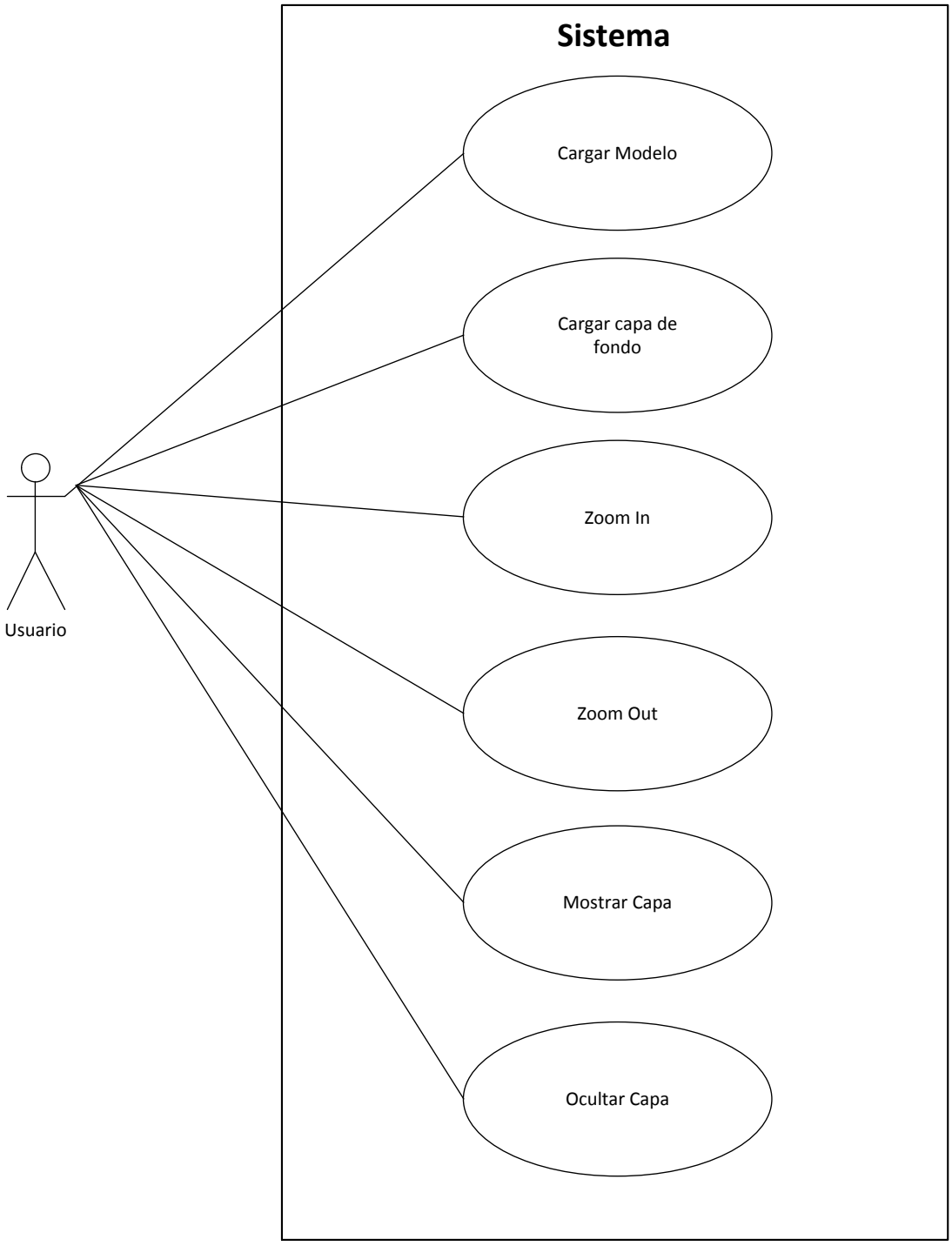

Figura 60: Requerimientos funcionales relativos al control de capas

La Figura 61 muestra el diagrama de casos de uso con las operaciones relevadas sobre la simulación:

Figura 61: Requerimientos funcionales relativos al control de simulación

5.4.1.2. Casos de uso: Especificación de alto nivel

Caso de uso	Cargar modelo
Actores	Usuario
Precondiciones	Ninguna
Sinopsis	El caso de uso comienza cuando el usuario hace clic en un botón específico de la UI. El sistema despliega un diálogo para seleccionar un archivo con extensión .manipulacion de igoR-tp. El usuario selecciona el archivo de manipulación. El sistema carga el modelo con sus capas estáticas en el mapa del visor y muestra cada una de las capas cargadas en la región de la UI llamada “Map Layer”. El sistema carga todos los logs de simulación que estén en la carpeta */Log* en el mismo nivel que el archivo de manipulación y carga los nombres de los archivos en un combo de la UI.

Caso de uso	Cargar capa de fondo
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente.
Sinopsis	El caso de uso comienza cuando el usuario hace clic en un botón específico de la UI del visor. El sistema despliega un diálogo para seleccionar un archivo con extensión shp. El usuario selecciona el archivo Shapefile. El sistema carga su contenido en una capa extra sobre todas las capas previamente cargadas.

Caso de uso	Zoom In
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente.
Sinopsis	El usuario presiona un botón específico de la UI del visor. El sistema cambia la imagen del puntero del mouse a una lupa con un símbolo de suma. El usuario selecciona una región rectangular dentro del mapa. El sistema ajusta proporcionalmente el contenido de la región rectangular al contenido del mapa.

Caso de uso	Zoom Out
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente.
Sinopsis	El usuario presiona un botón específico de la UI del visor. El sistema cambia la imagen del puntero del mouse a una lupa con un símbolo de resta. El usuario realiza un clic dentro de la región del mapa. El sistema ajusta proporcionalmente el contenido de la región rectangular a la doble del tamaño actual, centrando la nueva región en el punto de clic.

Caso de uso	Mostrar capa
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente, la capa a mostrar está oculta.
Sinopsis	El usuario ve las capas desplegadas en un árbol a la izquierda del mapa, donde cada capa tiene un componente de UI que permite marcar/demarcar la capa. El usuario marca la capa deseada haciendo clic en el componente de la UI. El mapa muestra el contenido de la capa.

Caso de uso	Ocultar capa
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente, la capa a ocultar esta visible.
Sinopsis	El usuario ve las capas desplegadas en un árbol a la izquierda del mapa, donde cada capa tiene un componente de UI que permite marcar/desmarcar la capa. El usuario desmarca la capa deseada haciendo clic en el componente de la UI. El mapa oculta el contenido de la capa.

Caso de uso	Seleccionar log de simulación
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente, la simulación está detenida.
Sinopsis	El sistema despliega todos los logs asociados al modelo cargado en algún componente de UI de tipo lista o combo. El usuario selecciona algún elemento de la lista. El sistema utilizará ese log al inicio de la próxima simulación.

Caso de uso	Incrementar velocidad de simulación
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente.
Sinopsis	El usuario realiza clic en un botón de la UI. El sistema despliega una ventana de diálogo con un control de selección de datos acotados con todos los valores posibles de velocidad de ejecución (un slider). El usuario selecciona un nuevo valor de velocidad al mover el control de UI a la derecha donde los valores son mayores. La velocidad queda almacenada y automáticamente se aplica a la simulación en curso o al próximo inicio de simulación.

Caso de uso	Decrementa velocidad de simulación
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente.
Sinopsis	El usuario realiza clic en un botón de la UI. El sistema despliega una ventana de diálogo con un control de selección de datos acotados con todos los valores de velocidad (un slider). El usuario selecciona un nuevo valor de velocidad al mover el control de UI a la izquierda donde los valores son menores. La velocidad queda almacenada y automáticamente se aplica a la simulación en curso o al próximo inicio de simulación.

Caso de uso	Precargar simulación
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente, la simulación está detenida.
Sinopsis	El usuario realiza clic en un botón de la UI. El sistema despliega una ventana de diálogo mostrando las opciones de carga de datos de simulación (como radiobuttons), uno con nombre Precarga y el otro AlVuelo. El usuario selecciona la opción de la UI con nombre Precarga haciendo clic en la misma. La próxima simulación que se lance, previamente leerá todo el log en memoria, antes de mostrar la animación de la simulación.

Caso de uso	Iniciar simulación
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente, la simulación está detenida.
Sinopsis	El usuario realiza clic en un botón de la UI. El sistema comienza a mostrar las animaciones correspondientes a la simulación en el mapa y desactiva el botón de la UI para iniciar la simulación y activa los botones de Pausa y Detener simulación.

Caso de uso	Pausar simulación
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente, la simulación está iniciada.
Sinopsis	El usuario realiza clic en un botón de la UI. El sistema detiene completamente la simulación sin borrar datos y el tiempo de simulación se detiene pero no se borra, el botón de Pausar y Detener simulación se desactivan y se activa el botón de Iniciar simulación.

Caso de uso	Detener simulación
Actores	Usuario
Precondiciones	Se ejecutó *Cargar modelo* previamente, la simulación está iniciada.
Sinopsis	El usuario realiza clic en un botón de la UI. El sistema detiene completamente la simulación borrando datos y pone a cero el tiempo de simulación, el botón de Iniciar simulación se activa y el de Detener y Pausar se desactivan.

5.4.2. Requerimientos no funcionales

Una vez seleccionado el prototipo sobre el que se realizó el proceso iterativo de desarrollo, quedaron determinados en forma precisa los requerimientos no funcionales de la herramienta de visualización:

- La entrada de datos es un archivo *XML* [33] generado por el simulador *PublicTransport*.
- El sistema se desarrollará sobre la plataforma *.Net 4.0* utilizando *Visual Studio 2010* y *C#*.
- La tecnología para visualizar mapas en formato *Shapefile* [34] es *DotSpatial* [23].
- El sistema dispondrá de un instalador.

5.4.3. Modelo de dominio

La Figura 62 especifica los conceptos relevados de la realidad del problema. Por un lado se tienen conceptos propios del simulador *PublicTransport*, y por otro, conceptos nuevos relevados de la realidad de este nuevo sistema (Mapa y Capa).

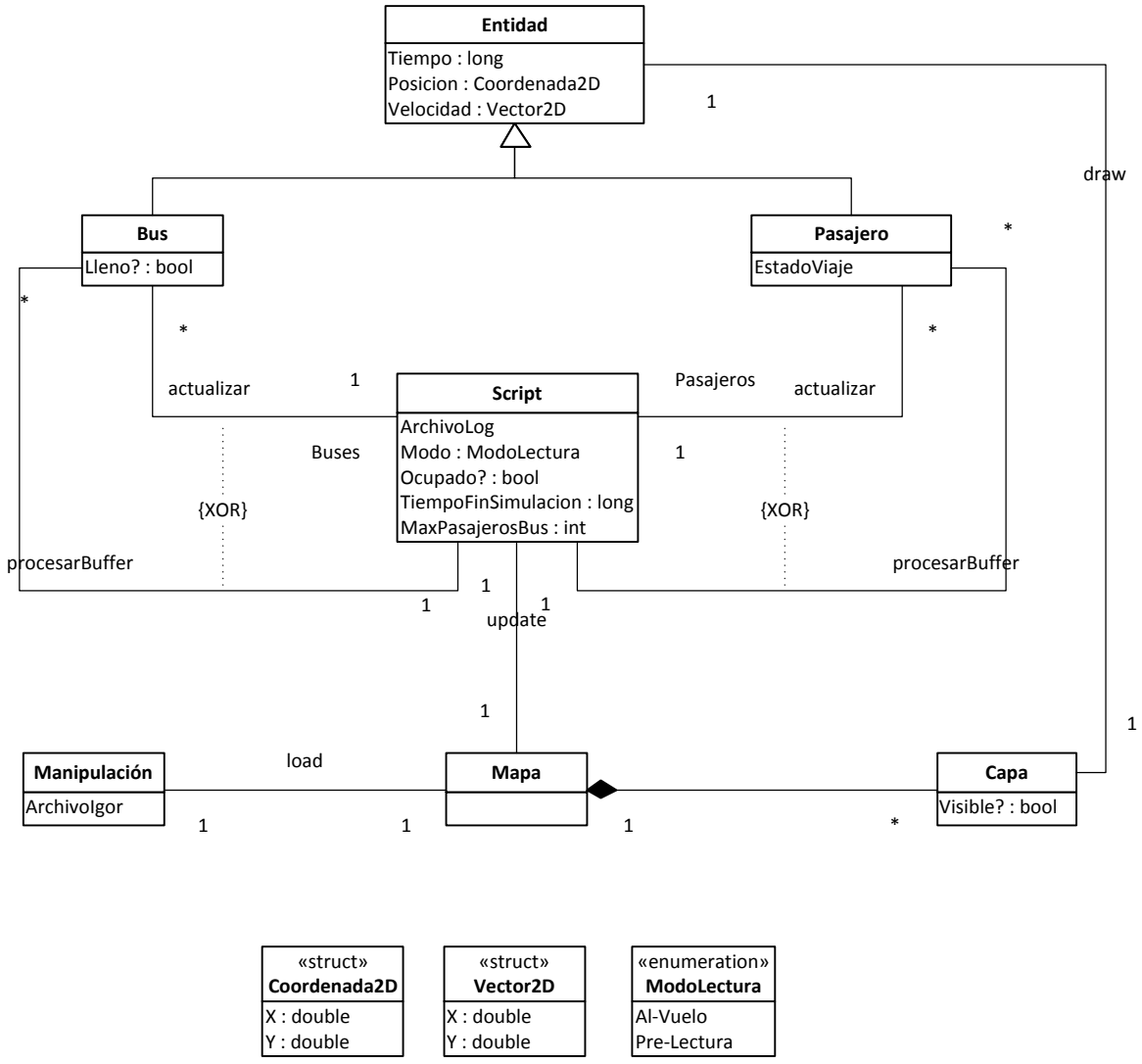

Figura 62: Modelo de Dominio

5.4.4. Glosario

Manipulación: Archivo de extensión *.manipulacion* generado por igoR-tp, el cual describe un modelo de red vial público dado por capas de elementos gráficos simples (puntos, polilíneas y polígonos), donde cada capa tiene un nombre arbitrario asociado.

Script: Archivo de log XML generado por el simulador *PublicTransport* el cual describe línea a línea el cambio de estado de los elementos que son móviles en una corrida del simulador.

Entidad: Elemento móvil de la salida visual del simulador *PublicTransport*. En determinado tiempo *t* se encuentra en una posición dada por el par *(x, y)* latitud y longitud.

Bus: Elemento móvil de la salida visual del simulador *PublicTransport*. Puede estar lleno o vacío.

Pasajero: Elemento móvil de la salida visual del simulador *PublicTransport*. Se encuentra visible cuando espera en una parada o cuando se mueve desde un punto origen del mapa a una parada o desde una parada a un punto destino en el mapa.

Mapa: Área de la pantalla donde se dibujan las capas de un modelo de Manipulación, los Buses y los Pasajeros.

Capa: División lógica del mapa que muestra elementos de la misma naturaleza. Puede ser visible o no. Todas las capas de un mismo mapa comparten el sistema de coordenadas (dadas por latitud, longitud) y el sistema de proyección.

5.5.Diseño

5.5.1. Arquitectura

La arquitectura de la aplicación se presenta en la Figura 63. La misma es una arquitectura en dos capas estricta (*presentación* consume exclusivamente servicios y datos de la capa *lógica* a través de una interfaz de comunicación bien definida pero no a la inversa).

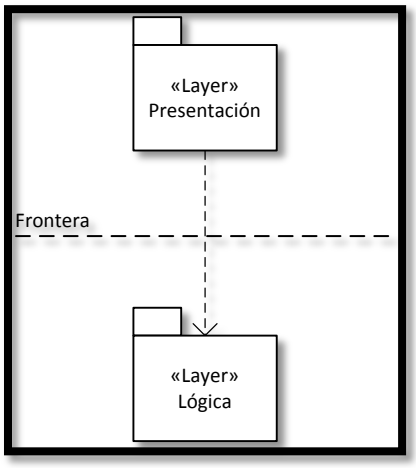

Figura 63: Arquitectura en capas del Visor

El motivo de la utilización de esta arquitectura y no una monolítica (ya que la aplicación es de pequeño porte), es que permite separar el desarrollo de la aplicación en dos grupos de trabajo y de esta forma asignar a cada capa desarrolladores con experiencia en cada área. Un equipo con especialistas en UI y *DotSpatial* asignados a la presentación y otro con especialistas en simulación a eventos discretos a la capa lógica.

El prototipo que se desarrolló y presentó en la sección 5.3.2 queda completamente inmerso en la capa de presentación (en esta altura del desarrollo forma parte del producto final) de forma que le permite a los encargados de la capa de presentación continuar mejorando la UI en las sucesivas iteraciones y el desempeño de las animaciones.

Por otro lado los encargados de la capa lógica pueden concentrar su esfuerzo en problemas tales como el diseño del formato del archivo de log (ver *Apéndice F*), la lectura del archivo de log y la reconstrucción de la simulación a partir del mismo.

La ventaja principal de esta arquitectura es que permite sustituir la capa de presentación con poco esfuerzo (pág. 237 de [15]) y reutilizar la capa lógica, el simulador *PublicTransport* y el formato de archivo de log sin necesidad de ser modificados.

Para paliar el problema de performance que se da en sistemas muy estratificados [15], se minimizó la estratificación de la arquitectura a dos capas, y en particular el acceso a datos y la lógica de negocios se ubicó en la capa lógica. El desempeño de las animaciones para el caso de la ciudad de Rivera es aprobado por el usuario.

Si profundizamos cada nivel de la arquitectura, se encuentra que está compuesta de dos paquetes de clases bien diferenciados alojados en capas distintas (Figura 64).

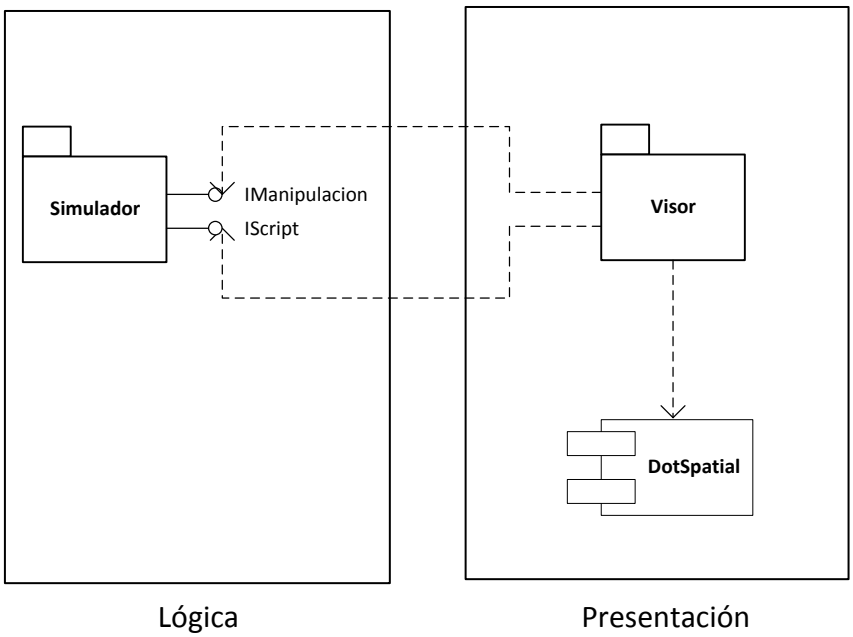

Figura 64: Agregación superior de las clases del Visor

El paquete Visor, agrupa todas las clases del prototipo que en su mayoría son de presentación, dichas clases utilizan los componentes *DotSpatial* para el despliegue de mapas y servicios y datos del paquete *Simulador*.

Es importante destacar que los conceptos *Capa* y *Mapa* relevados en el análisis, quedan implícitos en el componente *DotSpatial* y no necesitan ser implementados.

El paquete *Simulador* agrupa las clases responsables del parsing del log de simulación generado por el simulador *PublicTransport* y las clases entidades, que son las representaciones transitorias del estado de un objeto concreto de la simulación (Bus o Pasajero).

La lógica de negocios del paquete *Simulador* queda oculta, y solo es accesible mediante interfaces *ISimulacion* e *IManipulacion* utilizando los design patterns *Factory* y *FactoryMethod* [12].

5.5.2. Diagrama de clases

En este apartado se presenta el diagrama de clases de la capa lógica únicamente (la capa de presentación consta únicamente de dos clases que representan las vistas como formularios Windows y el componente de mapas). El diagrama de la Figura 65 muestra la relación de las principales clases de la capa lógica y cómo se expone el comportamiento al exterior.

Sobre la línea punteada rotulada como frontera, se muestran las interfaces y la Factory que son visibles desde la Presentación. Las demás clases quedan ocultas exceptuando *Bus* y *Pasajeros* que actúan como DTO (el pattern DTO se describe en [35]) entre la lógica y la presentación.

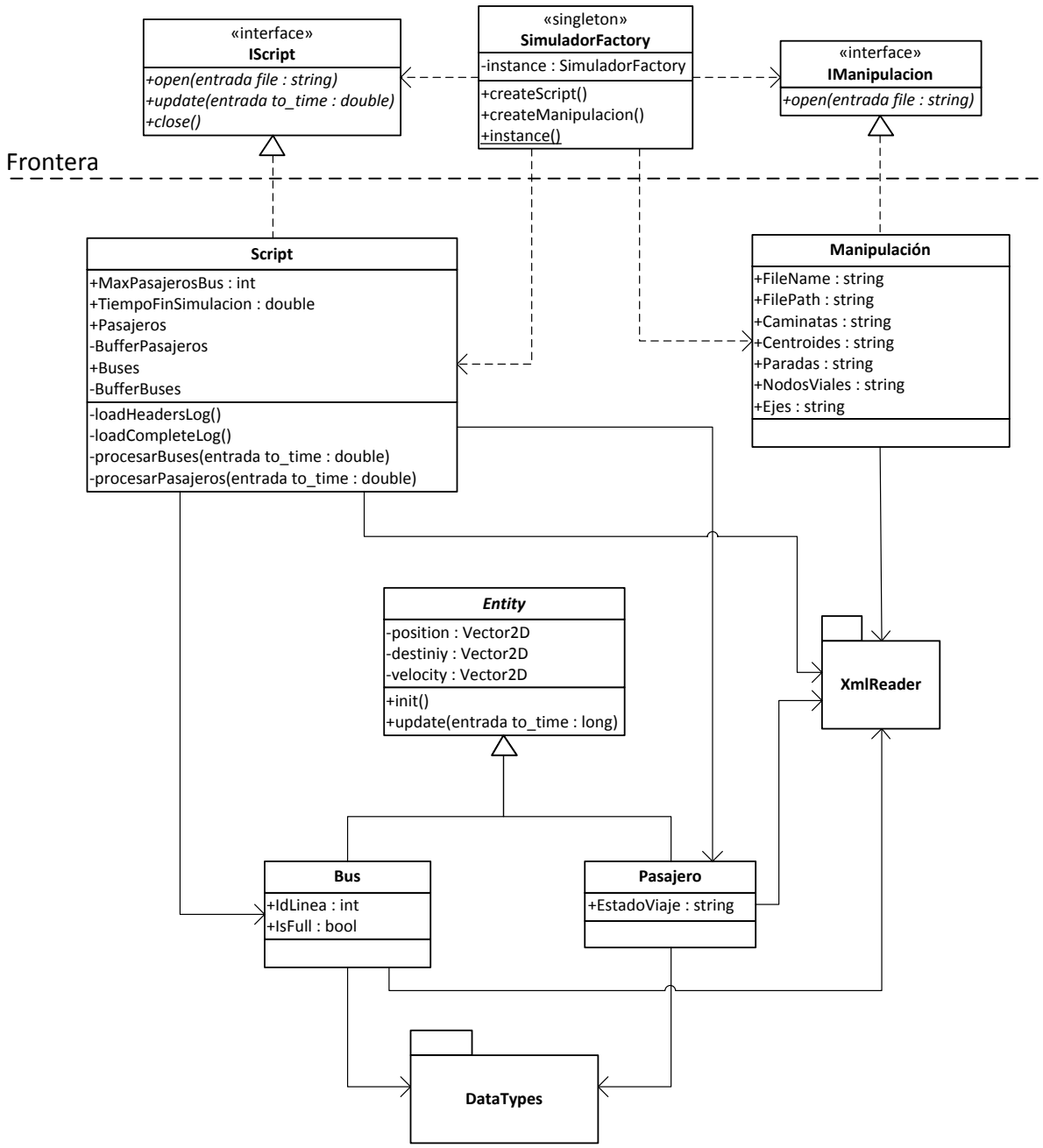

Figura 65: Diagrama de clases de la capa lógica (paquete Simulador)

La Figura 66 detalla el contenido del paquete *DataType*, dicho paquete agrupa como su nombre sugiere los datatypes de la capa lógica y los enumerados.

En particular la estructura *Vector2D* ofrece un conjunto de operaciones sobre vectores reales de dos dimensiones y operaciones de vectores por escalares. Esta estructura es muy útil para determinar posición, velocidad y siguiente posición en un tiempo *t* dado, la posición y la velocidad de un móvil en la simulación. Toda clase derivada de *Entity* hereda tres propiedades del tipo *Vector2D* que determinan la posición, la velocidad y el destino de la misma. La estructura Vector2D se implementó completamente desde cero, el pseudocódigo de las operaciones presentadas en la Figura 66 se puede consultar en la sección 5.6.5.

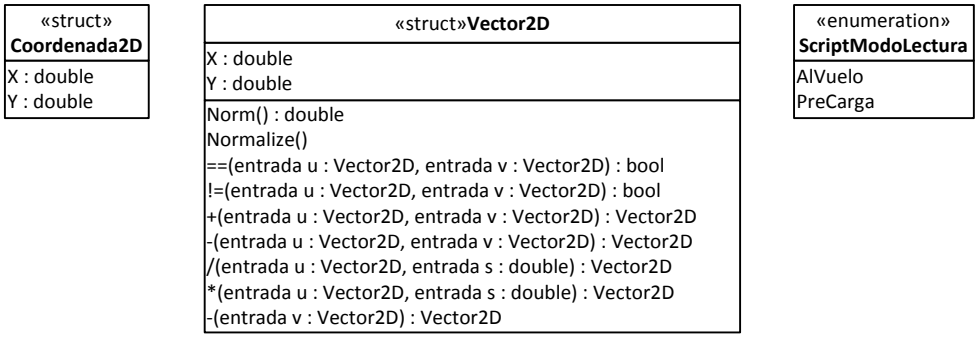

Figura 66: El paquete DataTypes

5.6.Implementación

Este capítulo explica principalmente en pseudocódigo haciendo uso de ejemplos la implementación de los elementos más relevantes de esta aplicación, en particular el funcionamiento de las animaciones y de cómo leer e interpretar el archivo de log para reconstruir una animación a partir del mismo.

5.6.1. Loop principal

El loop principal es un procedimiento que se invoca cada cierta cantidad de milisegundos y es el encargado de mantener actualizada la simulación y la representación visual de la misma.

Básicamente consiste en un procedimiento asociado a un temporizador, el cual realiza una comprobación del estado de la UI (modificada por la acción del usuario en la barra de herramientas de comandos: pausa, detener o ejecutar) y en caso de estar habilitada por el estado ***ejecutar*** realiza determinadas acciones en este orden:

1. borrar la región de mapa.
2. actualizar el estado interno de la simulación al tiempo actual de simulación.
3. redibujar la región de mapa con el nuevo estado.
4. incrementar el tiempo de simulación en cierto factor dado para la siguiente iteración.

Debido a que el temporizador proporcionado por el entorno de desarrollo *.Net* carece de la funcionalidad pausa, si la UI indica la voluntad del usuario de hacer ***pausa***, se debe simular este comportamiento ignorando las acciones anteriores al ejecutar `Loop` (se hace *busy waiting*). En caso que el usuario seleccione desde la UI ***detener*** simplemente se detiene el temporizador.

El siguiente pseudocódigo muestra como se implementa la rutina de `Loop`:

```
//Vista principal (FormMain.cs)
Long tics = 1                  //Tiempo de simulación a cero
Timer timer                    //Se instancia un nuevo temporizador
timer.Interval = DuracionTic   //Intervalo de tiempo entre tics
timer.tick = Loop              //Puntero a runtina de loop
timer.start()                  //Se inicia el temporizador
//Fin bloque

//Loop principal (FormMain.cs)
Proc Loop
      If (detener) Then //Si la UI indica stop => detener temporizador
            timer.stop()
      Else
            If (not pausar) Then     //Si la UI indica pausa => busywait
                  Clear()            //Borra el área de mapa.
                  Update(tics)       //Actualiza los objetos internos.
                  Draw()             //Dibuja los objetos nuevos en el mapa.
                  tics += FactorVelocidad //Se incrementa el tiempo
            EndIf
      EndIF
EndProc
```

5.6.2. Control del tiempo de simulación

La herramienta de visualización trabaja internamente con valores de tiempo de tipo entero, cada ejecución de un loop procesa el estado de un segundo particular de la simulación y actualiza la pantalla. Se denomina *tic* al momento del tiempo en el que el temporizador dispara la ejecución de un loop. El pseudocódigo del loop principal utiliza dos variables (*DuracionTic* y *FactorVelocidad*) que son modificadas desde la UI.

- ***DuracionTic*** se utiliza por el temporizador para determinar el intervalo de tiempo entre dos tics. Evidentemente si la ejecución del loop insume más tiempo que el valor dado por *DuracionTic* se tendrán problemas a la hora de dibujar el mapa, es por eso que la UI está acotada a valores enteros del rango 1-100 milisegundos y se inicializa con valor de 10 milisegundos.
- ***FactorVelocidad*** determina cuantos segundos de simulación se avanzan entre dos tics, de esta forma el usuario puede hacer evolucionar el sistema mas rápido dando la sensación que la simulación se acelera, cuando lo que en realidad sucede es que la herramienta siempre está dibujando una y otra vez el mapa cada *DuracionTic* milisegundos (a no ser que se cambie desde la UI), pero entre cada intervalo se dibuja el mapa con valores de tiempo de simulación más espaciados.

5.6.3. Lectura y proceso del log

La estructura y generación del log por parte del simulador *PublicTransport* se trata en el *Apéndice F*. El objetivo de este apartado es explicar cómo la herramienta de visualización recompone la simulación a partir del log, para tal fin se expone un ejemplo simple el cual ilustra cómo partiendo de un log la herramienta realiza el parsing, interpretación y ejecución del mismo.

Es conveniente recordar que el log generado por el simulador es un archivo *XML* donde cada línea representa el estado de una entidad particular (Bus o Pasajero) en un momento dado en la simulación. Dicha línea está compuesta por el tipo de entidad, un identificador de entidad, el tiempo de simulación en el que ocurre la acción de esa entidad, un código que indica la acción, y un par de coordenadas origen y destino. Dependiendo del código de acción la coordenada destino se considera o no. Si la coordenada destino se considera, es imperativo incluir otro atributo que es el tiempo que lleva desplazarse del origen al destino para calcular la velocidad en ese tramo.

Supóngase como ejemplo un Bus que recorre cuatro nodos A, B, C y D:

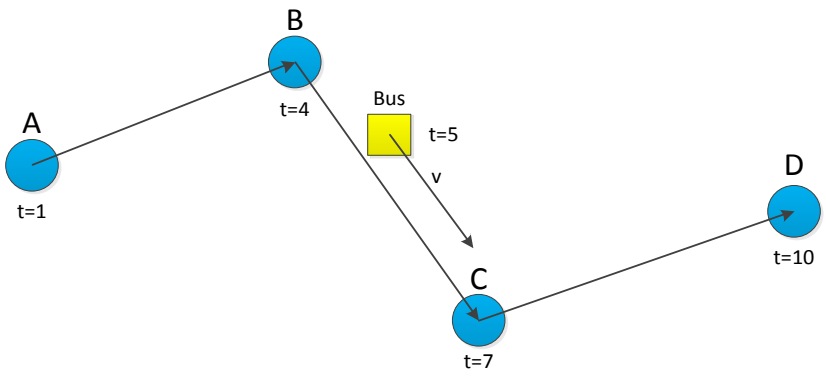

Figura 67: ilustración de recorrido de un bus

El recorrido de la Figura 67 tiene un XML equivalente con la siguiente estructura de datos:

```
<Bus> t=1 Id=1 COMIENZO_RECORRIDO Org=XA,YA Dst=XB,YB tv=3 </Bus>
<Bus> t=4 Id=1 ARRIBO_BUS Org=XB,YB Dst=XC,YC tve=3 </Bus>
<Bus> t=7 Id=1 ARRIBO_BUS Org=XC,YC Dst=XD,YD tv=3 </Bus>
<Bus> t=10 Id=1 FIN_RECORRIDO Org=XD,YD Dst= tv= </Bus>
```

La lectura y proceso de este *XML* se realiza mediante el método *Update(t)* que a su vez es invocado desde el procedimiento *Loop*.

```
//Lectura y proceso del log XML (Script.cs)
Proc Update(long tics)
      //Lectura de líneas del XML y creación de entidades
      switch (ModoLectura)
            case ModoLectura.AlVuelo
                  loadParcialXml(tics) //Le las líneas XML con (t <= tics)
            break

            case ModoLectura.PreCarga
            loadTotalXml() //Se ejecuta una sola vez y lee todo
            break
      end
      //Inspección de buffers, invocación de Update(t) para cada entidad
      //seleccionada con (t <= tics)
      procesarBuses(tics)
      procesarPasajeros(tics)
EndProc
```

La herramienta de visualización mantiene dos colecciones del mismo tipo por cada entidad (Bus o Pasajero), una privada y otra pública. La colección privada es mantenida por los métodos ***loadXml*** del seudocódigo y la pública por los métodos ***procesarEntidad***.

Los métodos ***loadXml*** realizan la lectura parcial hasta el tiempo *t* o total de las entradas del XML creando un objeto por cada entrada procesada, luego estos objetos son almacenados en la colección privada que actúa de buffer. Por otro lado los métodos ***procesarEntidad*** inspeccionan el buffer de la entidad correspondiente (Bus o Pasajero), recuperando únicamente los objetos que deben ser visibles en ese tiempo *t* y copiándolos en la colección pública (si el objeto está repetido se sobrescribe).

La Figura 68 muestra la evolución partiendo del log *XML* hasta la colección pública que es la que expone al exterior los objetos que son redibujados.

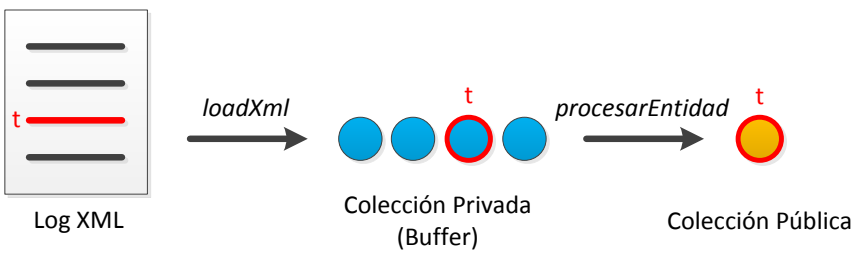

Figura 68: proceso de parsing y actualización de entidades

Los objetos generados por los métodos ***loadXml*** son objetos planos con métodos *get* y *set* para cada uno de los atributos de una línea del XML. Para el caso de modo de lectura de pre-carga, se ejecuta en el `Update(t)` el método ***loadParcialXml*** y la lectura del archivo se realiza completamente sin considerar el tiempo la primera vez que es invocado, mientras que todas las invocaciones siguientes serán ignoradas (no tiene sentido recargar el *XML* en memoria como objetos planos si ya se hizo una vez) hasta que la simulación se detenga desde la UI y sea reiniciada.

5.6.4. Coordenadas intermedias de una entidad

Volviendo a la Figura 67 y al log *XML* que la describe, se plantea la pregunta de cómo saber la posición de la entidad para *t=5*, si solo se disponen de coordenadas en el log para los tiempos *t = 1, 4, 7 y 10*.

Como se mencionó anteriormente, cada línea del archivo *XML* está etiquetada con un código de acción, cada uno de estos códigos se corresponde con un evento en particular del modelo de simulación. Los códigos tratados se presentan en la Tabla 32.

Bus	Pasajero
COMIENZO_RECORRIDO	COMIENZO_VIAJE
ARRIBO_NODO_VIAL_LLENO	ARRIBO_PARADA
ARRIBO_NODO_VIAL	BAJA_PARADA_INTERMEDIA
FIN_RECORRIDO	ARRIBO_PARADA_FINAL
	SUBE_BUS
	LLEGA_CENTROIDE

Tabla 32: Códigos posibles de líneas de log

Cada vez que se lee una línea *XML* y se crea una nueva entidad, si el código asociado a la línea está en la tabla, se considera además la segunda coordenada y se calcula el vector velocidad *v* mediante operaciones algebraicas utilizando los atributos *origen= (XA, YA), destino= (XB, YB)* y el tiempo de viaje entre origen y destino *tv*. Estas operaciones se encapsulan en el método *Init*:

```
Proc Init()
      //Partiendo del origen y destino se calcula el vector dirección
      // v = destino-origen
      Vector2D origen(XA, YA);
      Vector2D destino(XB, YB);
      Vector2D v = destino - origen;

      //Con la distancia euclidea entre el origen y el destino y el tiempo
      //que lleva recorrer esa distancia, se calcula la velocidad media.
      double distancia = v.Norma();
      double velocidad_escalar = distancia / tv;

      //Se normaliza la dirección y se multiplica por la velocidad media
      //escalar. El resultado es el vector velocidad de la entidad.
      v.Normaliza();
      v = v * velocidad_escalar;
EndProc
```

Cada entidad dispone de un método `Update(t)`, el objetivo de este método es calcular la posición actual de la entidad para un tiempo arbitrario *t*. Cuando se crea una entidad (porque se leyó del log en el tiempo *to*) esta tiene un origen y una velocidad dadas en forma vectorial utilizando *Vector2D*, para saber cual es la posición de la entidad en un tiempo *t = to + n*, basta con multiplicar el vector velocidad por el escalar *n* y sumar el vector resultante al vector posición.

Posición siguiente = Posición origen + (t x velocidad)

Se observa que no es necesario almacenar ambas coordenadas (origen y siguiente), basta con utilizar el origen para almacenar la posición siguiente y tomar como convención que la posición de una entidad en cualquier momento *t* luego de invocar `Update(t)` está dada por el origen.

Posición origen = Posición origen + (t x velocidad)

Esta última igualdad es lo que se codifica en el método `Update(t)`.

```
Proc Update(long t)
      Vector2D desplazamiento = v * t;
      origen = origen + desplazamiento; //actualiza el nuevo origen
EndProc
```

Continuando el caso del ejemplo para la Figura 67, interesa saber la posición de la entidad para *t=5*, basta con hacer `Bus.Update(5)`, luego el vector de dos dimensiones *origen* de la entidad se actualiza y es utilizado por el método `Draw()` del `Loop` como punto del mapa para dibujar el objeto.

5.6.5. Tipo de dato Vector2D

El tipo de datos *Vector2D* encapsula todas las operaciones necesarias para cálculos de velocidades y desplazamientos de entidades. Implementa comparación de vectores y operaciones algebraicas entre vectores y escalares. No son necesarias las operaciones de producto interno de vectores ni producto vectorial. Toda instancia de *Vector2D* (data value) tiene un par *x, y* que son las coordenadas del vector modelado.

```
//Se calcula la norma como la distancia euclidea de (0,0) y (x,y)
Proc Norm():double
   return Raiz(x^2 + y^2);
EndProc

//Convierte un vector en versor (vector de norma 1).
Proc Normalize()
   Self.x /= Norm();
   Self.y /= Norm();
EndProc

//Operaciones de comparación de vectores.
Proc operator ==(Vector2D u, Vector2D v): bool
   return (Coord(u) == Coord(v));
EndProc

Proc operator <>(Vector2D u, Vector2D v): bool
   return Coord(u) <> Coord(v);
EndProc

//Operaciones de suma, resta y opuesto de vectores.
Proc operator +(Vector2D u, Vector2D v): Vector2D
   return Vector2D(u.x + v.x, u.y + v.y);
EndProc

Proc operator -(Vector2D u, Vector2D v): Vector2D
   return new Vector2D(u.x - v.x, u.y - v.y);
EndProc

Proc operator -(Vector2D u): Vector2D
   return new Vector2D(-u.x, -u.y);
EndProc

//Operaciones entre vectores y escalares.
Proc operator *(Vector2D u, double escalar): Vector2D
   return new Vector2D(escalar * u.x, scalar * u.y);
EndProc

Proc operator /(Vector2D u, double escalar): Vector2D
   return new Vector2D(u.x / escalar, u.y / escalar);
EndProc
```

5.7.Validación

La validación de la herramienta de visualización se realizó utilizando el mismo caso de prueba de la ciudad de Rivera, la cual se ha utilizado en las validaciones de la herramienta *igoR-tp* y el simulador *PublicTransport*.

El usuario realiza la validación de las funcionalidades de la herramienta, en particular las funcionalidades zoom in y zoom out. Por último cargó y ejecutó una simulación a la que le realizó una validación visual y verificó que efectivamente los pasajeros arriban a paradas, realizan esperas, abordan buses, realizan trasbordos y realizan caminatas hacia y desde los centroides. También validó que los buses se mueven normalmente y que se detectan buses llenos en el sistema. Se comprobó además que los mismos realizan los recorridos que tienen asignados.

Es importante destacar que la elección de este caso de prueba en particular permite verificar además, que las tres herramientas que componen este proyecto de grado (como muestra en la sección 5.1) pueden trabajar en forma conjunta.

La Figura 69 muestra la aplicación real validada por el usuario ejecutando el caso de prueba de Rivera.

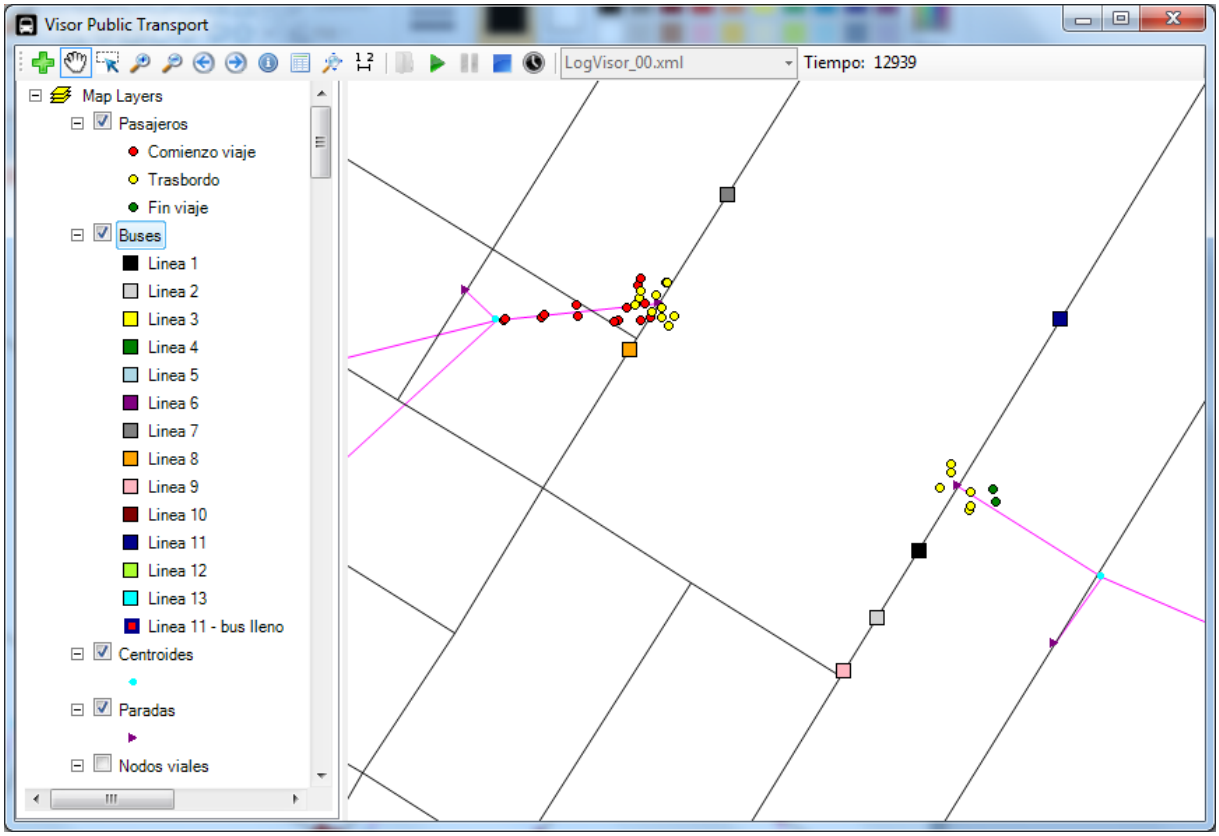

Figura 69: La herramienta de visualización en funcionamiento.

6. Conclusiones y trabajos futuros

Conclusiones

En primer lugar podemos afirmar que se han cumplido con éxito los objetivos del proyecto.

Se cuenta con una nueva versión de la herramienta igoR-tp la cual presenta como principal mejora la posibilidad de insertar paradas en cualquier punto de un eje de calle, logrando de esta forma construir modelos de red que representan de manera más fiel la realidad. Esta nueva versión permite además asignarle un sentido a la parada la cual restringe los buses que pueden detenerse en la misma.

Luego de haber trabajado y evaluado esta herramienta se concluye que la misma es una herramienta válida para el diseño de soluciones del TNDP para casos de pequeño y mediano porte (por ejemplo, la ciudad de Rivera).

Se extendió de manera exitosa el simulador *PublicTransport* con un nuevo módulo de comportamiento de pasajeros (estrategia), que permite viajes con opción de trasbordo. Se le agregaron nuevos indicadores basándonos en los modelos de Baaj y Mahamassani y de Florian-Spiess. Estos indicadores fueron utilizados para realizar la validación del simulador utilizando la nueva estrategia con un caso de estudio artificial. Este caso de estudio fue diseñado especialmente para poder calcular en forma teórica el valor de los indicadores, concluyéndose que los valores obtenidos son muy similares a los teóricos (con cierta tolerancia) lo que nos permitió efectivamente dar por válido el simulador.

Posteriormente se realizó una comparación con el caso de estudio de la ciudad de Rivera. En este caso se compararon los valores obtenidos de los nuevos indicadores para las dos estrategias con los valores presentados en una tesis de maestría para el mismo caso de estudio. De esta comparación podemos concluir que los valores obtenidos para la estrategia directa son muy similares con los presentados en la tesis, mientras que los de la estrategia de trasbordo presentan una diferencia considerable. Esto nos permite afirmar que la solución del caso de estudio de la ciudad de Rivera no favorece los viajes por trasbordo.

En base a las validaciones realizadas podemos concluir que el simulador con cualquiera de sus dos estrategias es una herramienta válida para evaluar soluciones del TNDP para casos de pequeño y mediano porte.

Por último se desarrolló una nueva herramienta para visualizar las simulaciones, la misma cuenta entre otras cosas con la funcionalidad de zoom y fue implementada utilizando tecnologías recientes de libre distribución. La elección de la tecnología a utilizar surge como resultado de una investigación detallada de las distintas alternativas tecnológicas disponibles. Del mismo modo se analizaron distintas posibilidades para la comunicación de la información de la simulación y su representación visual. Como solución a esta problemática se diseñó un protocolo de comunicación entre el simulador y la nueva herramienta de visualización mediante un archivo *XML*. De esta manera la información dinámica de la simulación se independiza de su salida visual, permitiendo en un futuro de ser necesario el desarrollo de nuevas herramientas de visualización.

Finalmente en base a todo lo expuesto se concluye que en el contexto de la planificación estratégica de un TPUC, se pueden utilizar estas herramientas para el ciclo de generación y evaluación de soluciones del TNDP en sistemas de pequeño y mediano porte.

Trabajos futuros

En esta sección se describen las modificaciones que consideramos deben ser tenidas en cuenta en un futuro para la mejora de los modelos y las herramientas tratados a lo largo de este trabajo.

Mejoras sobre los modelos de red:

- Permitir más de un centroide por zona: Este cambio obligaría a reformular el concepto de centroide por el de punto de atracción pues ya no estaríamos hablando del punto centro de una zona. Esta mejora permite ubicar en una zona más de un punto donde originar y destinar pasajeros. Por ejemplo, si en una zona actual se tiene un hospital, un shopping y una escuela el modelo actual resume todo en el centroide, con el nuevo concepto es posible asignar a cada uno de estos elementos un nuevo punto con sus caminatas y su propia demanda. Como resultado de este cambio se debe reformular también la matriz de demanda para que considere los nuevos puntos de atracción.
- Extender el modelo de red vial añadiendo una capa de red de caminatas: La nueva capa representa todas las posibles rutas por las que un pasajero puede caminar, además esta capa debe incluir como nodos a las paradas y los centroides. Toda caminata que se defina en el modelo de red debe pertenecer a esta nueva capa de red de caminatas. La ventaja de disponer de esta nueva capa es que independiza el movimiento de pasajeros de la red vial pudiéndose modelar sendas peatonales, senderos, cruces peatonales, etc.

Mejoras sobre igoR-tp:

- Actualización de la tecnología de mapas: Actualmente igoR-tp utiliza el componente *ActiveX MapWinGIS* el cual presenta problemas (descritos en el *Apéndice E - Conclusiones*) con su instalación en algunos ambientes de trabajo, en particular a largo de este proyecto en ambientes con *Windows Xp* como ser el *PC* del usuario y el de un integrante de este grupo. La arquitectura de igoR-tp (ver [2] pág. 24) encapsula el componente de forma que es posible remplazarlo por otro sin afectar el funcionamiento del mismo. Se recomienda el remplazo de *MapWinGIS* por *DotSpatial* el cual se probó con éxito en la herramienta de visualización.
- Incluir en el módulo de construcción la generación de redes viales en forma manual: Actualmente como entrada de un nuevo proyecto del módulo de construcción es necesario un archivo con el grafo de la red vial, sin embargo la tecnología con la que está elaborado igoR-tp permite el diseño de dicho grafo.
- Incluir en el módulo de manipulación el concepto de línea de bus: Actualmente para vincular los recorridos con las líneas y sus frecuencias se utiliza un archivo externo “líneas.txt” cuyo mantenimiento es responsabilidad del usuario. Nuestra experiencia nos indica que elaborar ese archivo para casos de estudio medianos o grandes (como el caso de Rivera), requiere un esfuerzo considerable y puede introducir errores difíciles de detectar. Una mejora de igoR-tp es incluir el concepto de línea a la cual se le pueda asociar recorridos previamente definidos y frecuencias de forma que el archivo “líneas.txt” se genere automáticamente.

- Introducir el sentido o flecha en los tramos de un recorrido de forma que no se pueda definir recorridos que permitan circulación en contra del sentido de sus tramos.
- En caso que se considere añadir una red de caminatas, sería conveniente implementar una nueva funcionalidad en el módulo de construcción que permita seleccionar dos paradas y que la herramienta genere automáticamente la poligonal de la caminata utilizando un algoritmo de camino más corto sobre la red de caminatas.

Mejoras sobre simulador:

- Implementar una estrategia que realice caminatas entre paradas: Por ejemplo en la estrategia de trasbordo desarrollada en este proyecto cuando un pasajero desciende en una parada intermedia actualmente espera en la misma, una extensión interesante sería darle la posibilidad (si es más conveniente) al pasajero de trasladarse a otra parada a través de una caminata para luego desde ahí realizar el trasbordo.
- Enriquecer el simulador con una interfaz de usuario de ventanas: Actualmente el simulador *PublicTransport* cuenta con una variedad de opciones para su ejecución las cuales son manipuladas desde líneas de comandos mediante argumentos y banderas (ver *Apéndice E - sección 2*).
- Mejorar la información ofrecida en el archivo “Validacion_NN.txt”: Actualmente el archivo “Validacion_NN.txt” (especificado en sección 4.6.2) ofrece en forma compacta para cada pasajero los tiempos de espera en parada y las líneas abordadas por cada uno. Para llevar un seguimiento más completo del pasajero es necesario incluir los tiempos en lo que se producen los eventos arribo a parada y subida a bus. De esta forma es posible saber donde está el pasajero para un instante dado de la simulación.

Mejoras sobre la herramienta de visualización:

- Habilitar la posibilidad de retroceder la simulación en el tiempo ya sea mediante un botón de ejecutar inverso o mediante una barra de desplazamiento, imitando los controles tradicionales de software de reproducción de sonido o video. Para retroceder en el tiempo de simulación basta con modificar el loop principal (sección 5.6) para que en vez de avanzar, retroceda en el tiempo. Implementar una barra de desplazamiento para avanzar y retroceder a voluntad implicaría modificar los algoritmos de lectura del log *XML*.
- Desplegar información contextual de los Buses, Pasajeros y Paradas. Por ejemplo, para los buses se podría desplegar cantidad de pasajeros en viaje, tiempo de recorrido, línea a la que pertenece. Para los pasajeros sería conveniente desplegar el destino, el tiempo total de viaje hasta el momento. Para las paradas es interesante desplegar cantidad de pasajeros esperando discriminando si están esperando trasbordo o primer viaje.
- Permitir habilitar o deshabilitar los elementos que representan los buses de una línea determinada.
- Almacenar la configuración visual de los elementos que componen la simulación: actualmente existe la posibilidad de modificar colores y formas de los elementos, pero no es posible guardarla como una preferencia de usuario para usar en reproducciones de simulación futuras.

- Ofrecer la posibilidad de personalizar las figuras de los elementos de la simulación mediante iconos en algún formato gráfico popular (*gif*, *png*, *jpg*, etc.).
- Permitir cargar imágenes como fondo o modificar el color del mismo.
- Añadir un componente visual que despliegue la evolución de los indicadores (Z1, tiempo espera en parada y Z2) a lo largo de la simulación.

Índice de ilustraciones

Índice de tablas

Apéndice A – Implementación de los cambios en igoR-tp

Como se explica en el capítulo 3, la nueva versión de igoR-tp permite que las paradas ya no sean únicamente nodos viales sino que puedan ubicarse en cualquier lugar de la calle. A continuación se detallarán los cambios en la implementación que hicieron posible este cambio funcional.

Para poder implementar la nueva representación de parada, es necesario ubicar una parada en cualquier lugar de una calle sin ser en una esquina. Al ser la representación de una calle una única línea (eje), de alguna manera se debe determinar en qué vereda está la parada para asignarle un sentido a la misma. Esto presentaba algunos problemas a nivel de programación que se tuvieron que resolver:

Problema 1: ¿Cómo hacer para ubicar la parada sobre la línea que representa la calle?

Para agregar una parada, el usuario debe seleccionar la opción correspondiente y luego hacer clic en el lugar del mapa donde desea ubicar la parada. Para que la ubicación sea válida esta tiene que estar sobre una de las líneas del mapa que representa una calle y no coincidir con un nodo vial existente (cruce de calles o parada).

Para cumplir con la primera parte se hizo lo siguiente: al hacer clic sobre el mapa con el objetivo de ubicar la parada, se crea una clase Extents con una cierta tolerancia con el fin de determinar si cerca del lugar donde se hizo el clic hay alguna calle. (La clase Extents es un objeto MapWinGis que sirve para determinar un área del mapa, en nuestro caso representa un cuadrado de lado=4 con centro en el punto del mapa donde el usuario hizo el clic). Dicho objeto se crea en el evento mouseUp y se pasa como parámetro a la función GenerarParada.

La función *GenerarParada* intersecta dicho Extent con la capa de calles y la capa de nodos viales. En caso que la intersección con la capa de nodos viales no de vacía, significa que dentro de esta área hay definido un nodo vial, por lo tanto no se puede agregar una parada allí. Al intersectar con la capa de calles se verifica que el punto seleccionado esté lo suficientemente próximo a una línea de calle, en caso de que no lo esté o el resultado de la intersección sea más de una línea, se muestra un mensaje de error advirtiendo dicha situación.

Luego se verifica la cercanía del punto a una calle, para esto hay que determinar el punto exacto donde se ubicará la parada. Dado que no se puede exigir al usuario que haga clic exactamente sobre una línea de calle y la biblioteca MapWinGis no cuenta con esta funcionalidad, se realiza el cálculo en la función AgregarParada:

```
public void agregarParada(int shp_arista, MapWinGIS.ExtentsClass origen,
int idNodoSentido)
```

Donde shp_arista es el id del shape intersección del Extent antes mencionado y la capa de calles, origen es el objeto Extent e idNodoSentido representa el sentido de la parada.

Las calles son representadas por un shapefile de polilíneas, lo que significa que cada shape es una polilínea. Se define polilínea a un conjunto de segmentos de recta conectados. (Ver Figura 70)

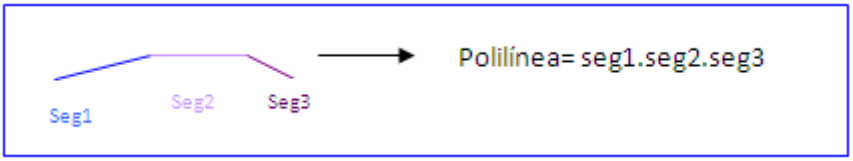

Figura 70: Polilínea

El resultado de la intersección entonces, es el id del shape o sea el id de la polilínea que representa la calle. En base a este resultado y al punto donde el usuario hizo clic debemos determinar el lugar exacto donde colocar la parada (Ver Figura 71).

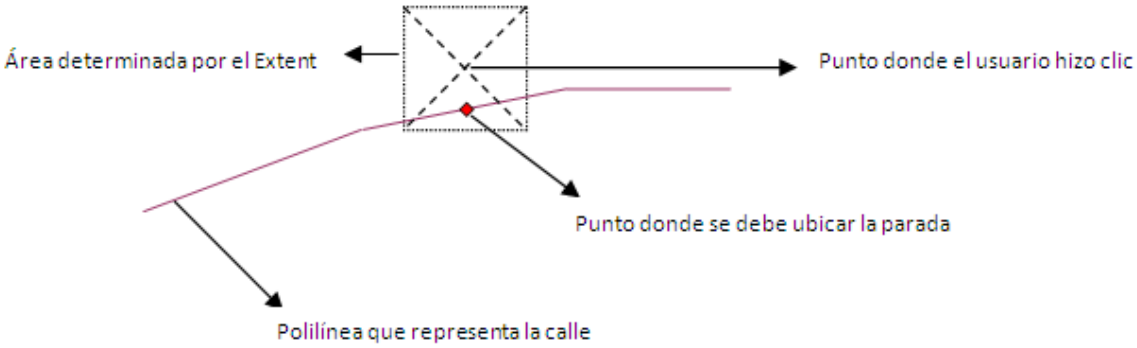

Figura 71: Punto donde se ubica la parada

A continuación se determina el segmento de la polilínea más cercano al punto donde se realizó el clic y luego hallar la proyección ortogonal del punto sobre dicho segmento. De esta manera el punto encontrado se ubica sobre la calle (Ver Figura 72).

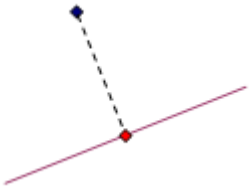

Figura 72: proyección ortogonal

Problema 2: ¿Cómo dividir la arista al agregar una parada?

Originalmente la capa que representa las calles era un shapefile de polilíneas llamado capaAristas donde para cada shape se contaba con la siguiente información: id del nodo vial origen, id del nodo vial destino, largo y costo, siendo todos estos campos calculados por la aplicación al crearse los nodos viales. Esta información determina los datos de una arista.

La nueva versión mantiene los campos originales de la capa de aristas, pero no es suficiente ya que se presenta el problema de que cuando se coloca una nueva parada en la calle se debe actualizar la información de las aristas. Por este motivo se tiene que dividir la arista original en dos nuevas aristas e eliminar la arista original (ver sección 3.3.1 Sentido de una parada- Ejemplo agregar parada).

Apéndice B – Modificaciones tecnológicas

Introducción

El presente apéndice describe las modificaciones tecnológicas realizadas en las aplicaciones heredadas del Proyecto de Grado de la carrera de Ingeniería en Computación de la Facultad de Ingeniería realizado por Aldaz-DeLeón en el año 2010 [3]. Es un complemento de los capítulos 3 y 4 el cual surge de la necesidad de dejar registradas las modificaciones no funcionales realizadas sobre dichas aplicaciones.

Las aplicaciones modificadas son la herramienta *igoR-tp* versión 2.0 y el simulador *PublicTransport* versión 1.0. Es importante destacar que todos los cambios tecnológicos realizados fueron planteados al usuario y validados por el mismo.

Este informe se encuentra estructurado en dos secciones, cada uno dedicado a las aplicaciones mencionadas anteriormente. En cada sección se realiza una introducción del estado tecnológico de las aplicaciones al comienzo de este proyecto, los problemas encontrados y las soluciones propuestas.

Se recomienda la lectura previa de los capítulos 3 y 4 para familiarizarse con conceptos relativos a *igoR-tp* y el simulador *PublicTransport*.

Modificaciones tecnológicas de igoR-tp 2.0

Contexto tecnológico inicial

Para conocer el estado tecnológico de igoR-tp 2.0 se realizó la lectura de la documentación técnica proporcionada en el informe de proyecto de grado [3], en particular el *Apéndice C*. Una vez finalizada la lectura no se logró una comprensión completa de la situación del mismo.

Se propone entonces realizar una inspección de código y luego una ejecución del código compilado en un ambiente de características similares a las del usuario (Windows XP 32 bits).

Inspección de código

El primer paso dado para obtener el estado tecnológico de igoR-tp 2.0 fue la inspección del código fuente, la configuración de las soluciones y el análisis de la distribución de los archivos fuentes. Se concluyó que el código fuente original de *igoR-tp* 2.0 se encuentra repartido en distintas carpetas cada una con una solución *Visual Studio 2008* (VS2008 de aquí en más), como muestra la Figura 73.

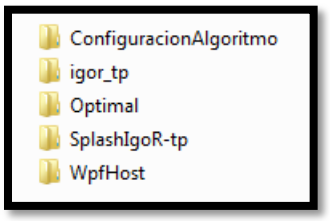

Figura 73: Distribución del código de igoR-tp 2.0

Luego de analizar las soluciones VS2008 alojadas en cada carpeta, se descubrió que había soluciones con más de un proyecto (Optimal y WpfHost), también que la carpeta WpfHost con la solución del mismo nombre contenía el código fuente del *módulo de construcción* de igoR-tp y que la carpeta Optimal con la solución del mismo nombre contenía el código fuente del *módulo de manipulación* de igoR-tp.

La Tabla 33 resume las relaciones entre las carpetas de soluciones VS2008 de la Figura 73 con los módulos funcionales de igoR-tp 2.0.

Carpeta	Módulo de igoR-tp	# proyectos VS2008
ConfiguracionAlgoritmo	Módulo de algoritmos	1
Igor_tp	Ventana principal de igoR-tp	1
Optimal	Módulo de manipulación	2
SplashIgoR-tp	Pantalla de presentación (Splash Screen)	2
WpfHost	Módulo de construcción	1

Tabla 33: Relación entre soluciones VS2008 y módulos de igoR-tp 2.0

Luego de inspeccionar las configuraciones de las soluciones VS2008 alojadas en cada carpeta, se concluyó que la herramienta igoR-tp 2.0 se implementó utilizando las siguientes tecnologías:

- Plataforma de desarrollo *.net 3.5* [29].
- Entorno de desarrollo *Visual Studio 2008* .
- Lenguaje de programación *C# 3.0* [30].
- Interfaz de usuario *Windows Presentation Foundation* [28] y *WinForms* [36].
- Componente *MapWinGIS ActiveX* versión 4.6 [37].

La distribución del proyecto igoR-tp 2.0 dificulta la depuración ya que cada módulo de igoR-tp 2.0 es un ejecutable distinto que trabaja en un proceso separado. Cada uno de estos ejecutables se corresponde con una solución VS2008, por lo que es necesario trabajar con varias instancias de VS2008 si se quiere probar más de un módulo de igoR-tp 2.0.

Por otro lado, los nombres de varios proyectos y sus carpetas no mantienen ninguna relación con los módulos funcionales de igoR-tp (módulo de construcción, módulo de manipulación o módulo de algoritmos) que implementan.

La Tabla 34 muestra los nombres de los proyectos contenidos en cada solución del código fuente original y la relación que guardan con la herramienta igoR-tp 2.0.

Proyecto	Solución	Módulo igoR-tp
ConfiguracionAlgoritmo	ConfiguracionAlgoritmo	Módulo de algoritmos
Igor_tp	Igor_tp	Ventana principal de igoR-tp
Optimal	Optimal	Módulo de manipulación
MWGisManipulacion	Optimal	Módulo de manipulación
SplashIgoR-tp	SplashIgoR-tp	Pantalla de presentación(Splash Screen)
WpfHost	WpfHost	Módulo de construcción
MWGISLib	WpfHost	Módulo de construcción

Tabla 34: Relación entre proyectos de fuentes y los módulos de igoR-tp 2.0

Compilación y prueba

Una vez compilado el código original, se detectó que la configuración del mismo condiciona la ejecución de la herramienta en sistemas operativos *Windows* de 64 bits. El problema detectado fue que no se tuvo en cuenta que el componente MapWinGIS es un componente binario de 32 bits, y debe forzarse la generación de igoR-tp 2.0 en 32 bits de lo contrario el componente MapWinGIS no puede ser invocado desde el *módulo de construcción* y el *módulo de manipulación* en ambientes de 64 bits (ver Figura 74).

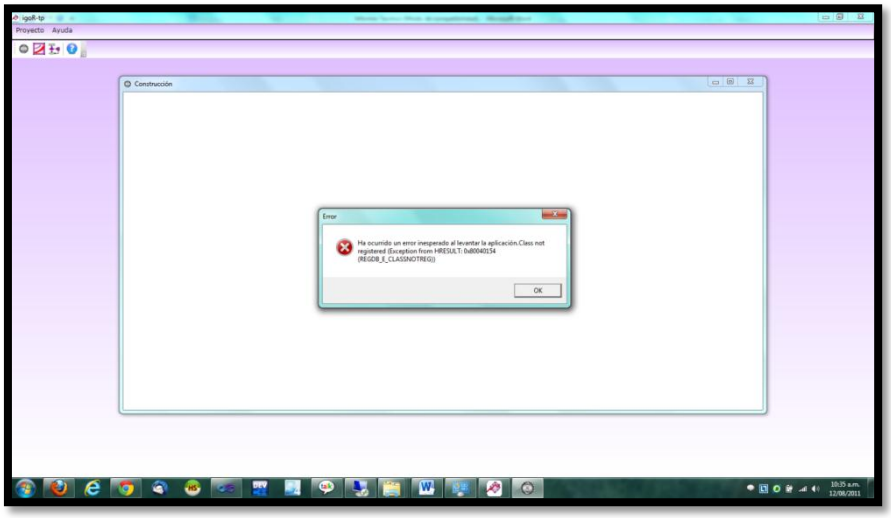

Figura 74: Error de ejecución del módulo de construcción en ambiente Windows 7 Home x64

Modificaciones propuestas

Se proponen cuatro modificaciones en el código y en la configuración del mismo al usuario:

1. Reorganización y reingeniería del código de igoR-tp 2.0, para que el código esté ubicado en una sola solución en una única carpeta y además que se genere un único ejecutable.
2. Garantizar la ejecución en sistemas *Windows XP*, *Vista* o *7* de 32 o 64 bits indistintamente.
3. Actualización de las tecnologías utilizadas, a sus versiones más recientes siempre que esto sea posible.
4. Renombrado de proyectos y soluciones, para facilitar el mantenimiento futuro del código.

Cambios tecnológicos realizados

Los cambios tecnológicos propuestos anteriormente se aplicaron sin contratiempos. El resultado final de realizar los cambios, es una solución auto contenida, que incluye todos los proyectos de la Tabla 34 renombrados, con todas sus dependencias y archivos necesarios para su ejecución.

A partir de ahora se hará referencia a la antigua aplicación como igoR-tp 2.0 y a la nueva aplicación resultante de aplicar los cambios tecnológicos como igoR-tp 3.0, esto es debido a que sobre igoR-tp 3.0 se realizan los cambios funcionales descritos en el capítulo 3.

La Tabla 35 muestra el resultado del renombrado de proyectos y la relación que todavía mantienen con los módulos funcionales de la herramienta igoR-tp 2.0.

Proyecto anterior	Proyecto nuevo	Módulo igoR-tp
ConfiguracionAlgoritmo	ConfiguracionAlgoritmo	Módulo de algoritmos
Igor_tp	IgorTp	Ventana principal de igoR-tp
Optimal	Optimal	Módulo de manipulación
MWGisManipulacion	MWGisOptimal	Módulo de manipulación
SplashIgoR-tp	SplashScreen	Pantalla de presentación(Splash Screen)
WpfHost	Contruccion	Módulo de construcción
MWGISLib	MWGisContruccion	Módulo de construcción

Tabla 35: Renombrado de proyectos de igoR-tp 2.0

No todas las tecnologías utilizadas en igoR-tp 2.0 descritas en el apartado anterior fueron actualizadas, las tecnologías que se actualizaron y su versión se listan en la Tabla 36:

igoR-tp 2.0	igoR-tp 3.0
.net framework 3.5	*.net framework 4.0*
Visual Studio 2008	*Visual Studio 2010*
Lenguaje de programación *C# 3.0*	Lenguaje de programación *C# 4.0*
MapWinGIS ActiveX 4.6	*MapWinGIS ActiveX 4.8.2*

Tabla 36: Tecnologías actualizadas

La actualización del lenguaje de programación queda implícita en el cambio de entorno de desarrollo (framework) y es compatible hacia atrás, esto implica que la nueva solución permite codificar utilizando los recursos que ofrece la actualización del lenguaje C *# 4.0* [38].

La nueva solución llamada IgorTp, incluye también un proyecto (el único del tipo ejecutable) llamado IgorTp y sus archivos se distribuyen en una única carpeta que también se llama IgorTp. Los demás proyectos se modificaron y se añadieron como proyectos de biblioteca. Esta solución tiene la ventaja de permitir acceder a todo el código con una sola instancia de *Visual Studio* y depurar también en la misma instancia.

La solución fue configurada (ver Figura 75) para generar siempre código CLR x86 (de 32 bits) [39], de esta forma se garantiza que el *módulo de construcción* y el *módulo de manipulación* trabajen adecuadamente con el control ActiveX MapWinGIS que es de 32bits evitando el problema de la Figura 74 en ambientes de *Windows de* 64 bits.

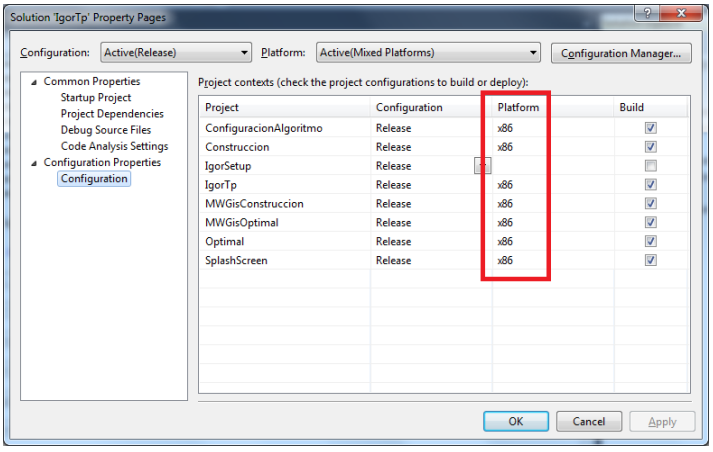

Figura 75: Configuración que garantiza ejecución en ambientes de 64 bits

La Figura 76 muestra la solución final en *Visual Studio 2010* (VS2010 de aquí en mas) resultante, con todos los proyectos en una misma solución. La misma incluye todos los archivos necesarios para la ejecución de igoR-tp 3.0.

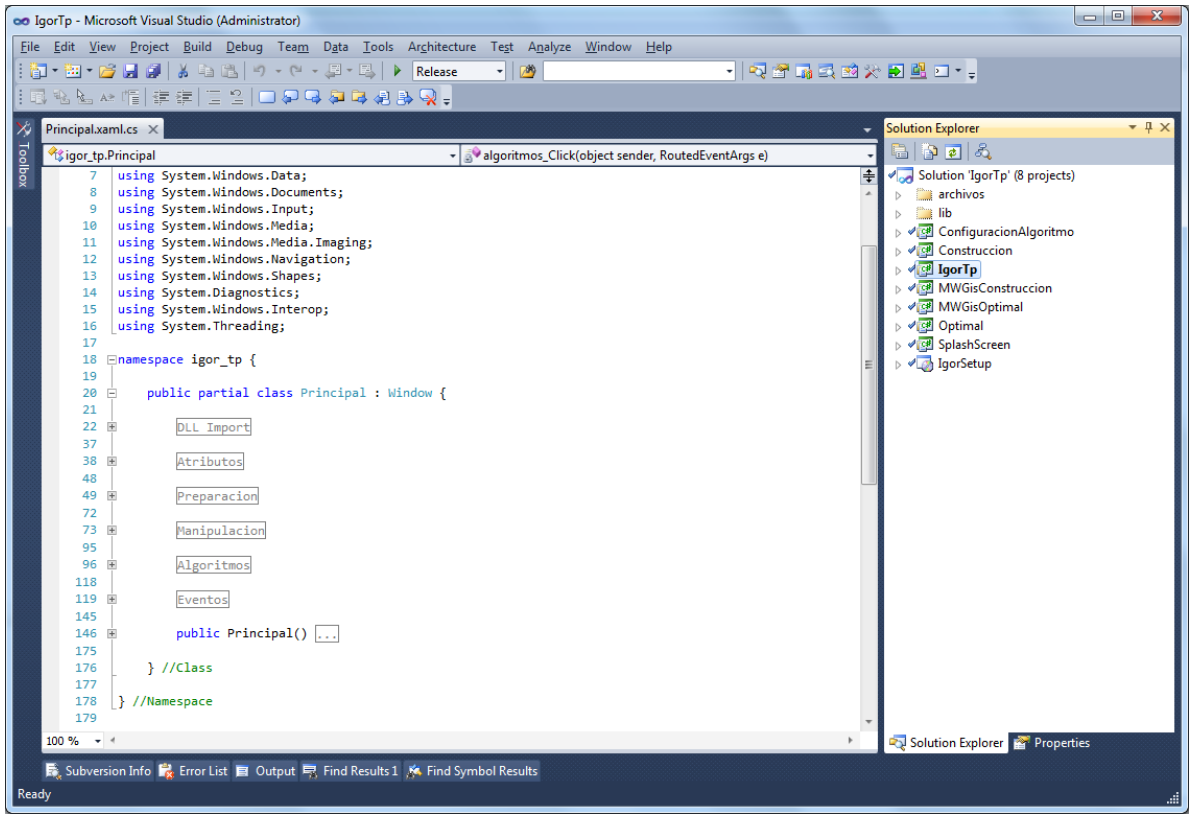

Figura 76: Solución final autocontenida de igoR-tp.

Esta solución está dotada con un nuevo proyecto llamado *IgorSetup* el cual una vez compilado genera un archivo instalador llamado *IgorSetup.msi*. Con este archivo instalador se está en condición de instalar una versión completa de igoR-tp 3.0 y el componente de terceros MapWinGIS en cualquier ambiente *Windows* que disponga de la siguiente configuración:

- *.net framework 4.0*
- *Windows Installer 3.1* [40]
- Runtime *Microsoft Visual C++2008 Redistributable Setup* [41]

Se recomienda la lectura del *Apéndice C*, para profundizar detalles acerca de la utilización del instalador *IgorSetup.msi.*

Por último, es importante destacar que el usuario validó las modificaciones tecnológicas propuestas. Por otro lado, validó también el nuevo instalador, realizando una instalación y ejecución de la aplicación igoR-tp 3.0 en ambientes *Windows Vista* de 32 bits y *Windows XP* de 64 bits.

Modificaciones tecnológicas del simulador PublicTransport 1.0

Contexto tecnológico inicial

Al igual que para igoR-tp 2.0, la primera medida adoptada para conocer la situación tecnológica del simulador PublicTransport 1.0 fue la lectura del informe de proyecto de grado [3]. En esta ocasión, la documentación especifica con precisión las tecnologías utilizadas. La siguiente lista resume la situación tecnológica del simulador PublicTransport 1.0:

- Lenguaje de programación *C++ y* Compilador *MinGW 3.4.2* [42].
- Entorno de desarrollo *Dev-C++ 4.9.9.2* [43].
- Biblioteca para desarrollo de simuladores *EOSimulator 1.1.1* [11].
- Biblioteca multimedia *SDL 1.2.11* [13].
- Biblioteca de manejo de imágenes *SDL_image 1.2.5* [44].
- Biblioteca de dibujo vectorial *AGG2.3* [45].
- Biblioteca de manejo de fuentes tipográficas *FreeType 2.1.10* [46].
- Biblioteca *Boost C++ Libraries 1.33.1* [47].

Nuevamente, se propone realizar una inspección de código y luego una ejecución del código compilado en un ambiente de características similares a las del usuario (Windows XP 32 bits).

Inspección de código

El código se distribuye mediante una carpeta que contiene un único proyecto *Dev-C++* (Figura 77). La inspección del mismo revela que existen múltiples archivos de código que no son utilizados ni referenciados desde ningún otro, también revela que existen múltiples definiciones de clases en un mismo archivo.

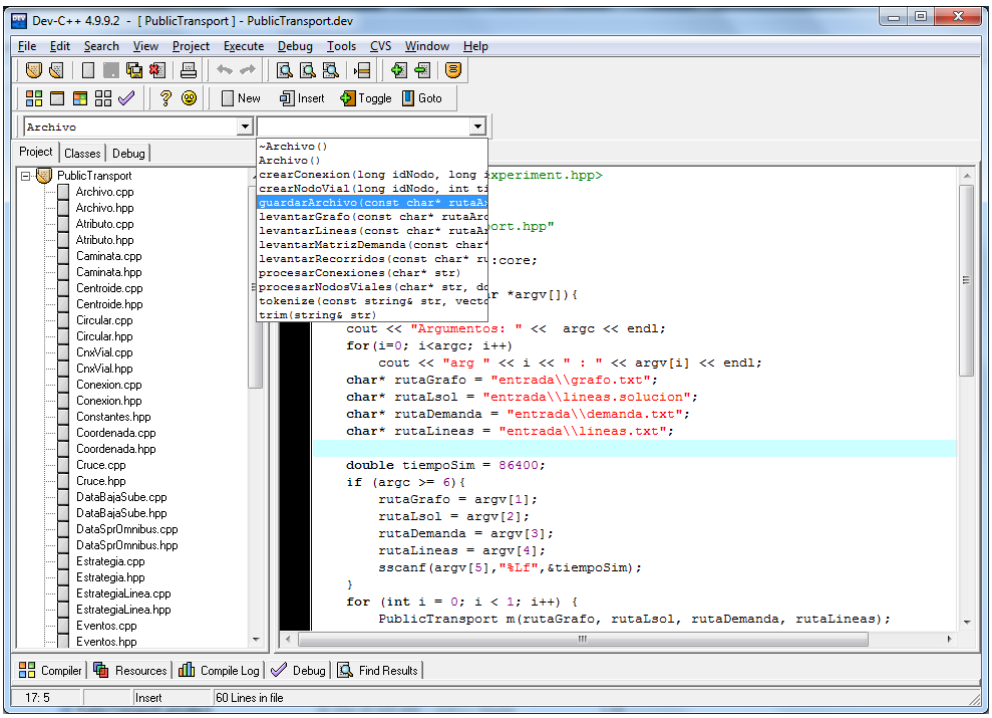

Figura 77: Entorno de trabajo Dev-C++

Compilación y prueba

El código compila sin problemas siguiendo la documentación proporcionada en [3]. Se observa que la salida estándar de la aplicación está sobrecargada de mensajes de depuración utilizados a lo largo del desarrollo de la versión 1.0.

Se observa que no es posible depurar utilizando el entorno *Dev-C++* mas allá de las clases que componen el simulador, pues el resto de las bibliotecas se distribuyen en forma binaria. Es de interés poder depurar el código del simulador utilizando el entorno *Dev-C++*, por lo que se considera la posibilidad de incluir el código de *EOSimulator* dentro del proyecto y así poder depurar la clases de este último utilizando código *C++*.

Dado que el proyecto que mantiene el entorno de desarrollo *Dev-C++* se encuentra discontinuado y en estado beta, se recomienda al usuario abandonar dicho entorno de desarrollo y utilizar el mismo entorno que utiliza igoR-tp (*Visual Studio*). Otro motivo que fundamenta el cambio de entorno de desarrollo, es que la mayoría de los integrantes del grupo encargado de llevar a cabo este nuevo proyecto de grado cuenta con experiencia (más de cuatro años de experiencia laboral) en el uso del entorno y el depurador de *Visual Studio*.

Modificaciones propuestas

Se sugiere al usuario realizar una migración de código a *Visual Studio* e integrarlo como un proyecto adicional *Visual C++* a la solución IgorTp descrita en el apartado anterior. El usuario descarta esa idea, pero aprueba la migración a un proyecto *Visual Studio* nuevo.

Se proponen al usuario las siguientes modificaciones no funcionales:

1. Migración del código del entorno *Dev-C++* a *Visual Studio*.
2. Eliminación de las clases y archivos innecesarios.
3. Inclusión del código fuente de EOSimulator como un proyecto de biblioteca.
4. Reorganización del código fuente, aplicando la política de un archivo fuente por clase (en este caso como el lenguaje es *C++*, es una pareja de archivos *.h* y *.cpp* por clase).
5. Remover todas las salidas estándar utilizadas en depuraciones previas.
6. Modernizar la entrada desde línea de comandos permitiendo el uso de banderas estilo consolas de Unix.
7. Permitir desactivar la salida visual SDL mediante el uso de banderas desde línea de comandos.
8. Actualizar la biblioteca *Boost C++ Libraries* a la versión 1.47.0 (la más reciente al momento de realizar estas modificaciones).

Cambios tecnológicos realizados

La Figura 78 muestra el resultado de integrar en una única solución *EOSimulator* y *PublicTransport*, se observa que el código se encuentra separado en carpetas lógicas lo que facilita la ubicación de los archivos fuentes (contrastar con Figura 77).

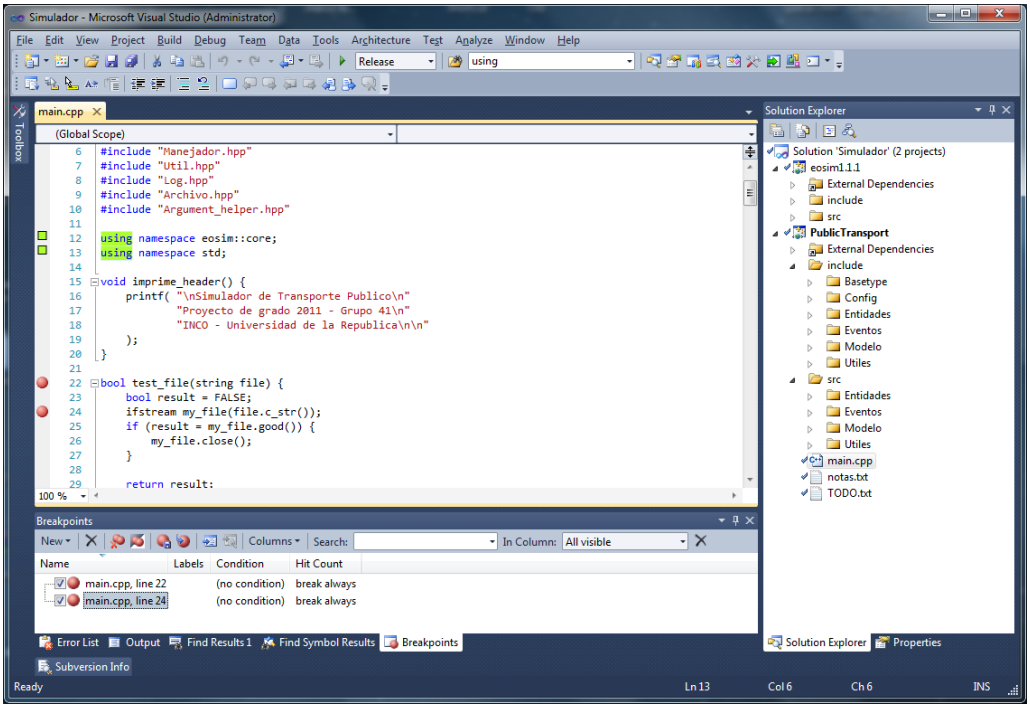

Figura 78: Migración completa a Visual Studio

La Figura 79 muestra un caso de uso del simulador desde línea de comandos, se eliminaron todas las salidas estándar antiguas y se sustituyeron por un conjunto de datos mínimo donde se muestra desde qué archivos se cargan los datos, el número de corrida que se está ejecutando y por último en qué carpetas se vuelcan los archivos de log y reportes de la simulación.

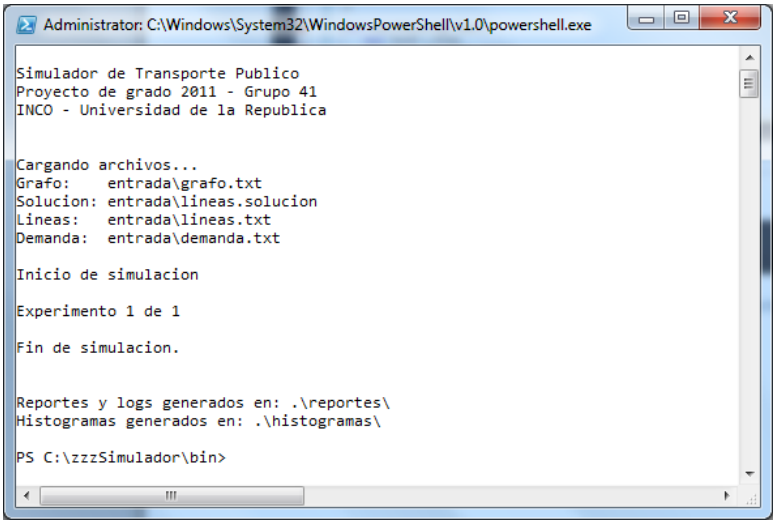

Figura 79: Nueva salida estándar del simulador PublicTransport

Todos los cambios propuestos se implementaron con éxito y fueron validados por el usuario.

Apéndice C – Manual de usuario Herramienta de visualización

1. Instalación

La Herramienta de visualización del simulador *PublicTransport* dispone de un instalador para *Windows XP* o superior. Para instalar la Herramienta de visualización es requisito que el equipo anfitrión disponga del entorno de desarrollo *.Net 4.0* instalado.

Una vez asegurado el requisito previo se deben seguir los siguientes pasos:

1. Buscar el archivo llamado *VisorSetup.msi* provisto en el DVD junto a este documento y ejecutarlo realizando doble clic o botón derecho e Instalar (Figura 80).

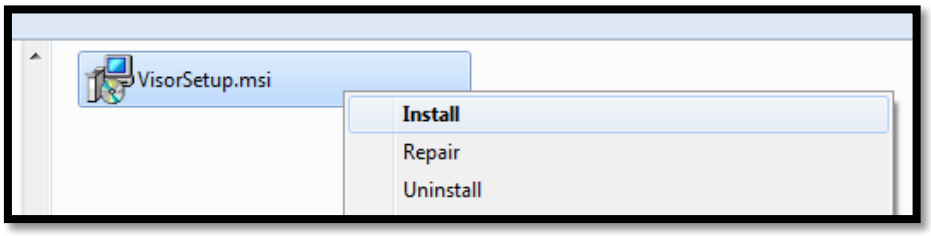

Figura 80: Ejecutable instalador

2. A partir de aquí da comienzo el asistente de instalación por lo que debería ver la siguiente pantalla (Figura 81).

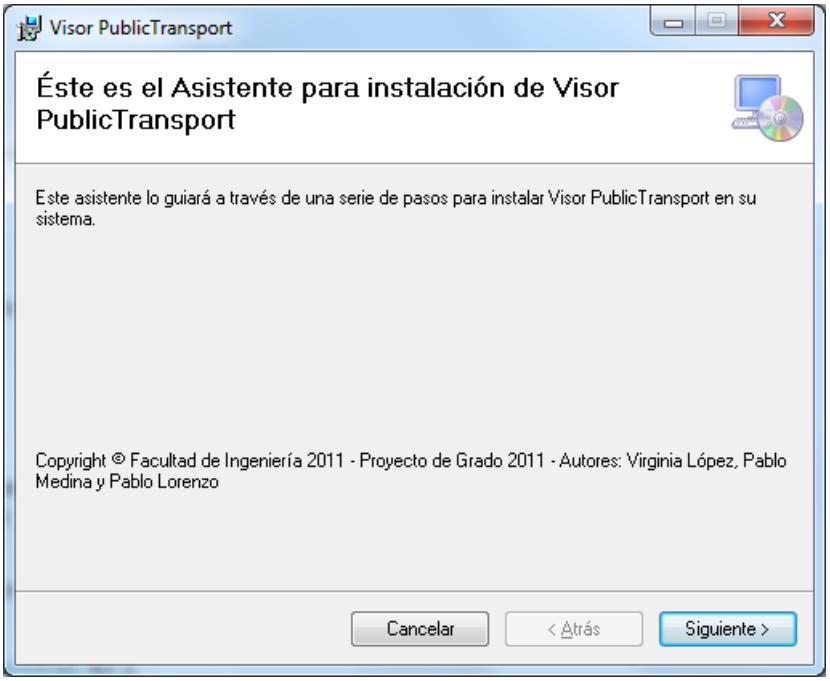

Figura 81: Introducción del instalador

3. Luego de la introducción el asistente pregunta por la carpeta destino (siempre ofrece una por defecto). Se sugiere instalar eligiendo la opción para todos los usuarios (Figura 82).

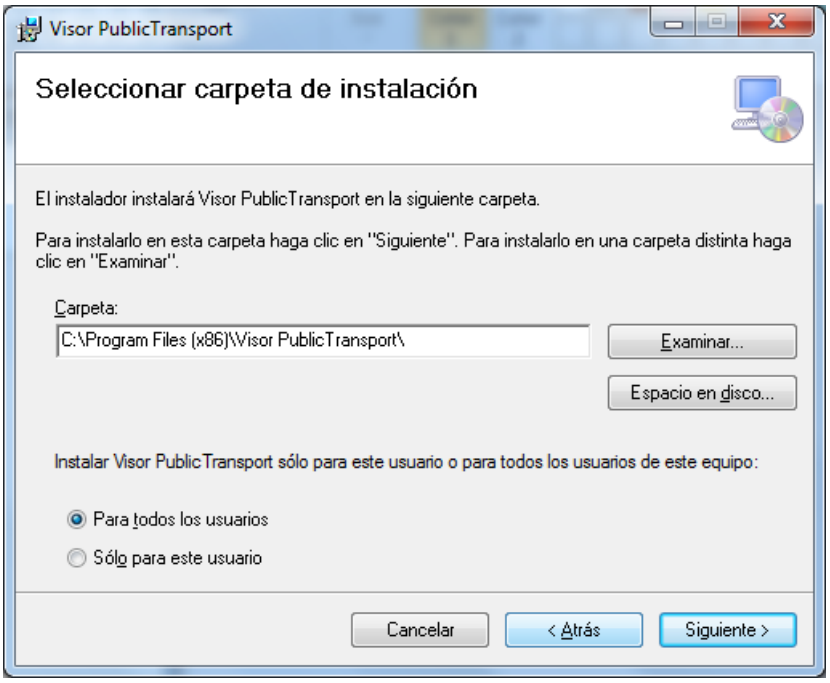

Figura 82: Destino de la instalación

4. Luego de seleccionar la carpeta de instalación, se suceden una serie de pantallas indicando el progreso de la misma. Finalmente si la instalación se da sin errores se obtiene la pantalla de instalación completada (Figura 83).

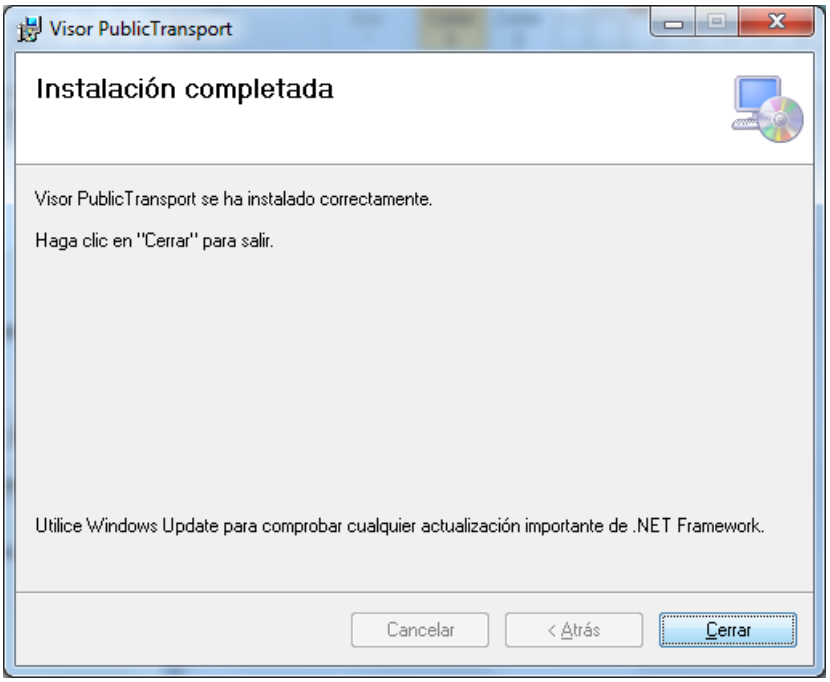

Figura 83: Instalación completada

Una vez finalizada la instalación es posible desinstalar la herramienta de Visualización. Una forma puede ser mediante “Agregar y quitar programas” desde el Panel de control de Windows o utilizando el archivo instalador, haciendo clic derecho sobre el mismo y eligiendo la opción “Desinstalar” como muestra la Figura 80.

Nota: Las opciones del menú emergente de la Figura 80 son las mismas en cualquier sistema operativo anfitrión, pero dependen del idioma en que se encuentra el mismo, en el ejemplo dado el sistema anfitrión está en Inglés y es por eso que las opciones del menú emergente se muestran en inglés.

2. Preparación de datos

La Herramienta de visualización está diseñada para ser usada como un gran caso de uso que involucra primero la generación de un modelo de red vial con *igoR-tp*, luego la simulación y generación de logs de simulación sobre ese modelo con el simulador *PublicTransport* y por último la visualización en la herramienta del modelo de red vial con alguna de las múltiples salidas del simulador (la salidas del simulador se tratan en la sección 4.6.2).

Evidentemente se tienen múltiples fuentes de datos, el modelo de red vial por un lado y los logs por otro, por esto es necesario que el usuario de la herramienta de visualización realice un trabajo previo de agrupación de datos en una misma carpeta y con cierta estructura.

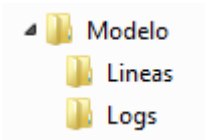

Figura 84: estructura de la entrada del visor

Se ilustrará como generar la estructura de la Figura 84 partiendo de un ejemplo ficticio.

2.1. Creación de estructura de carpetas de entrada

Como primer paso se debe crear la estructura de carpetas de la Figura 84. El nombre de la carpeta raíz puede ser cualquiera que el usuario desee, lo que es imperativo es que se respete el nombre de las carpetas internas */Líneas* y */Logs*.

2.2. Copia de datos desde el modelo de manipulación

En segunda instancia se debe copiar a estas carpetas los archivos de un modelo de manipulación y de logs. Supóngase que se dispone de un proyecto de manipulación en igoR-tp llamado *ModeloManipulacion* con al menos una solución manual de recorridos llamada *Solucion1*.

IgoR-tp persiste los proyectos en una carpeta interior al directorio de instalación (representado con “. “) llamada *Proyectos*, en esta carpeta se encuentra.

```
(1) .\igoR-tp\Proyectos\ModeloManipulacion
(2) .\igoR-tp\Proyectos\ModeloManipulacion\Soluciones\Manuales\Solucion1
```

Se deben copiar solo los archivos de (1) en la carpeta *Modelo* de la Figura 84 y todos los archivos de (2) en la carpeta */Líneas* de la Figura 84.

2.3. Copia de datos desde el simulador

Por último, se debe copiar dentro de la carpeta */Logs*, todos los archivos XML de log generados con el simulador sobre el modelo anterior.

3. Modo de uso

Una vez instalada la aplicación y preparados los datos de entrada, se está en condiciones de ejecutar la misma y visualizar una simulación.

3.1. Ejecución de la aplicación

El instalador de la aplicación crea un grupo de programas llamado *Visor PublicTransport* en el área del botón de inicio de Windows. Este grupo cuenta con un solo icono llamado *Visor* como muestra la Figura 85, el cual es el lanzador de la aplicación.

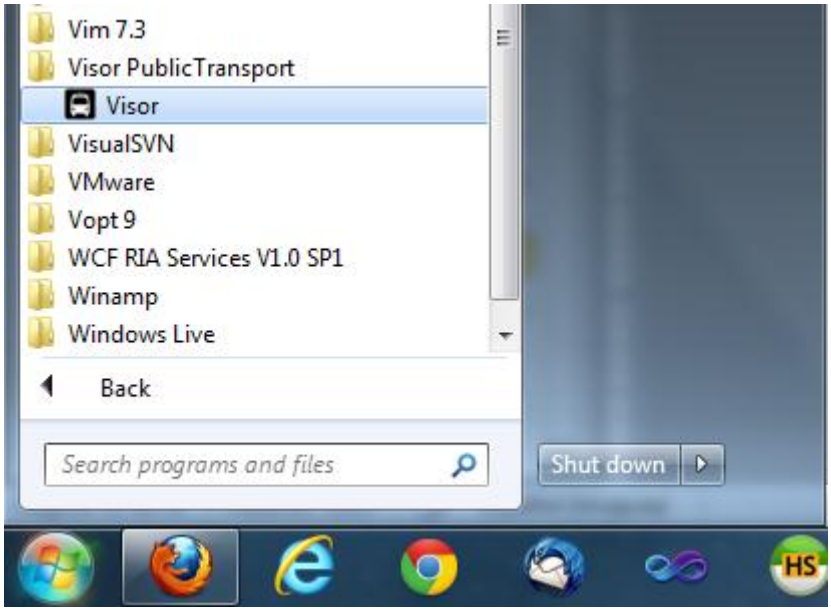

Figura 85: Lanzador de la aplicación

Una vez lanzada la aplicación se debe observar la pantalla de la Figura 86.

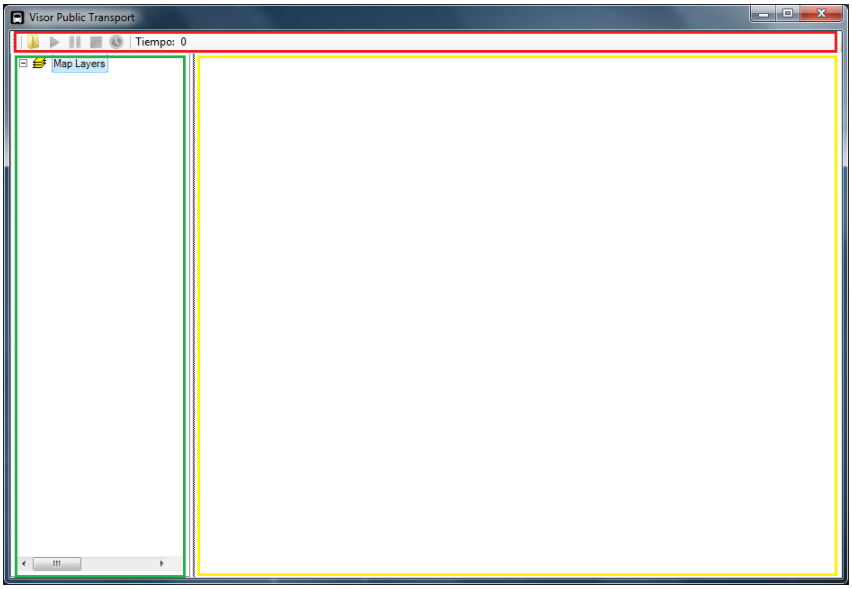

Figura 86: pantalla inicial de la herramienta

La Figura 86 muestra la separación lógica en zonas de la herramienta. La zona superior es el área de comandos de control, la zona izquierda es el área de control de capas que componen una solución y la zona derecha, que es la mayor de todas, es el área de mapa donde se visualiza la simulación.

3.2. Apertura de una simulación

Una vez que se ejecuta la aplicación es necesario cargar algún juego de datos para visualizar una simulación. Si los pasos detallados en la sección *Preparación de datos* se siguieron adecuadamente, es posible cargar un juego de datos como el de la Figura 87 (nótese que en dicho ejemplo llamado *Ciudad6* se encuentra la estructura de carpetas exigida en la Figura 84).

Pasos a seguir:

- Hacer clic en el comando ***Abrir*** (recuadro en la Figura 87).
- Navegar con el diálogo hasta la carpeta de datos preparados.
- Selecciona en el diálogo el archivo con extensión .manipulacion y clic en el botón ***Open***.

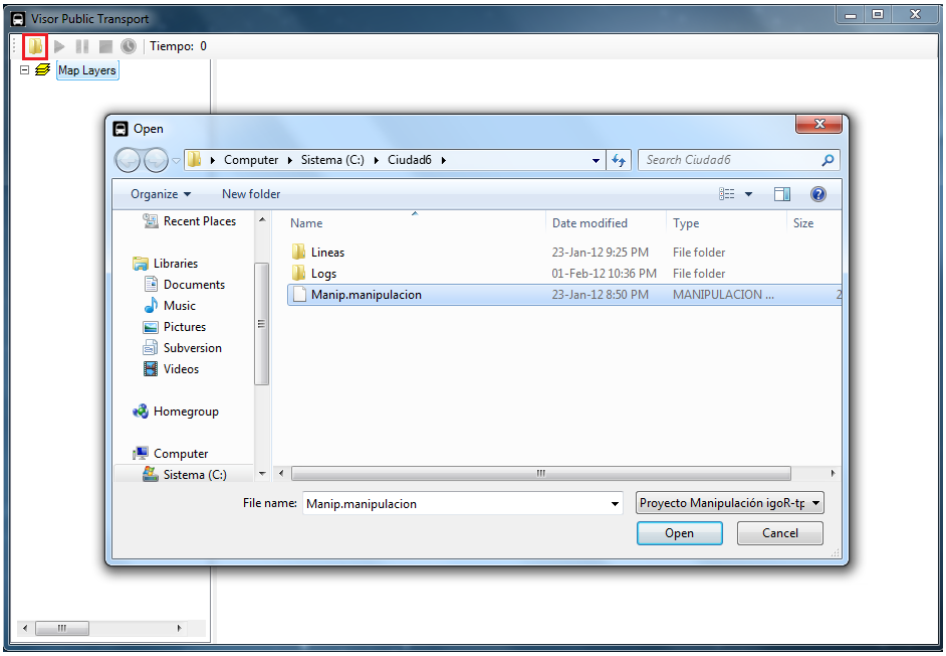

Figura 87: juego de datos de ejemplo

Si la apertura se realiza con éxito se debe observar una pantalla como muestra la Figura 88 , donde las áreas de control de capas y de mapa muestran los datos cargados y el área de comando de control se vuelve más compleja mostrando nuevos comandos para controlar el mapa y la simulación. Para iniciar la simulación basta con hacer clic en el comando ***Play*** (recuadrado en rojo en la Figura 88).

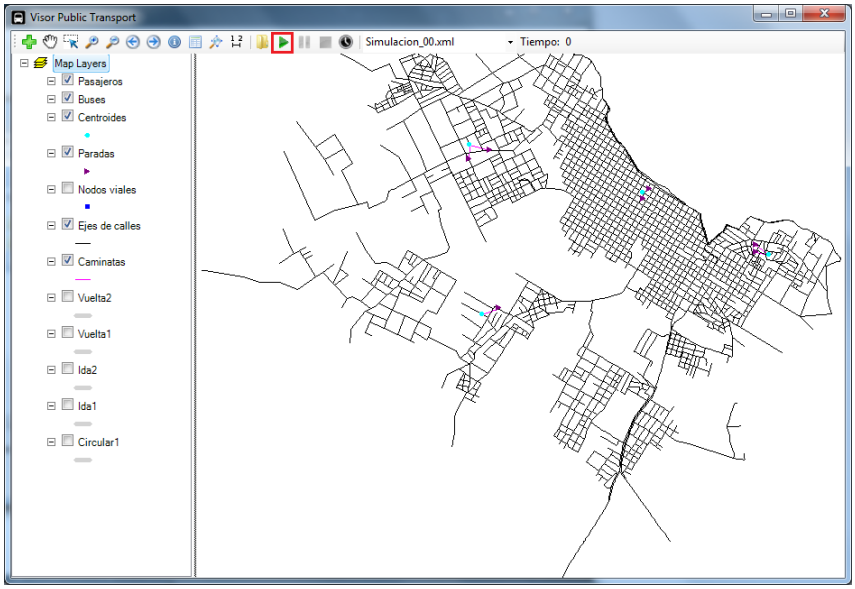

Figura 88: Simulación lista para ser ejecutada

3.3. Configuración de simulación

Se dispone del comando ***Configuración*** el cual no solo permite modificar la velocidad de la simulación sino que también permite seleccionar la forma de procesar los datos de log. La pantalla de configuración se invoca haciendo clic en el comando ***Configuración*** (recuadrado en rojo en la Figura 89).

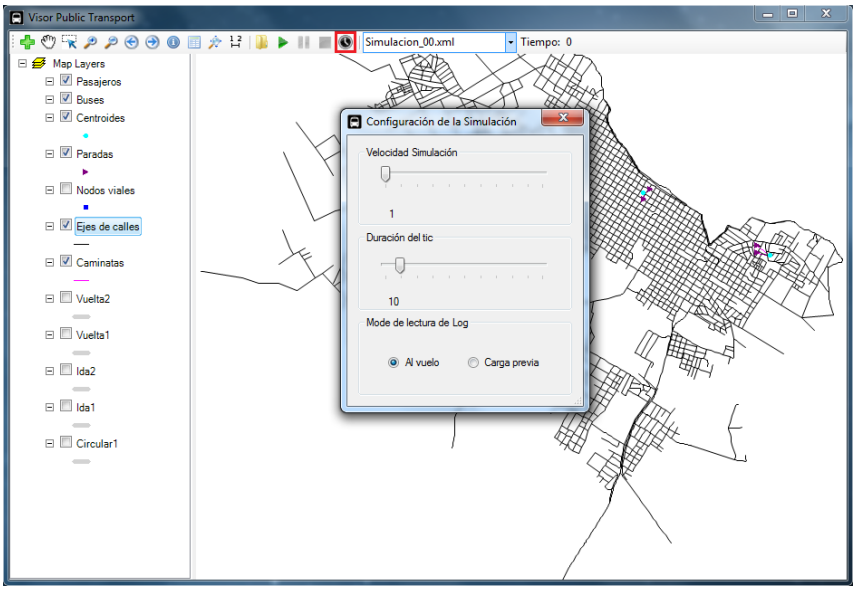

Figura 89: pantalla de configuración de simulación

La configuración de la simulación se divide en tres áreas:

- Área de **Velocidad de simulación** permite mediante un control de desplazamiento seleccionar un valor entre 1 y 100 segundos, dicho valor se explicó en el capítulo *Implementación* y se corresponde con el *Factor de Velocidad.*
- Área de ***Duración del tic*** permite establecer el intervalo de tiempo en milisegundos en el que la rutina del bucle principal de la simulación es invocada. Se sugiere no modificar este valor a no ser que se observen bloqueos en la animación. En algunos sistemas puede ser necesario ajustar este valor para dar tiempo a que se ejecute todo el bucle de simulación antes que se vuelva a invocar otra vez.
- Área de ***Modo de lectura de Log*** solo se encuentra activa si la simulación se cargó pero no se ejecutó en cualquier otro estado de la simulación se encuentra desactivada. Permite modificar la forma de lectura del archivo de log XML donde se almacena la simulación.
    - **Al Vuelo**: Opción por defecto que establece el modo de lectura del log a demanda. Este modo tiene la ventaja que utiliza poca memoria RAM, pues en cada iteración lee del log lo necesario hasta el tiempo actual de simulación pero por otro lado tiene la desventaja que accede muchas veces a disco. Las pruebas en sistemas Core Duo con discos SATA estándar resultan en animaciones fluidas sin interrupciones.
    - **Carga Previa**: Opción que establece la lectura y carga completa del log en memoria antes que se de inicio a la simulación. Permite ejecuciones fluidas y sin saltos en cualquier sistema siempre y cuando se cuente con suficiente RAM para cargar el log. En las pruebas usando simulaciones de 24 horas de duración con la geometría de la ciudad de Rivera el proceso del Visor no supero los 140MB de consumo de memoria.

3.4. Configuración de los elementos visuales

La herramienta establece por defecto las formas y colores de los elementos visuales que se despliegan en el área de mapa, si el usuario desea cambiar algunas de estas propiedades, es posible realizarlo mediante la opción ***Properties*** del menú contextual desplegado al hacer clic derecho sobre alguno de los elementos en el área de capas del mapa (Map Layers).

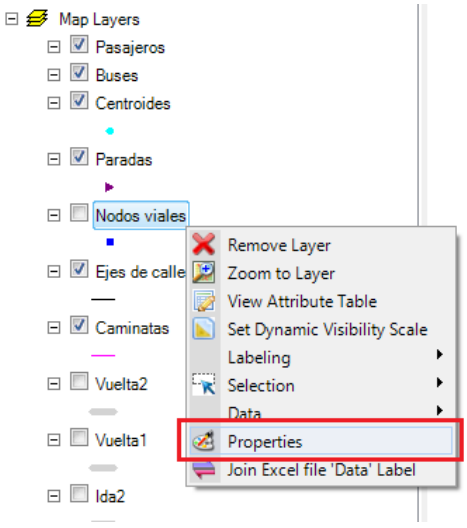

Figura 90: Configuración de los elementos visuales

El diálogo que se despliega es sensible al contexto, es decir, no es lo mismo desplegar las propiedades de un elemento polilínea (color de línea, espesor de línea, tipo de línea, etc.) que desplegar las propiedades de un elemento punto (forma de elemento punto y color).

En la Figura 90 se muestra un ejemplo donde se tratan de establecer las propiedades de una capa de puntos (los Nodos Viales), en este caso el diálogo desplegado es similar a la Figura 91. Este diálogo permite fácilmente cambiar la forma, el color y la transparencia de manera que el usuario pueda adaptar la salida visual en el mapa a la configuración que le resulte más amigable.

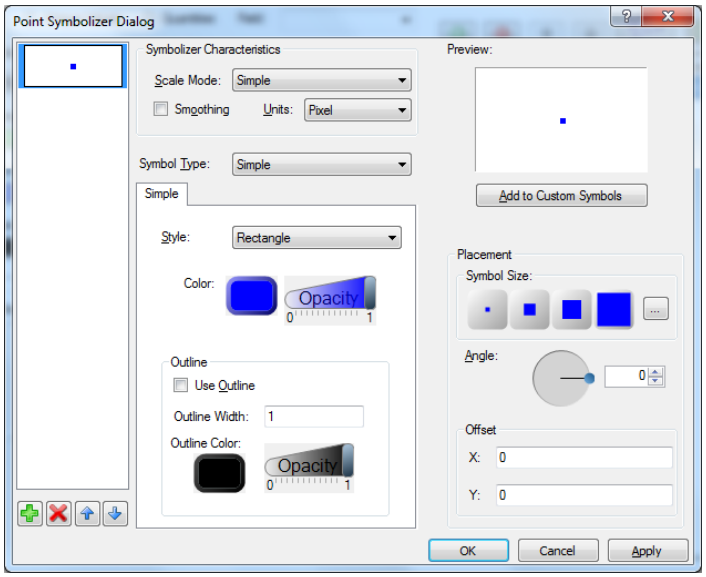

Figura 91: diálogo específico de una capa de puntos

3.5. Cartilla de comandos de control

	Añadir capa de fondo desde un archivo Shapefile
	Movimiento libre de mapa mediante arrastrar y mover, incluso durante una simulación activa
	Selección de elemento de mapa
	Acercamiento de mapa (Zoom in), se recomienda realizarlo con la simulación en pausa
	Alejamiento de mapa (Zoom out), se recomienda realizarlo con la simulación en pausa
	Estado previo de mapa (los comando sobre el mapa se almacena en una pila)
	Estado siguiente de mapa
	Información de un elemento
	Muestra la tabla de datos adjuntos a la capa activa
	Permite ajustar todo el mapa a la pantalla
	Activa la herramienta para medición de longitudes sobre los elementos del mapa
	Abrir un nuevo juego de datos y volver al estado inicial (tiempo de simulación a cero)
	Iniciar o continuar una simulación
	Pausar una simulación en marcha
	Detener una simulación en marcha y volver al estado inicial (tiempo de simulación a cero)
	Acceder al diálogo de configuración de simulación (ver sección ***Configuración de simulación***)

Apéndice D – Investigación de tecnologías

1. Relevamiento de aplicaciones de simulación

Antes de empezar a definir la tercera parte de este proyecto (la herramienta de visualización), se realiza un estudio para ver que ofrece el mercado actualmente en materia de herramientas para simulación. En particular interesaba conocer herramientas de simulación de transporte público y ver cómo dichas herramientas manejan la salida visual.

El principal propósito de este estudio es determinar si es una práctica habitual modelar la red en una herramienta, luego correr la simulación y por último visualizar el resultado de la misma en otra, o si esto ya ha quedado en desuso. Además en la medida de lo posible observarla calidad de las salidas gráficas de las herramientas comerciales para construir nuestra herramienta no muy alejada de esa realidad.

Para realizar este estudio se parte de la encuesta del año 2003 realizada por la publicación OR/MS Today [48] donde se evalúan una cantidad significativa de herramientas para simulación. Luego de publicada una nueva versión de dicha encuesta en Octubre 2011 [49] , se continúa el presente estudio basándose en las herramientas que aparecen en ella.

A continuación se presenta un extracto del artículo que acompaña dicha encuesta [50] donde se describen algunas cualidades y variedad de herramientas de simulación que podemos encontrar hoy en día.

> “Las herramientas de software de simulación de la última encuesta son impresionantes en su gama, sofisticación y capacidades. Como grupo ofrecen apoyo incomparable para la construcción, visualización y análisis de modelos de una amplia variedad de aplicaciones. Muchas de ellas incluyen la capacidad para comunicarse con otras aplicaciones o para aprovechar las rutinas de optimización para ajustar el personal u otros recursos de simulación para reducir el costo, riesgo o retraso con el cliente. Representan la culminación de más de medio siglo de convergencia de desarrollo de software, hardware y los esfuerzos de investigación de simulación. “

> “Los productos de simulación de hoy en día representan muchos avances en las capacidades para la construcción, estudio y análisis de modelos de simulación. La evolución de los productos de simulación se entrelaza con la evolución de la teoría de la simulación y la práctica, así como con los avances en hardware y software. Las limitaciones de una generación a menudo han sido la base de las herramientas de la próxima generación, como la capacidad de evolución para abordar los problemas. “

De las cincuenta y cinco herramientas evaluadas en la encuesta, seleccionamos las que se sugerían más adecuadas para su aplicación a la industria del transporte.

1. Enterprise Dynamics – ED Transport
2. ExtendSim Suite
3. Micro Saint Sharp
4. SAS Simulation Studio
5. ServiceModel Optimization Suite

ED Transport

INCONTROL Simulation Solutions [50], en colaboración con el Instituto de Logística de Transporte (ITL -Institute of Transport Logistics) construyó una plataforma de simulación para el transporte y la logística, *Enterprise Dynamics Transport*, con dos bibliotecas independientes: *ED TransSim-Node* y *ED TransSim-Net*.

ED TransSim-Node se puede utilizar para el modelado de diferentes tipos de procesos y cadenas de transporte dentro de un depósito, por ejemplo maniobras de ordenación de la zona, transfiriendo las cajas móviles de los carriles hacia camiones o remolques, gestión individual así como la carga y descarga de todo el proceso. El átomo Yard Management gestiona el encaminamiento de todos los vehículos de acuerdo a su función en la cadena del proceso. *TransSim-Node* tiene la capacidad de modelar las redes de montacargas, carga y descarga de buffers, puertas, rampas, zonas de almacenamiento para nombrar solo unos pocos objetos. Por otra parte, un apartadero de ferrocarril con las grúas se incluye para ofrecer el modelado de los nodos logísticos diferentes, incluidos los terminales de tráfico bimodal.

Esta herramienta está pensada para el transporte de mercaderías y no para el problema del transporte público, por este motivo no se profundizó su estudio. Sin embargo en el sitio web de la empresa se pudieron observar varias capturas de pantalla de la interfaz gráfica lo que igualmente sirvió para empezar a tener una idea de los estándares actuales. Lo más relevante en este punto es que ofrece tanto visualización en 2D como en 3D.

ExtendSim Suite

ExtendSim [51] es una herramienta de simulación de procesos de gran alcance.

Con *ExtendSim* ofrece las siguientes funcionalidades:

- Predecir el curso y los resultados de determinadas acciones.
- Visualizar los procesos de logística en un entorno virtual.
- Identificar las áreas problemáticas antes de la aplicación de un cambio.
- Explorar los posibles efectos de las modificaciones.
- Confirmar que todas las variables son conocidas.
- Optimizar las operaciones.
- Evaluar ideas e identificar las ineficiencias.
- Entender por qué ocurren los fenómenos observados.
- Comunicar la integridad y la viabilidad de sus planes.

La característica más relevante de esta herramienta es que los modelos de simulación en *ExtendSim* se representan mediante el uso de bloques de proceso, en donde cada uno de los bloques tiene como función describir una actividad propia del proceso.

A continuación se muestran dos ejemplos sencillos de simulaciones realizadas con *ExtendSim* [52]. La primera imagen (Figura 92) muestra la definición del modelo de simulación para la atención de una cola de clientes por dos cajeros. La segunda (Figura 93) muestra la definición del modelo de simulación para un lavadero de autos.

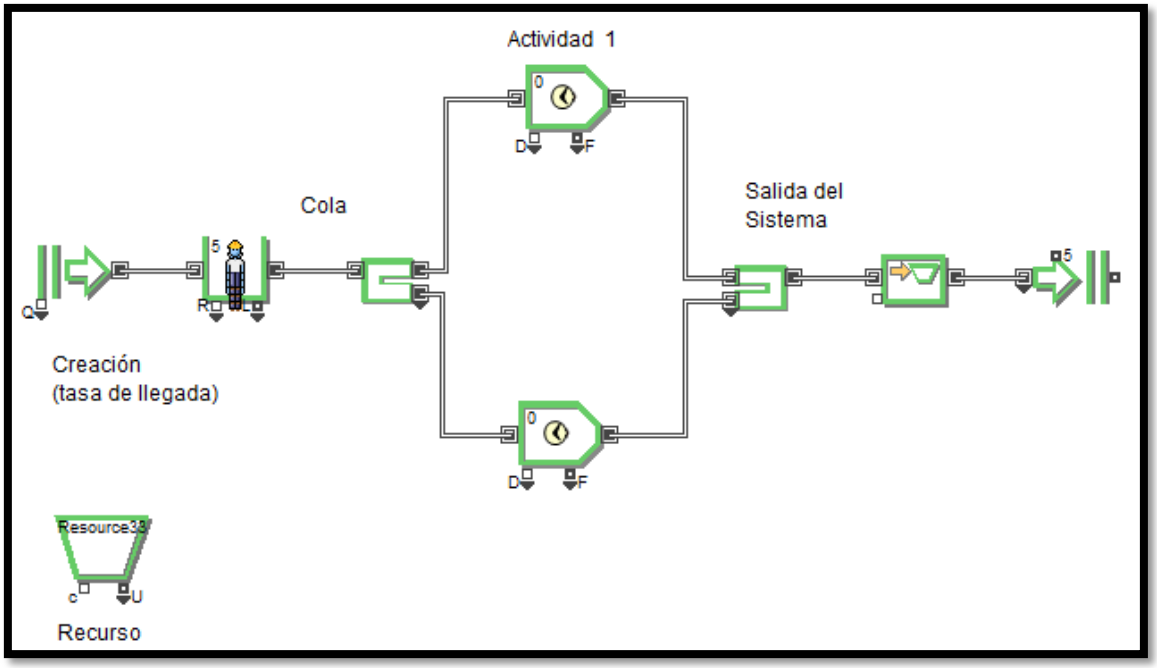

Figura 92: Simulación con Extensim

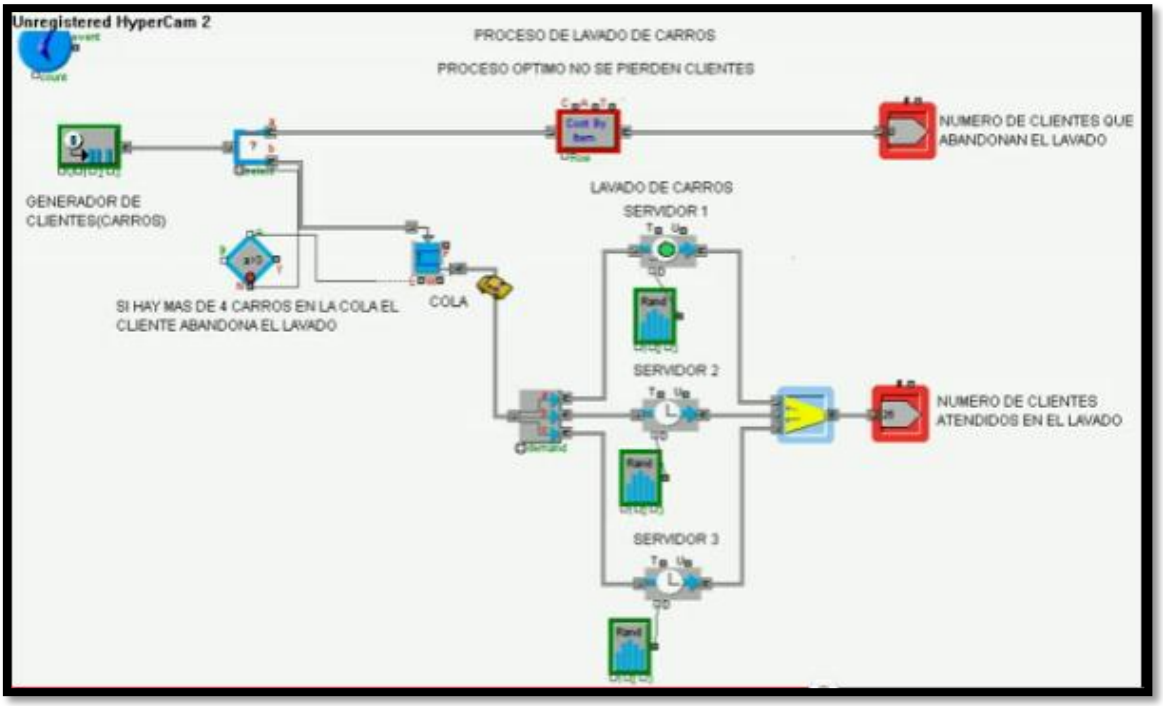

Figura 93: Proceso de lavado de autos

Micro Saint Sharp

Micro Saint Sharp [53] es una herramienta de simulación de eventos discretos que permite crear modelos de simulación. Cualquier proceso que puede ser representado por un diagrama de flujo puede ser modelado utilizando *Micro Saint Sharp.* Está diseñado para ser flexible para su uso en una variedad de aplicaciones.

Micro Saint Sharp ofrece tres maneras de animar el modelo. La primera manera es ejecutar el modelo mediante un diagrama de red, la segunda es elegir *Animator* que muestra animaciones del proceso y la tercera es tener un entorno virtual que representa el proceso deseado usando *Animator3D.*

A continuación se muestran dos ejemplos de la salida visual, la primera (Figura 94) es una animación 2D usando *Animator* de una simulación correspondiente al servicio de cajas de un banco. En la captura de pantalla se puede ver como los clientes van llegando al banco y formando las colas correspondientes. La segunda (Figura 95) muestra el uso de *Animator3D* en la simulación de una sala de emergencia de un hospital.

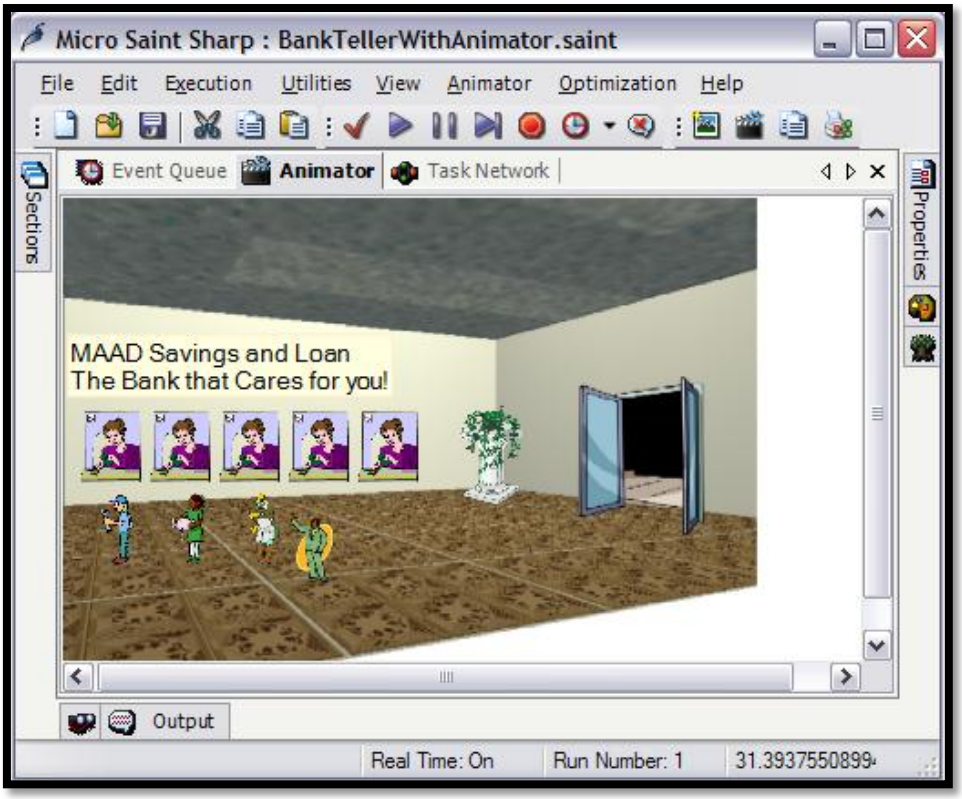

Figura 94: Micro San Sharp Salida visual Animación

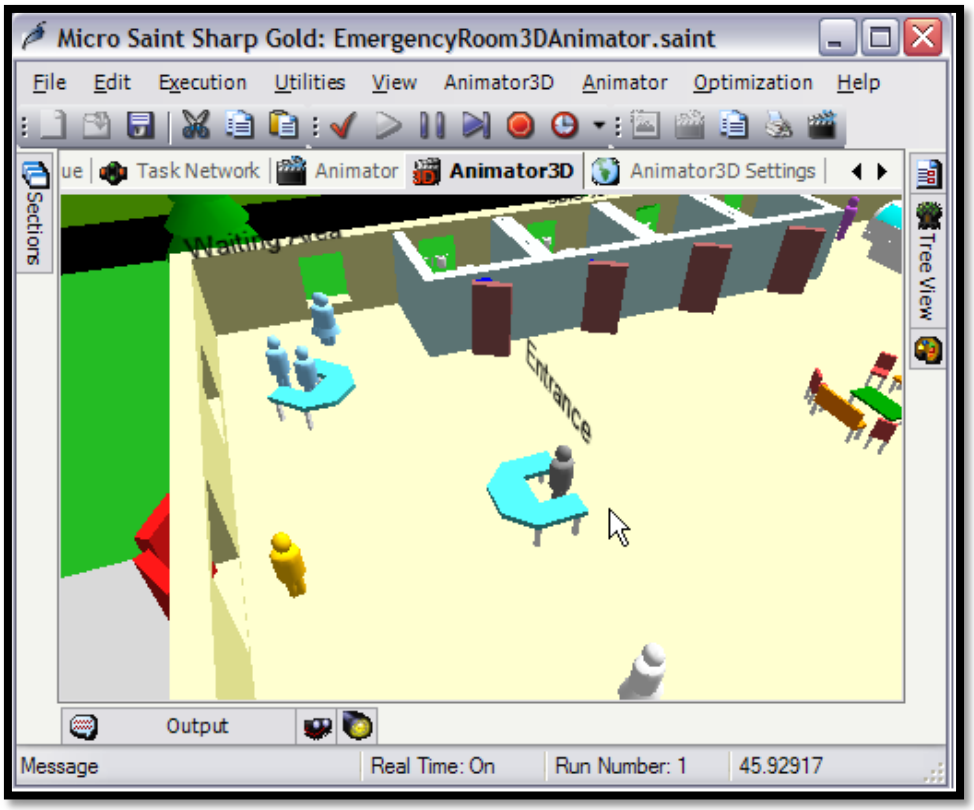

Figura 95: Micro San Sharp Salida visual 3D

SAS Simulation Studio

SAS Simulation Studio [54] es una aplicación que utiliza la simulación de eventos discretos para modelar y analizar sistemas. Se desarrolló en el lenguaje de programación Java y proporciona las siguientes interfaces de usuario:

- una interfaz gráfica de usuario que no requiere programación y proporciona todas las herramientas para la construcción, ejecución y análisis modelos de simulación a eventos discretos.
- una interfaz de programación que le permite ejecutar modelos en procesamiento por lotes.

Este software está diseñado para interactuar tanto con otras herramientas SAS para que pueda llevar a cabo sofisticados análisis estadísticos de los resultados. Los datos generados por el modelo se pueden guardar y analizar posteriormente.

Además la herramienta incluye una ventana de experimentos que permite analizar como los cambios de distintos parámetros de la simulación afectan la salida del modelo de simulación. La ventana de experimentos permite la realización de análisis de sensibilidad, comparar dos o más sistemas SAS o utilizar un diseño experimental.

El manual consultado es detallado en el uso de la herramienta para construir modelos de simulación y presenta varios ejemplos de modelos de simulación realizados usando diagramas de bloques. Sin embargo no se encontró aquí ni en otra fuente un ejemplo de la salida gráfica de una simulación con *SAS Simulation Studio*.

A continuación se presenta un ejemplo de un modelo de simulación creado mediante bloques con esta herramienta (Figura 96). Este modelo representa el proceso de un centro de reparaciones.

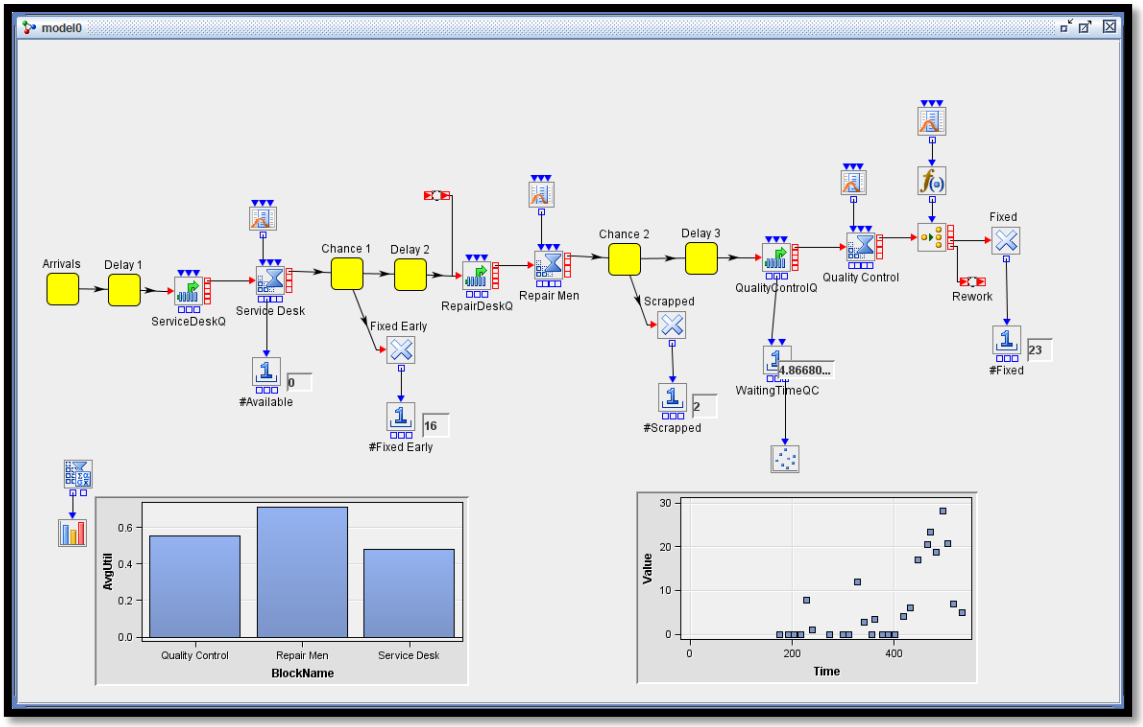

Figura 96: Simulación con Simulation Studio

ServiceModel Optimization Suite

ServiceModel [55] es un simulador para procesos de negocio y de servicios. En este software, al igual que en los mencionados anteriormente, la construcción del modelo de simulación se realiza en forma gráfica por medio de bloques. La salida visual brinda animaciones tanto en 2D como en 3D.

ServiceModel tiene varias características a destacar:

- Permite importar o crear nuevos íconos para la biblioteca de gráficos.
- Integración a Excel, Lotus, Visual Basic y herramientas principales de Microsoft.
- Facilidad de importar y simular sobre un archivo de AutoCAD.
- Tiene dos módulos de optimización disponibles: *SimRunner* y *OptQuest*
    - *SimRunner* [56]*:* Software de Optimización basado en Algoritmos Genéticos y Programación Evolutiva. Partiendo de una Función Objetivo (Z) definida por el usuario y las restricciones automáticamente crea los escenarios posibles de la simulación y los ejecuta en *ServiceModel* obteniendo el resultado óptimo.
    - *OptQuest* [57]*:* es un módulo de optimización diseñado para facilitar su integración en aplicaciones que requieren de la optimización de sistemas de alta complejidad, ya sea que éstos utilicen simulación o no. Usa metaheurísticas, optimización matemática, y redes neurales para realizar búsquedas eficientes de las mejores soluciones a todo tipo de problemas de decisión y planificación.
- Tiene un módulo para ajuste de curvas: *Stat:Fit* [58] ayuda a encontrar la mejor distribución para representar los datos.

En las siguientes figuras se muestran las dos opciones de salida gráfica. Dichas imágenes presentan una vista 2D (Figura 97) y una vista 3D (Figura 98) de una simulación de un banco. Se puede apreciar como los autos estacionan en el estacionamiento del banco, los clientes bajan de los mismos y entran al banco donde se dirigen a los distintos escritorios o cajas para ser atendidos.

Figura 97: Simulación 2D de un banco con ServiceModel

Figura 98: Simulación 3D de un banco con ServiceModel

Legion Studio

Legion Studio [59] simula y analiza los movimientos paso a paso de los peatones dentro de un espacio definido. Esta aplicación no figura en la encuesta pero resultaba interesante incluirla en la evaluación ya que cuenta con módulos independientes para la construcción de modelos de simulación y la ejecución de los mismos. Los distintos módulos de *Legion Studio* son:

- Módulo construcción: dado un conjunto de insumos clave se puede construir un modelo preciso del espacio que se desea simular y analizar.
- Módulo simulación: partiendo del modelo creado en el módulo construcción se pasa al módulo de simulación. Este módulo permite grabar las corridas de las simulaciones como archivos de video con extensión ".avi".
- Módulo Análisis: automatiza los análisis comunes, proporcionando información detallada sobre cada uno de los peatones.

La Figura 99 muestra una captura de pantalla de la ejecución de una simulación con *Legion Studio*. Entre otras cosas se pueden observar las distintas opciones de menú disponible como zoom in, zoom out, mover, barra de progreso de tiempo, controles para pausar, parar o continuar la simulación.

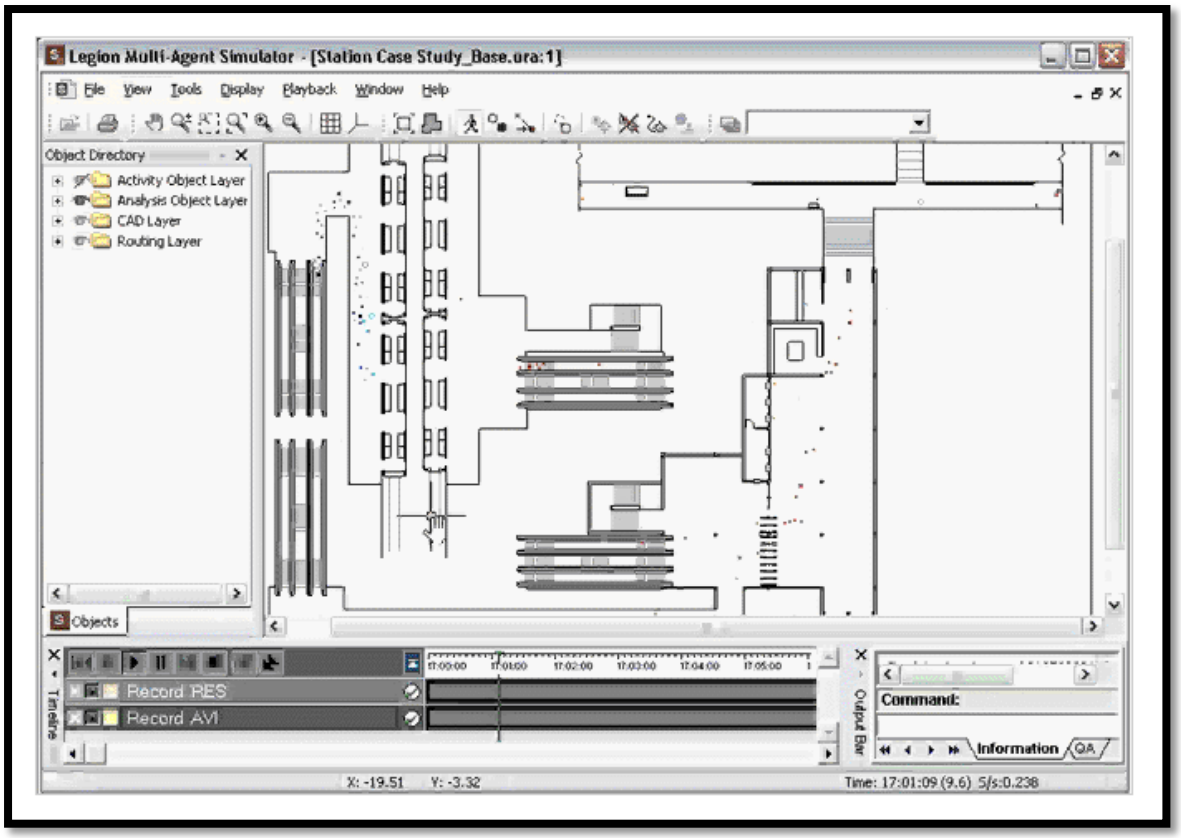

Figura 99: Legion Studio – Módulo simulación

Como se ha visto existen un sin fin de herramientas para simulación, sin embargo no es tan fácil encontrar ejemplos de su uso específico para simulación del transporte público. A continuación se muestran otras dos herramientas encontradas que tampoco figuran en la encuesta pero son específicas para este problema.

VISUM - *VISSIM*

VISUM [60] es un software para la planificación de transporte, modelos de demanda de viajes y gestión de redes. Se puede utilizar para la planificación del transporte a nivel metropolitano, regional, estatal y nacional. Está diseñado para el análisis multimodal, ya que integra todos los modos relevantes de transporte (automóviles, microbuses, camiones, autobuses, trenes, peatones y ciclistas) en un modelo de red consistente. Ofrece una variedad de procedimientos para la asignación de la demanda.

VISUM se integra con *VISSIM* [61] para combinar el análisis macroscópico de *VISUM* con la simulación del tráfico microscópico *VISSIM*. Un modelo abstracto de red (ANM) se crea a partir de una red VISUM y luego puede ser importado y editado en *VISSIM*.

VISSIM es una aplicación especialmente orientada a la simulación y el modelado de sistemas dinámicos complejos. El programa combina un entorno de diseño de modelos orientado a bloques para la definición del sistema del proceso a simular con un potente motor de simulación. La interfaz visual de *VISSIM* ofrece al usuario un método simple para desarrollar y construir, modificar y mantener modelos complejos; el motor de simulación proporciona soluciones para diseños de sistemas lineales, no lineales, sistemas continuos, discretos e híbridos. Si los requerimientos del diseño se extienden más allá de lo contemplado en la biblioteca de bloques de *VISSIM*, se pueden incluir bloques de usuario escritos en C, Fortran o Pascal.

Las siguientes figuras muestran dos ejemplos de simulaciones realizadas con *VISSIM*. En la Figura 100 se puede observar la simulación del tráfico en una rotonda, su finalidad es analizar el comportamiento del esquema de prioridades definido para dicha rotonda.

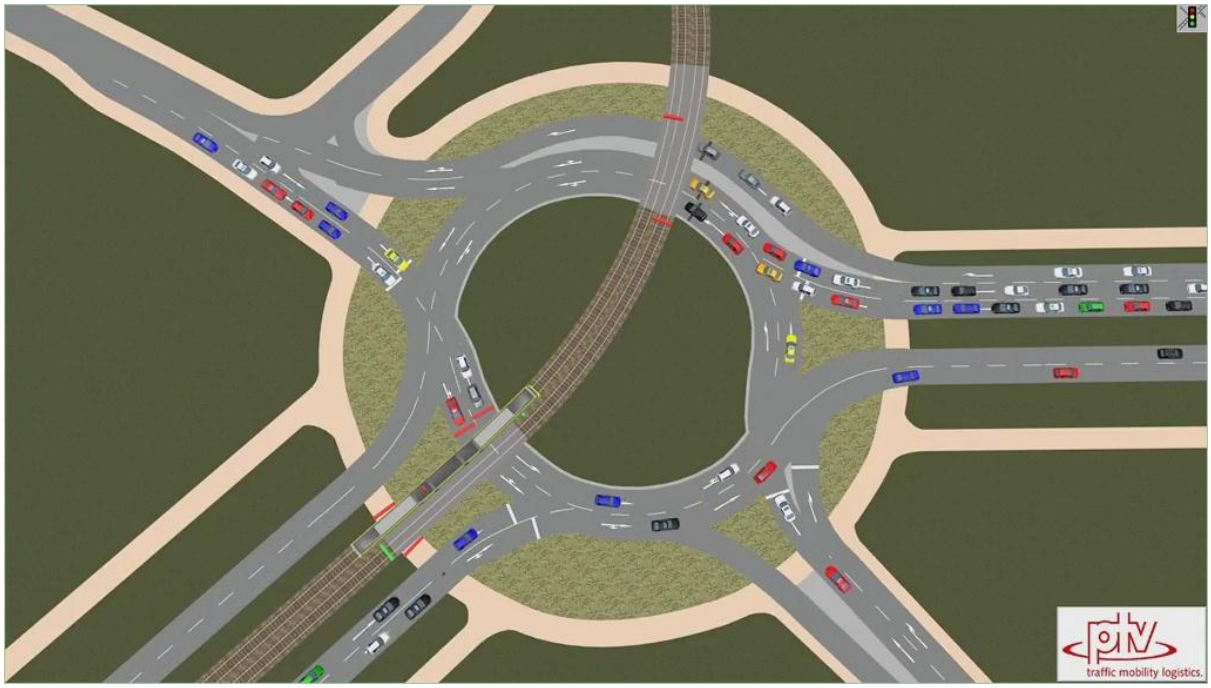

Figura 100: VisSim – Simulación de esquema de prioridades en una rotonda

La Figura 101 muestra la simulación 3D del tránsito de pasajeros en una terminal de tren. Se puede observar como los pasajeros se desplazan desde y hacia las paradas.

Figura 101: VisSim- Simulación de tránsito de pasajeros

S-Paramics microsimulation

S-Paramics [62] simula los componentes individuales del flujo de tráfico y la congestión, presenta su salida como una pantalla de visualización en tiempo real para la gestión del tráfico y el diseño de la red de carreteras. *S-Paramics* representa las acciones e interacciones de los vehículos individuales a medida que viajan a través de una red de carreteras. Se puede modelar la distribución detallada de la red física, e incluye características tales como las operaciones de autobuses, la configuración de las señales de tráfico, las características de comportamiento de los conductores y la cinemática de los vehículos.

A continuación se muestran varios ejemplos de la salida visual en un caso de estudio correspondiente a la ciudad de Londres. Allí se puede observar como: los buses paran en las paradas establecidas (Figura 103), se puede evaluar en detalle la interacción de los buses con otros vehículos (Figura 104), sirve para determinar la mejor ubicación de las paradas (Figura 102).

Figura 103: S-Paramics Londres Public Transport

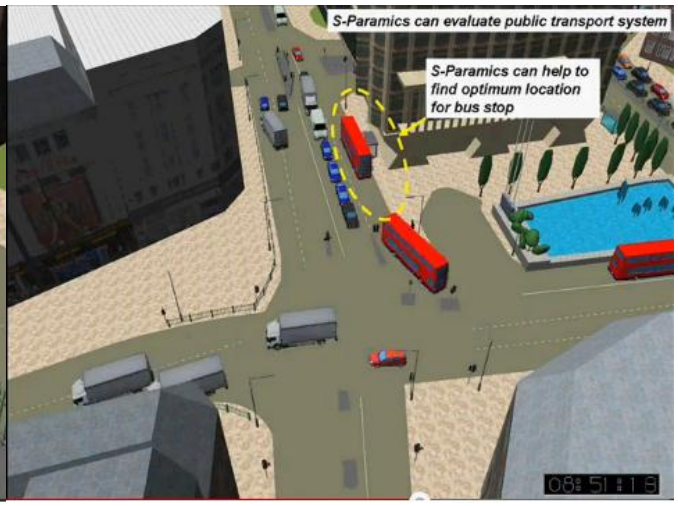

Figura 102: S-Paramics Londres Public Transport

Figura 104: S-Paramics Londres Public Transport

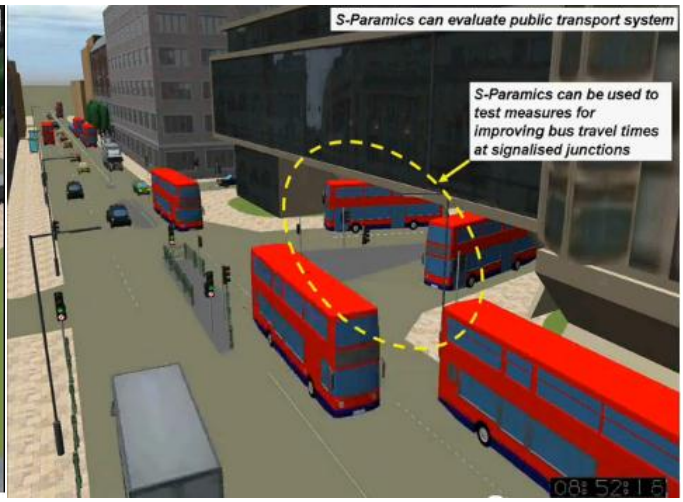

Figura 105: S-Paramics Londres Public Transport

2. Tecnologías para el desarrollo de sistemas de información geográfica

El siguiente estudio se realizó con el fin de seleccionar la tecnología con la cual implementar el nuevo visor del simulador.

MapWindow

MapWindow [63] es un proyecto de código abierto (MPL 1.1 [64]) que facilita la programación de sistemas de información geográfica para el sistema operativo *Windows*. Soporta la manipulación, el análisis y la visualización de datos geoespaciales junto a sus datos asociados en varios formatos estándar.

MapWindowsGIS.ocx consta de un control *ActiveX* [65] gratuito y libre desarrollado en *C++* para la programación de sistemas de información geográfica y una Interfaz de Programación de Aplicaciones (API). El control MapWindow se puede agregar a un formulario desarrollado en *Visual Basic, C#* o en cualquier otro lenguaje de programación que soporte *ActiveX*. Además de ser fácil de incorporar dentro la familia de productos *MS-Office*. La API permite el acceso a objetos, funciones, propiedades y métodos relativos a la visualización y manipulación básica de información geográfica en formato *Shapefile, Grid* y *TIN.*

ArcGIS

ArcGIS [66] es un sistema de información geográfica on-line para utilizar mapas e información geográfica. Consta de un variado conjunto de clientes (clientes de servicios), diseñados para distintas plataformas: Web, plataformas móviles y escritorios de estación de trabajo. *ArcGIS* incluye también un conjunto de servidores que exponen los servicios SIG. Estos servicios se pueden alojar en una variedad de configuraciones:

- En un equipo local (y como archivos en disco).
- Publicados como servicios web SIG.
- Publicados y compartidos a través de internet (computación en la nube).

En función del cliente que se utilice *ArcGIS* admite varias API de desarrollo para aplicaciones SIG. En particular en este estudio se analizaron algunas de las API para aplicaciones web.

ArcGIS API para Silverlight

La *ArcGIS* API para *Silverlight* [67] permite crear aplicaciones web o de escritorio que utilizan las capacidades de mapa, geocodificación y geoprocesamiento que proveen los servidores *ArcGIS*. La API está construida sobre la plataforma *Microsoft Silverlight* la que se integra con *Visual Studio 2010* y *Expression Blend 4*. Esta API es de uso gratuito.

Para crear un mapa usando la ArcGis API para *Silverlight* se debe:

- Agregar la referencia a la biblioteca ESRI.ArcGIS.Client.dll.
- Añadir una referencia al ensamblado ESRI.ArcGIS.Client en el código Silverlight.
- Agregar un control de mapa a un contenedor incluido en la página.
- Agregar un componente para visualizar mapas en forma de mosaicos. Dicho componente consume mapas ofrecidos por un servidor ArcGIS Server en forma de colección de capas.

- Agregar las capas que se necesiten al mapa. Estas capas son las que contendrán las geometrías (punto, polígono, línea).

Servicios de ArcGIS Server

ArcGIS Server es el principal producto de ESRI para sistemas de información geográfica, permite crear y distribuir web services geoespaciales. ArcGIS Server ofrece servicios para la gestión de datos espaciales, la visualización y análisis espacial.

Si bien las APIs de desarrollo son de uso gratuito, la instalación de un *ArcGIS* Server no lo es. Sin embargo, se puede utilizar la API para *Silverlight* sin necesidad de instalar un *ArcGIS* Server propio ya que ESRI brinda servicios de mapas, imágenes online y múltiples servicios geoespaciales de forma gratuita.

Algunos servicios de mapas de uso gratuito brindados por ESRI [68]:

- Mapa mundial de imágenes satelitales: este servicio de mapas presenta imágenes satelitales de baja resolución para todo el mundo e imágenes de alta resolución para Estados Unidos, Gran Bretaña y alguna otra región.
- Mapa mundial de calles: este mapa de calles presenta datos a nivel de carreteras y autopistas en todo el mundo, además de un mapa detallado de calles para Estados Unidos, gran parte de Canadá, Japón, la mayoría de los países de Europa, partes de Asia, Australia y Nueva Zelanda, partes de América del Sur, incluyendo Argentina, Brasil, Chile, Colombia y Venezuela, y partes de África del Sur entre ellos Botswana, Lesotho, Namibia, Sudáfrica y Suazilandia. No incluye datos a nivel de calle para Uruguay.
- Mapa mundial topográfico: este servicio de mapas está diseñado para ser utilizado como un mapa base y como un mapa de referencia. El servicio de mapas incluye límites administrativos, ciudades, cursos de agua, características fisiográficas, parques, monumentos, autopistas, carreteras, ferrocarriles y aeropuertos.
- Mapa mundial de relieve: este servicio de mapas muestra la elevación de la superficie en relieve sombreado. La resolución del mapa (tamaño de la celda) es la siguiente: 30 metros para los Estados Unidos 90 metros para todas las zonas terrestres entre los 60 ° norte y 56 ° de latitud sur. 1 km de resolución superior a 60 º norte y 56 ° sur.
- Mapa mundial de *National Geographic*: este mapa fue desarrollado por *National Geographic* y Esri y refleja el distintivo estilo cartográfico de *National Geographic* en un mapa multi-escala del mundo. Este mapa de referencia incluye los límites administrativos, ciudades, áreas protegidas, carreteras, caminos, vías férreas, cauces de agua, edificios y monumentos históricos. El mapa incluye la cobertura global hasta la escala 1:144 k ~ y una cobertura más detallada para América del Norte hasta ~ 01:09 escala k.

Servicios de imágenes de uso gratuitos [69]:

Este conjunto de servicios de imágenes dinámicas se basa en los conjuntos de datos creados por el Servicio Geológico de EE.UU. (USGS) y la National Aeronautics and Space Administration (NASA), utilizando imágenes Landsat. [4]

Algunos ejemplos son:

- Penetración atmosférica (754) 1990-2005: Esta capa contiene las imágenes mundiales desde 1990 hasta 2005. La combinación de bandas 745 no incluye bandas visibles, lo que le otorga la mejor penetración atmosférica. Costas y riberas están bien definidas. Puede ser utilizada para buscar texturas y humedades de los suelos. La vegetación aparece azul. Esta combinación es particularmente útil para estudios geológicos.
- Análisis de vegetación (543) 1990-2005: Esta combinación de bandas proporciona al usuario una gran cantidad de información y contraste de color. La vegetación sana es de color verde brillante y los suelos son de color malva. Es útil para los estudios de vegetación y es ampliamente utilizado en el sector maderero y el control de pestes.

Servicios geoespaciales de uso gratuito:

La ArcGIS Server REST API [70] proporciona una interfaz Web abierta para los servicios SIG alojados por ArcGIS Server. Se puede acceder a todos los recursos y operaciones expuestos por el API REST a través de una dirección URL para cada servicio SIG publicado con ArcGIS Server.

La API REST soporta cuatro tipos de geometrías: puntos, polilíneas, polígonos y envelopes.

En particular ESRI tiene publicados los siguientes servicios de uso libre:

Servicios de geocodificación

Geocodificación es el proceso de asignación de un lugar, generalmente en forma de valores de coordenadas (puntos), a una dirección mediante la comparación de los elementos descriptivos de localización en la dirección a los presentes en el material de referencia. Las direcciones vienen en muchas formas, que van desde el formato de dirección común de un número de la casa seguido por el nombre de la calle hasta descripciones de zona como ser el código postal. En esta categoría se encuentran:

- Encontrar direcciones candidatas: Devuelve una lista de candidatos basándose en una dirección y ubicación.
- Geocodificación inversa: Devuelve información sobre todos los campos de dirección relacionadas con la dirección inversa geocodificada, así como su ubicación exacta.

Servicios de geoprocesamiento

El geoprocesamiento es una parte fundamental de las operaciones de SIG, ofrece el análisis de datos, gestión de datos y herramientas de conversión de datos necesarios para todos los usuarios de SIG. Un servicio de geoprocesamiento representa una colección de herramientas publicadas que realizan las tareas necesarias para la manipulación y análisis de información geográfica a través de una amplia gama de disciplinas.

[4] LandSat: serie de satélites construidos y puestos en órbita por EE. UU. para la observación en alta resolución de la superficie terrestre

El servicio de geoprocesamiento de la REST API proporciona la información básica asociada con el servicio, tal como la descripción del servicio, las tareas previstas, el tipo de ejecución, y el nombre del resultado del servidor de mapas. Las operaciones soportadas son:

- Execute task: Se utiliza cuando el tipo de ejecución es sincrónico. Cuando una tarea se ejecuta de forma sincrónica, el usuario debe esperar los resultados.
- Submit job: Se utiliza cuando el tipo de ejecución es asíncrona.

Servicios geométricos

El servicio geométrico de la REST API es principalmente un recurso de procesamiento algorítmico que apoya las operaciones relacionadas con las geometrías. Tiene las siguientes operaciones: proyección, simplificación, buffer, área, longitud, diferencia, generalización, densificar, intersección, offset, unión, y algunas otras.

Servicios de imágenes

El servicio de imagines de la REST API representa un servicio de imágenes publicado con el ArcGIS Server. El recurso ofrece información básica asociada con el servicio de imágenes, tales como la descripción del servicio, su nombre, descripción, extensión, tamaño de píxel, y cuenta de bandas.

Las operaciones disponibles son: download, export, identify y query.

Servicios de análisis de red

Realizan análisis de redes de transporte como enrutamiento, instalación más cercana y área de servicio.

Servicios de entidades (Feature services)

Los servicios de entidades permiten servir entidades en Internet y proporcionar la simbología para utilizar cuando se muestran las entidades. Los clientes pueden ejecutar consultas para obtener entidades y realizar ediciones que se puedan aplicar en el servidor. Los servicios de entidades proporcionan plantillas que se pueden utilizar para obtener una experiencia de edición mejorada para el cliente. Los datos de las clases de relación y las tablas no espaciales también se pueden consultar y editar a través de los servicios de entidades.

.DotSpatial

DotSpatial [71] es una biblioteca de sistemas de información geográfica escrita para .NET 4 Framework Permite a los desarrolladores incorporar datos espaciales, análisis y funcionalidades de mapa en sus aplicaciones.

DotSpatial proporciona un control de mapa de uso totalmente gratuito para .NET que permite:

- Visualizar mapas en formularios Windows o Web desarrollados con .Net
- Abrir archivos en formato shapefile, grillas, rasters e imágenes.
- Simbología render.
- Leer datos GPS.
- Análisis científico.
- Manipular y desplegar atributos de los datos geográficos.

Los requerimientos de sistema para *DotSpatial* son los siguientes:

- Sistema operativo Windows XP o superior.
- Plataforma .net 4.0.
- Entorno de desarrollo Visual Studio o Sharp Develop.

El diseño de la arquitectura de *DotSpatial* sigue dos paradigmas. El primero es separar las interfaces gráficas de usuario de la lógica empresarial. El segundo consiste en utilizar un mayor número de módulos para aumentar la reutilización del código, y permitir el acceso a las partes del framework sin necesidad de tenerlo todo.

A continuación se presenta un esquema con los principales componentes de cada capa que forman parte de la arquitectura de *DotSpatial* (Figura 106).

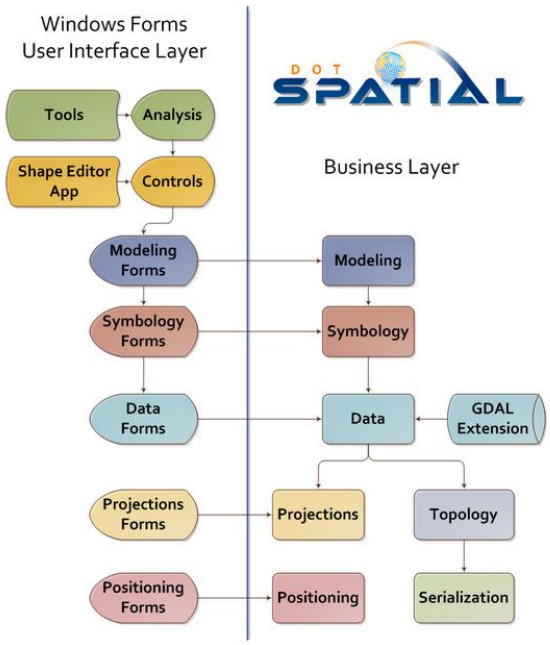

Figura 106: Arquitectura .DotSpatial

En la Tabla 37 se describe la funcionalidad principal de cada paquete *DotSpatial*.

Nombre del Paquete (DotSpatial.)	Contenido
Analisys	WhiteBox y algunas funcionalidades de MapWindows
Data	Acceso a datos de objetos de tipo vectorial, raster e imagines.
Data.Rasters.GdalExtension	Soporte GDAL para imagines y archivos raster
Modeling	Espacio reservado para código de modelización no-GUI
Positioning	GeoFrameworks 2 y GPS.NET 3
Projections	Proj4
Serialization	XML serialization para guarder proyectos.
Symbology	Capas y símbolos cartográficos
Topology	Métodos topológicos. NetTopologySuite/JavaTopologySuite
Controls	Map, Legend, Ribbon, Toolbox, Layout etc.
Data.Forms	Diálogos de datos y diálogos para abrir archivos
Modeling.Forms	Controles de modelado, herramientas y elementos de diálogo.
Projections.Forms	Diálogos windows form relativos a proyeccciones.
Symbology.Forms	Diálogos windows form y componentes para simbología.

Tabla 37: Descripción del contenido los paquetes de .DotSpatial

Bing Maps

Bing Maps [72] ofrece acceso a imágenes y datos de mapa así como también búsqueda robusta, localización y servicios de ruteo.

Sin embargo para tener acceso a dicha información es necesario registrarse explícitamente en Microsoft. Existen varios tipos de licencia, entre los tipos de licencias gratuitas se destacan:

- Licencia para instituciones educativas: Tienen derecho a esta licencia profesores, estudiantes y personal de instituciones educativas como escuelas públicas o privadas, universidades, etc. Su uso no debe ser para un proyecto de investigación financiado por una empresa comercial o compañía privada.

  Se restringe su uso a sitios web públicos, o sea que no tengan contraseña protegida.

  Se provee acceso gratuito a la plataforma *Bing Maps* incluyendo: acceso completo a la API, gama completa de mapas con vista aérea, de calle y Streetside, acceso a artículos de documentación y acceso al foro de soporte a desarrolladores.
- Licencia a organizaciones sin fines de lucro: Acceden a los mismos servicios que la licencia educativa. Tiene también la restricción que el sitio web debe ser público.
- Licencia de desarrollador: Acceden a los mismos servicios que las licencias anteriores pero se a la restricción de que el sitio web sea público se le agrega una restricción de cantidad de accesos, limitada a 125.000 sesiones o 500.000 transacciones máximo en el período de un año.

La plataforma BingMaps cuenta con varias opciones para desarrollar aplicaciones de mapas:

- *Bing Map* App SDK: la SDK permite desarrollar aplicaciones de mapas propias y así como también extensiones (plugins) para que podrán ser utilizados por otros desarrolladores.
- *Silverlight* Map Control: este control permite desarrollar aplicaciones de mapas usando el entorno de desarrollo de Silverlight con Expression Blend y Visual Studio.
- *Bing Map* AJAX Control: es una API para *JavaScript* que permite integrar mapas a un sitio web.
- Windows Phone 7 SDK
- *Bing Maps* REST Services: utiliza servicios REST que son accedidos mediante llamadas *HTTP* y permiten para realizar tareas como crear mapas, geocodificación de direcciones y cálculo de rutas.

GoogleMaps

GoogleMaps API

Google Maps [73]dispone de varias APIs de desarrollo entre ellas:

Google Maps JavaScript API: permite insertar *Google Maps* en páginas web. La versión 3 de esta API está especialmente diseñada para proporcionar una mayor velocidad y que se pueda aplicar más fácilmente tanto a móviles como a las aplicaciones de navegador de escritorio tradicionales. Esta API proporciona diversas utilidades para manipular mapas y para añadir contenido al mapa mediante diversos servicios, permitiéndote crear sólidas aplicaciones de mapas en un sitio web. La versión 3 de *Google Maps JavaScript* API es un servicio gratuito disponible para cualquier sitio web que sea gratuito para el consumidor.

Google Maps JavaScript API para Flash: Esta API permite a los desarrolladores de Flex insertar *Google* Maps en aplicaciones Flash. De forma similar a la versión JavaScript, esta API de ActionScript proporciona varias utilidades para manipular y añadir contenido a los mapas a través de distintos servicios, lo que permite insertar aplicaciones de mapas interactivas y complejas en un sitio web.

Google Static Maps API: *Google* Static Maps API permite insertar una imagen de *Google Maps* en un sitio web sin utilizar *JavaScript* ni ningún sistema de carga de páginas dinámicas. El servicio *Google* Static Maps creará un mapa a partir de los parámetros de URL enviados a través de una solicitud HTTP estándar y generará una imagen de mapa que se puede mostrar en una página web. El uso del API de *Google* Static Maps está sujeto a un límite de 1000 solicitudes únicas (distintas) de imágenes por visitante y día.

Google Static Maps API crea mapas en distintos formatos, que se especifican a continuación:

- *Roadmap:* (predeterminado) especifica una imagen de mapa de carreteras estándar, como se muestra habitualmente en la página de *Google* Maps.
- *Satellite:* especifica una imagen obtenida por satélite.
- *Terrain*: especifica una imagen de mapa de relieve físico, en la que aparece terreno y vegetación.
- *Hybrid:* especifica un híbrido de imagen obtenida por satélite e imagen de mapa de carreteras, en la que aparece una capa transparente de calles principales y nombres de lugares en la imagen obtenida por satélite.

Servicios web de Google Maps API: son un conjunto de interfaces HTTP para los servicios de *Google* que proporcionan datos geográficos para las aplicaciones de mapas. Las solicitudes para servicios web del API de *Google* Maps que crean las aplicaciones del API de *Google* Maps (edición premier) requieren firmas digitales que utilizan una clave criptográfica que se proporciona para tal fin. Entre estos servicios se encuentran:

- *Google Directions API*: es un servicio que utiliza una solicitud HTTP para calcular rutas para llegar de una ubicación a otra. Las rutas pueden especificar los orígenes, los destinos y los hitos como cadenas de texto o como coordenadas de latitud/longitud. Directions API puede devolver rutas segmentadas mediante una serie de hitos. El uso de *Google* Directions API está sujeto a un límite de consultas de 2.500 solicitudes de indicaciones al día. Las solicitudes individuales de indicaciones pueden incluir 8 hitos intermedios como máximo por solicitud. Los clientes de *Google* Maps Premier pueden consultar hasta 100.000 solicitudes de rutas al día y cada una de ellas puede incluir hasta 23 hitos.
- *API de matriz de distancia de Google:* es un servicio que proporciona el tiempo y la distancia de viaje para una matriz de orígenes y destinos. La información devuelta se basa en la ruta recomendada entre los puntos de partida y llegada, según los cálculos del API de *Google* Maps, y se compone de dos filas que incluyen los valores de duración y distancia para cada par. El servicio no devuelve información detallada de la ruta. cuenta con los siguientes límites para uso gratuito: 100 elementos por consulta, 100 elementos en 10 segundos, 2 500 elementos en un período de 24 horas.
- *Elevation API*: proporciona datos sobre elevación para todas las ubicaciones en la superficie terrestre, incluidas las ubicaciones en la profundidad del océano (que devuelve valores negativos). En estos casos en los que Google no disponga de medidas de elevación exactas sobre la ubicación concreta que haya sido solicitada, el servicio interpolará y devolverá un valor medio, para lo que utilizará las cuatro ubicaciones más cercanas. El uso del API de elevación de Google está sujeto a un límite de 2.500 solicitudes al día (los usuarios de la versión premier pueden enviar hasta 100.000 solicitudes al día). En cada solicitud se puede consultar la elevación de hasta 512 ubicaciones, aunque no se puede superar el total de 25.000 ubicaciones al día (1.000.000 para usuarios de la versión premier).
- *Google Geocoding API*: proporciona una forma directa de acceder a un geocodificador mediante solicitudes HTTP. Además, el servicio permite realizar la operación contraria ("codificación geográfica inversa"). El uso de Google Geocoding API está sujeto a un límite de consultas de 2.500 solicitudes de codificación geográfica al día.
- *API de Google Places*: es un servicio que devuelve información sobre lugares, definidos en el API como establecimientos, ubicaciones geográficas o lugares de interés importantes, mediante solicitudes HTTP. Las solicitudes de lugar especifican ubicaciones en forma de coordenadas de latitud/longitud.

Están disponibles cuatro solicitudes de lugar básicas:

- Búsquedas de lugares: devuelve una lista de lugares cercanos en función de la ubicación del usuario.
- Solicitudes de detalles de lugar: devuelve información más detallada sobre un lugar concreto.
- Visitas de lugar: permite registrar las visitas de un usuario a un lugar. Las visitas se utilizan para medir la popularidad de un lugar; las visitas frecuentes mejorarán la clasificación de un lugar en los resultados de búsqueda de lugar de tu aplicación.
- Informes de lugar: permite añadir lugares nuevos al servicio de *Google Places* y eliminar aquellos que haya añadido tu aplicación.

Extensión de ArcGIS Extension para la API de Google Maps

La extensión de *ArcGIS* para la API de *Google Maps* permite usar mapas y tareas *ArcGIS* en el framework de *Google Maps*. Se utiliza JavaScript tanto para hacer peticiones al servidor *ArcGis* como para desplegar dicha información en el mapa de *Google*.

No es necesario tener instalado un *ArcGIS* server para programar con esta extensión, solamente se necesita tener un servidor ArcGIS disponible a través de una URL

OpenStreetMap

OpenStreetMap [74] es un proyecto dirigido expresamente a crear y ofrecer datos geográficos libres, tales como planos de calles, a cualquiera que los desee. El proyecto comenzó debido a que muchos mapas que se cree que son libres, tienen en realidad restricciones legales o técnicas para su uso.

OpenStreetMap facilita los datos en bruto para su descarga desde su propia página web. Estos pueden ser modificados para cada proyecto así como presentados con estilos personalizados. La cobertura de datos se extiende al conjunto de todos los países del mundo.

OpenStreetMap se utiliza únicamente como fuente de datos geográficos, para incluirse en aplicaciones web desarrolladas con alguna biblioteca como ser *OpenLayers*, biblioteca *Javascript* de *GoogleMaps*, etc.

Conclusiones

Aplicaciones de simulación

A lo largo de este estudio se pudieron relevar una cantidad de herramientas de simulación, sin embargo no fue posible tener conocimiento en ningún caso de la tecnología con la cual se desarrollaron las salidas visuales ya que los fabricantes no hacen pública esta información.

Sobre las dos interrogantes que nos planteamos contestar como objetivo del estudio, la primera era determinar si seguía siendo válido modelar y simular en una herramienta y luego visualizar la salida de dicha simulación en otra. Al respecto se puede concluir que en la actualidad todavía existen herramientas comerciales que utilizan esta práctica.

El segundo objetivo era tomar conocimiento del tipo y calidad de la salida visual de los productos comerciales. En este punto se concluye que la tendencia del mercado es dar la opción de ver las simulaciones en un escenario 3D.

Por último, no se pudo determinar si las herramientas de simulación analizadas que son específicas para transporte público utilizan alguna herramienta de manipulación de mapas para su implementación.

Tecnologías analizadas

A continuación se describirán las evaluaciones que se realizaron de cada tecnología.

MapWindow

La principal ventaja de *MapWindow* es que trabaja sobre *Shapefiles* en forma nativa, lo que soluciona el problema de los datos geográficos de la ciudad de Rivera ya que se utilizarían los mismos que se utilizan para generar los proyectos en IgoR-tp. Esta tecnología es la que su usó para el desarrollo del proyecto IgoR-tp original. Por lo tanto, surge como una opción para el desarrollo del visor.

Otra ventaja es el conocimiento adquirido al usarla previamente en el proyecto, sin embargo es por este mismo motivo que se descarta esta tecnología por completo, ya que a lo largo del proyecto se encontraron numerosas dificultades técnicas. Entre ellas:

1. Problemas con el instalador de la biblioteca, en algunos sistemas (en particular en el PC del usuario) no actualiza correctamente el Runtime de C. Este runtime es necesario para la correcta ejecución del componente OCX.
2. Incompatibilidad con Windows 7 y Windows XP: alguna versión del componente OCX instalado en Windows XP provocaban caídas de la aplicación, sin embargo no se generaban estos errores cuando la misma versión era instalada en Windows 7.
3. Falta de soporte técnico: en varias ocasiones se encontraron errores del componente de los que se no encontró documentación alguna.

ArcGIS – BingMaps – GoogleMaps

Estas tres tecnologías se investigaron por ser referentes en esta área de estudio. Por otro lado, uno de los requerimientos de la nueva herramienta de simulación era "El software de visualización debe trabajar con la tecnología más nueva posible para el despliegue de gráficos en pantalla".

Tienen en común que de seleccionarlas, se necesitaría conexión a internet para desplegar los datos de la simulación. A priori se podría utilizar cualquiera de ellas para la implementación de la nueva herramienta de visualización, ya que las tres brindan las funcionalidades necesarias.

La idea era utilizar este tipo de tecnología como mapa base y luego dibujar sobre el mismo los buses, paradas y pasajeros. Para que esto fuera posible lo primero que debían proveer era un mapa detallado de la ciudad de Rivera. Al momento de realizada la evaluación de tecnologías la única que cumplía este requisito era *GoogleMaps.*

Luego se debía investigar si el sistema de coordenadas utilizado por los datos geográficos de Rivera que maneja igoR-Tp y por tanto las coordenadas de los datos de las entidades del simulador, eran compatibles con el sistema de coordenadas de los mapas seleccionados. Es en este punto donde se descartan las tres tecnologías y se empieza a buscar otra alternativa, ya que no fue posible hacer coincidir las coordenadas de los datos que se tienen como entrada (*Shapefile*) con las coordenadas de la ciudad de Rivera de los mapas web disponibles. El problema específico con los datos con los que se cuenta es que se desconoce su DATUM [75] .

Surge entonces como una opción alternativa utilizar la *ArcGis API Silverlight* pero cargando los datos geográficos desde los archivos *Shapefile* del caso de estudio de Rivera (ver sección 5.3.1).

OpenStreetMap

En la búsqueda de mapas que incluyeran los datos de la ciudad de Rivera analizamos como otra posibilidad utilizar *OpenStreetMap* como fuente de mapas para utilizar con alguna de las herramientas web analizadas.

Si bien en la actualidad *OpenStreetMap* cuenta con un mapa detallado de la ciudad de Rivera, al momento de realizada nuestra investigación estos datos no estaban disponibles por lo que se descartó esta opción.

DotSpatial

Como se mencionó anteriormente *.DotSpatial* permite manejar datos geográficos de tipo *Shapefile* de forma simple y sin necesidad de tener conexión a internet. El control de manejo de mapas que provee se incluye fácilmente a un formulario C# y brinda un número importante de funcionalidades sin necesidad de programación extra. Entre ellas se destacan: zoom in, zoom out y mover mapa, tres funcionalidades que están entre los principales requerimientos de la nueva herramienta de visualización.

Por otra parte al estar esta biblioteca implementada en .Net, no es necesario registrar componentes (como pasaba por ejemplo con el componente OCX de *MapWindows*) basta con adjuntar los archivos ".dll" de la biblioteca con el ejecutable de la aplicación, para garantizar la ejecución de la misma en cualquier sistema que tenga instalado el framework *.Net.*

Al haberse descartado el uso de otras tecnologías como ser *GoogleMaps* y *BingMaps* por el problema con las coordenadas de los datos, la ventaja de poder usar los datos de la ciudad de Rivera en formato *Shapefile* hacen de esta biblioteca un candidato de peso para implementar la nueva herramienta de visualización.

Sin embargo es necesario hacer un prototipo para evaluar la performance del control de mapas al utilizarlo para realizar una animación de la simulación de transporte público de la ciudad de Rivera, ya que es un riesgo que con un número importante de datos el control no se comporte como es esperado. (Por detalles de este prototipo ver sección 5.3.2).

Apéndice E – Material extra Simulador

1. Conceptos de Simulación a Eventos Discretos

La Simulación a Eventos Discretos de ahora en más *SED*, es un proceso cuyos modelos de simulación se representan por una serie de eventos, los cuales cambian a medida que el tiempo transcurre, permitiendo que los sistemas evolucionen a través de estos [8]. Es deseable que el modelo sea simple pero que mantenga sus objetivos y que produzca resultados verificables.

Cada modelo de *SED* es descrito mediante un conjunto de conceptos que se resumirán a continuación:

- El concepto de tiempo es fundamental en los modelos porque la evolución de los mismos se produce a medida que los golpes de reloj producen cambios de estado. El tiempo de simulación se da en unidades de tiempo acordes según cada caso. Cuando comienza un sistema de simulación, lo hace en tiempo 0 y luego ejecuta todos los eventos (en el orden que fueron registrados) hasta que se cumple una de las siguientes condiciones:
    - no existen más eventos por ejecutar.
    - la ejecución del próximo evento, supera el tiempo de duración de la simulación.
    - se ejecutó un evento de fin de simulación.
- Los eventos son ocasionados por las entidades al momento de realizar algún cambio que afecta al sistema. Existen eventos fijos y condicionados. El evento fijo o “evento B” se sabe de antemano cuando va a ocurrir por lo tanto es predecible; mientras que el evento condicionado o “evento C” ocurre según ciertas restricciones. Cada vez que una entidad participa de un evento B, el tiempo de ocurrencia del mismo se escribe en su reloj interno.
- Existe un evento llamado alimentador el cual es el encargado de generar entidades al sistema.
- Las entidades son objetos cuyas actividades son de interés para el modelado. Las entidades cambian de estado a lo largo del sistema. Sus estados pueden ser: ocupada, encolada y desocupada. Los recursos son objetos que restringen las actividades de las entidades. Son instrumentos que se toman y liberan por parte de las entidades, según sus requisitos. Las entidades están relacionadas con una o varias actividades.
- Las actividades son acciones que realizan las entidades a lo largo de su existencia en el sistema. Generalmente comienzan con eventos condicionados y finalizan con eventos fijos. A través de las actividades las entidades interaccionan con otras entidades y transitan por el sistema.
- Existe un procedimiento principal llamado ejecutivo, el cual es el encargado de la ejecución ordenada de los eventos en el sistema y es el responsable del mecanismo de avance del tiempo de la simulación. Para ello necesita un calendario, el cual es una lista de entidades que identifican los próximos eventos a ejecutar o una lista de eventos identificando las entidades participantes en los mismos. El calendario mantiene un ordenamiento dado por los tiempos de las entidades asociadas. El funcionamiento del ejecutivo se puede observar en la Figura 107.

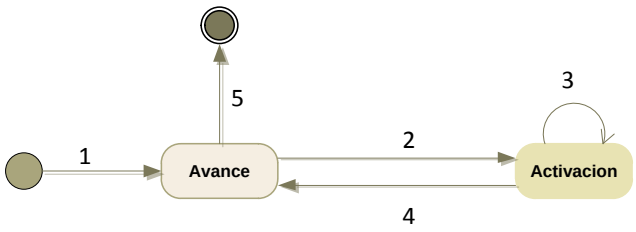

Figura 107: Diagrama del ejecutivo

Al comenzar la simulación se inicializa el ejecutivo, se agenda una entidad en el calendario (feeder o alimentador) y se pasa al estado de **Avance**, como muestra la transición 1. En este estado, el ejecutivo avanza el tiempo de la simulación hasta el tiempo del próximo evento indicado por la primera entidad del calendario. Si no se llegó al fin de la simulación, el ejecutivo pasa al estado de **Activación** como lo indica la transición 2. En caso contrario se finaliza la simulación como se observa en la transición 5.

En el estado de **Activación**, se activará el evento asociado a la entidad y se procederá de igual forma para todos los eventos agendados en el mismo tiempo de reloj como lo indica la transición 3. Una vez finalizada la activación de los eventos para ese golpe de reloj, se retorna al estado de **Avance** como se ve en la transición 4.

El ejecutivo puede componerse por distintos métodos de estructuración de la simulación que se corresponden a distintos enfoques de la realidad a estudiar.

1. Dos fases. La lógica de los eventos condicionados esta embebida en los eventos fijos. Su orientación está dada por los eventos.
2. Tres fases. Los eventos fijos y condicionados se programan como procedimientos separados, lo cual presenta, como en el caso anterior una orientación a eventos.
3. A procesos.

El proceso de activar los eventos por parte del ejecutivo será de mayor o menor complejidad. Si se utiliza un ejecutivo de dos fases entonces solamente se activaran los eventos fijos agendados en el calendario. Si se utiliza el de tres fases, primero se activarán los eventos fijos, e inmediatamente después se verificarán qué eventos condicionados cumplen las condiciones para su ejecución y se activarán en tal caso. La Figura 108 muestra cómo se divide el estado de **Activación** según la estructuración del ejecutivo:

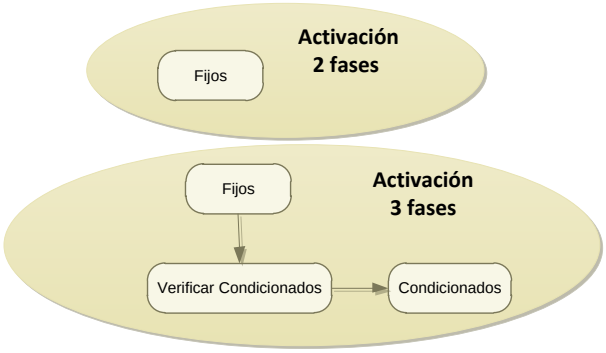

Figura 108: Activación según estructura del ejecutivo.

Existe una descripción visual de las entidades del sistema y la interacción con otras entidades mediante un artefacto llamado diagrama de actividades. Dicho diagrama permite estudiar el ciclo de vida de las entidades, desde su creación hasta la destrucción, el uso de recursos y colas. La Figura 109 es un ejemplo de un diagrama de actividades para un caso sencillo de un hospital extraído de [76].

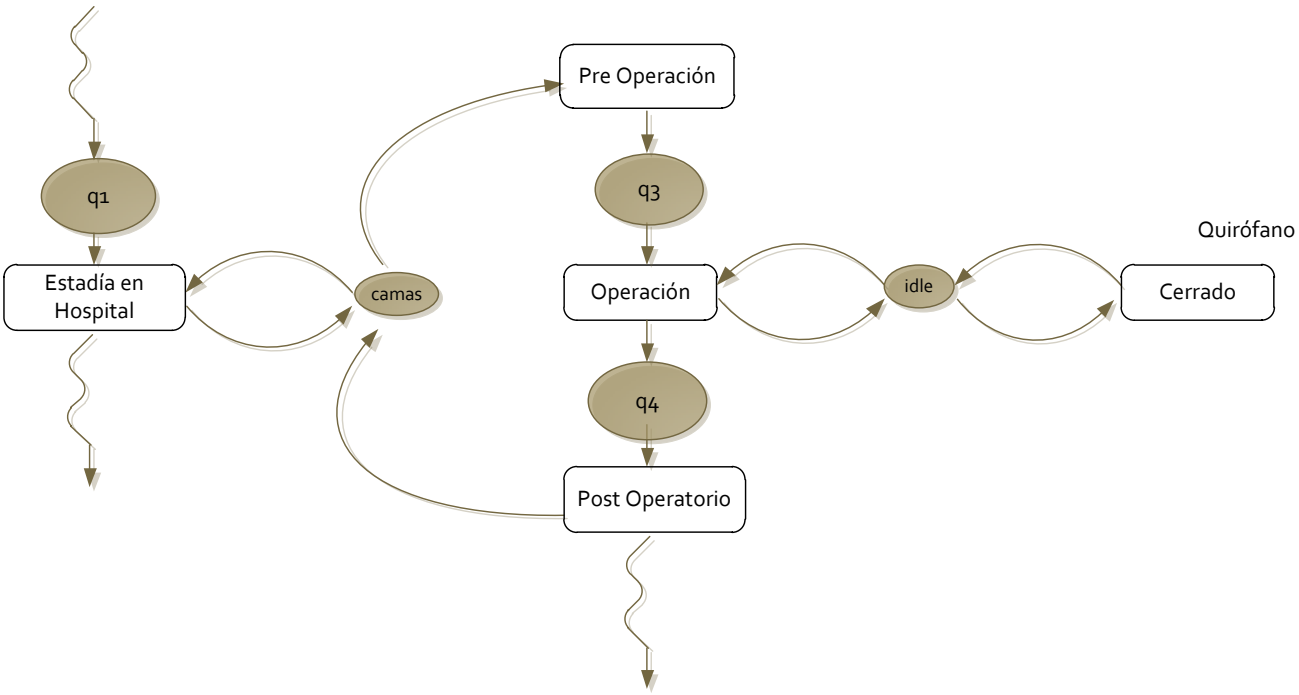

Figura 109: Ejemplo diagrama de actividades.

Las colas se representan con círculos, las actividades con rectángulos y los recursos por círculos pequeños. La creación y la destrucción de las entidades se simbolizan por las líneas en zigzag. El sentido de las interacciones está dado por las líneas dirigidas. Los modelos basados en S.E.D incluyen diagramas de este tipo en la especificación del modelo.

2. Eventos del modelo de sistema de transporte público

A continuación se brinda la descripción formal de cada evento del sistema con su correspondiente diagrama de flujo. Estos diagramas son complementarios al pseudocódigo de los eventos presentados en [3].

- Subida Pasajero Bus: Evento Condicionado ***{C1}***

Este evento es el encargado de gestionar para todas las paradas, la subida de los pasajeros a los buses. El diagrama asociado se puede observar en la Figura 110.

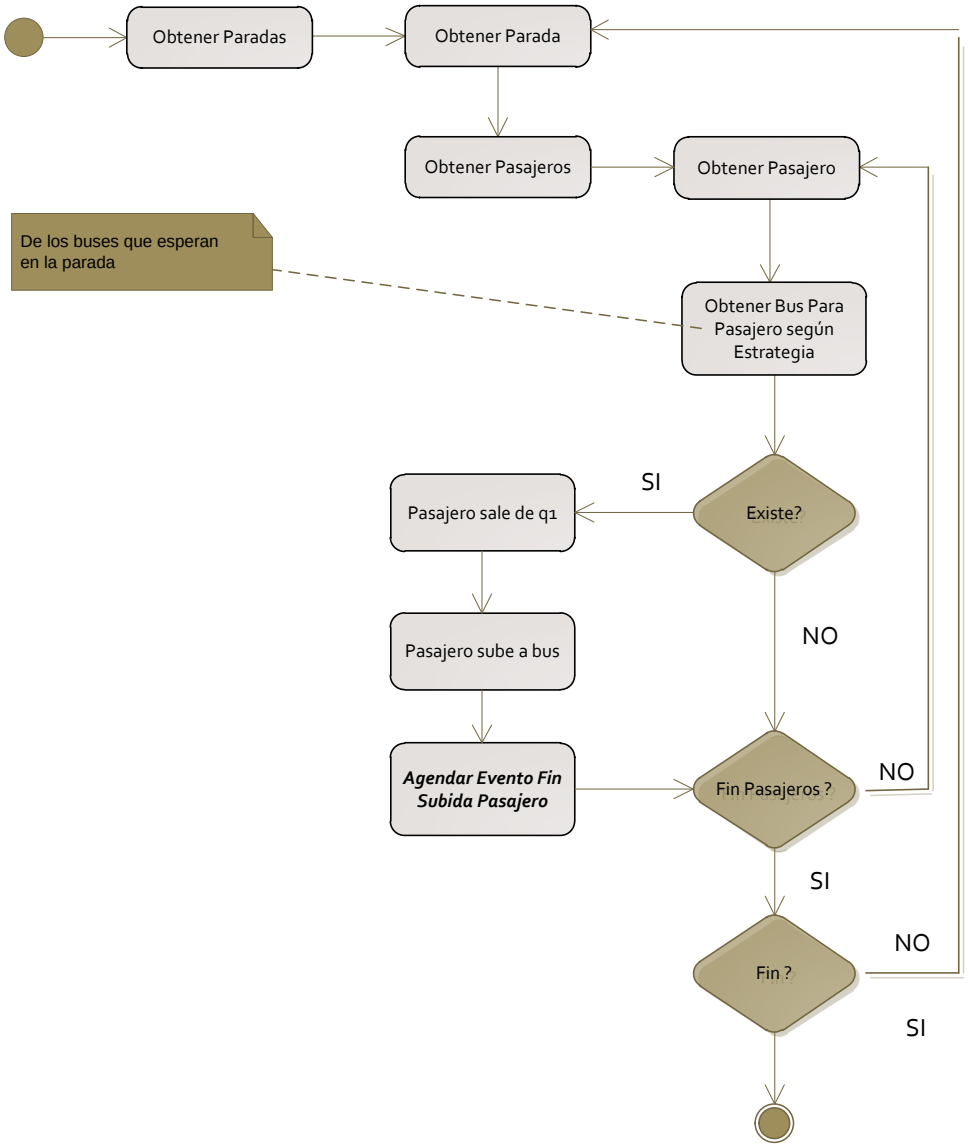

Figura 110: Evento Subida Pasajero Bus

- Comienzo Recorrido Bus: Evento Fijo ***{B1}***

Cuando un bus comienza su recorrido, se obtiene el tiempo entre el nodo actual y el siguiente y se procederá a avanzar el bus hacia el próximo nodo vial de su recorrido. La Figura 111 refleja dicha secuencia.

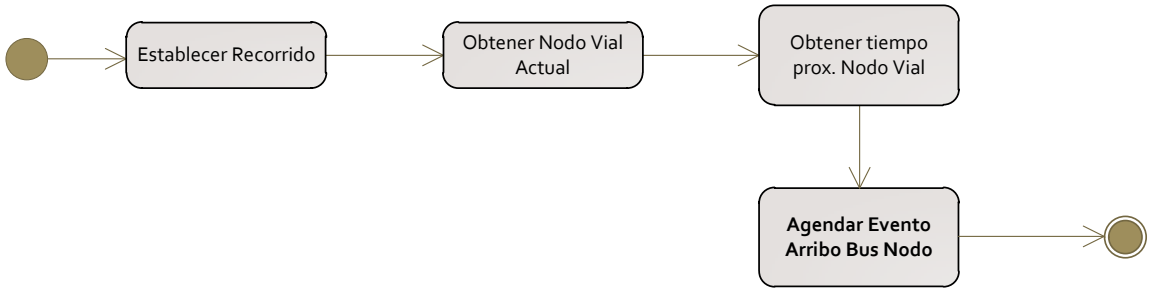

Figura 111: Evento Comienzo Recorrido Bus

- Arribo bus a Nodo (Vial): Evento Fijo ***{B2}***

  En el momento que un bus arriba a un nodo, se procesará mediante este evento. Se bajarán los pasajeros que así lo deseen y se procederá a continuar el viaje del bus. La Figura 112, refleja dicho evento.

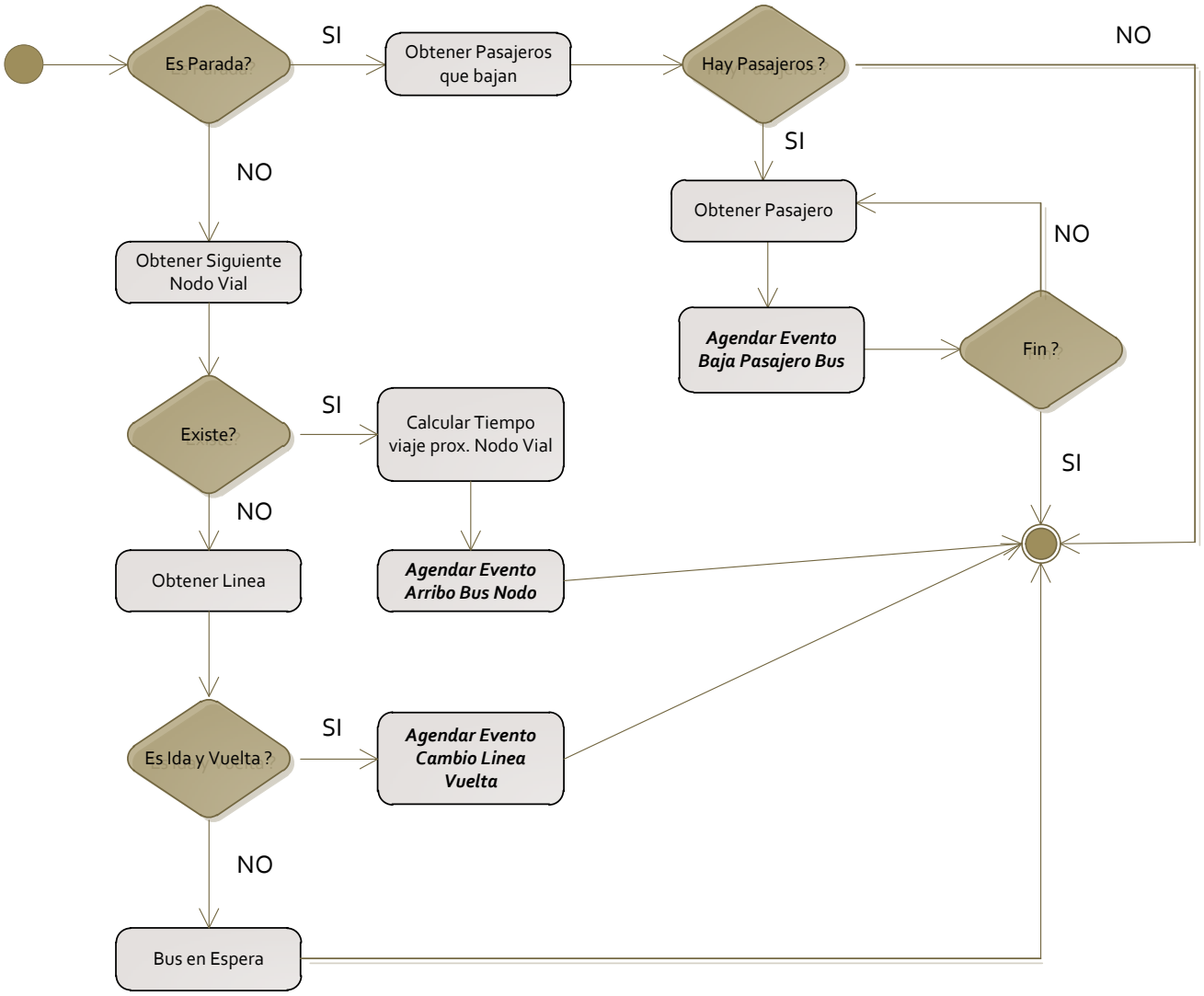

Figura 112: Evento Arribo Bus a Nodo Vial

- Comienzo Viaje Pasajero: Evento Fijo ***{B3}***

  Si un pasajero comienza su viaje, este evento es el encargado de calcular el tiempo de viaje desde el centroide origen a la parada y asignarle una estrategia. La Figura 113, muestra este evento.

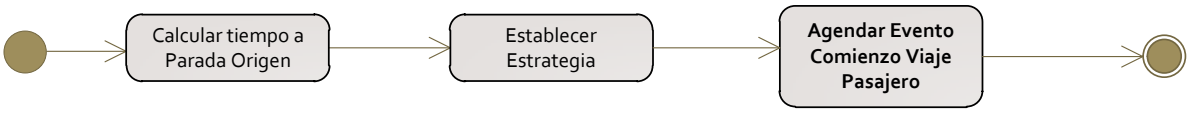

Figura 113: Evento Comienzo Viaje Pasajero

- Arribo Pasajero a Parada: Evento Fijo ***{B4}***.

  Al momento de arribar a una parada, el pasajero se encolará en la misma. La Figura 114 muestra este evento.

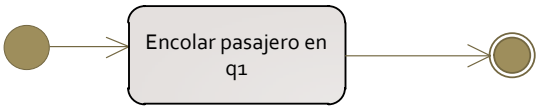

Figura 114: Evento Arribo Pasajero Parada.

- Baja Pasajero Bus : Evento Fijo ***{B5}***

  Este evento representa la acción que realiza un pasajero al momento de bajar de un bus en una parada intermedia (de trasbordo) o de destino. Si la parada es de destino, se calcula tiempo de viaje hacia la caminata destino; en caso contrario no se encola nuevamente el pasajero en la parada de trasbordo. La Figura 115 refleja dicho comportamiento.

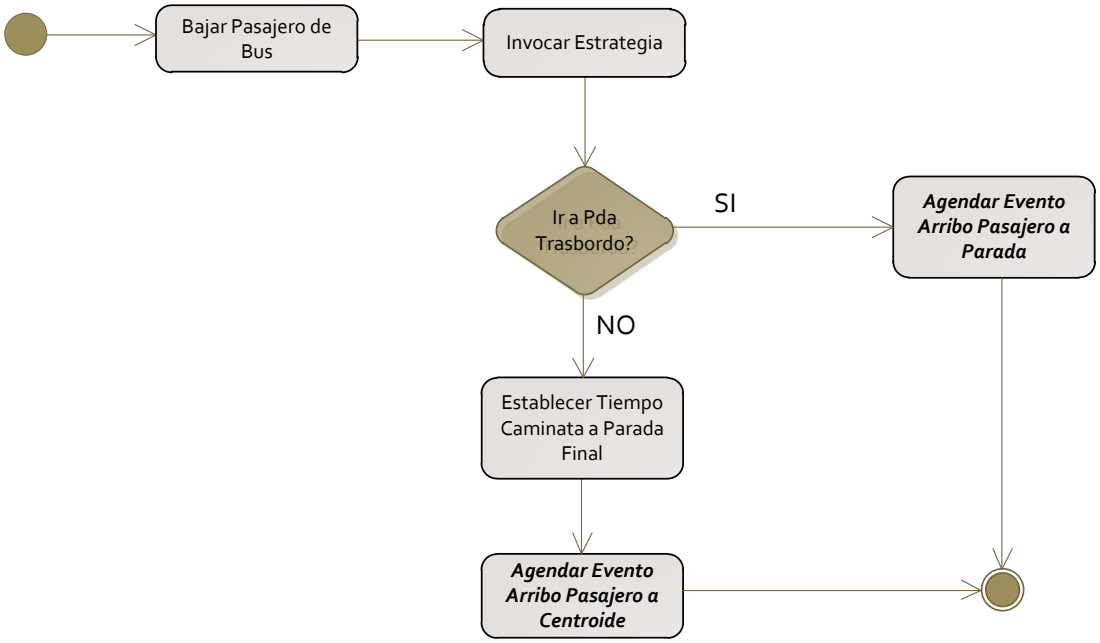

Figura 115: Evento Baja Pasajero Bus

- Arribo Pasajero Centroide: Evento Fijo. ***{B6}***

  Se dispara cada vez que un pasajero arriba al centroide destino y sale del sistema. La Figura 116 muestra el diagrama asociado.

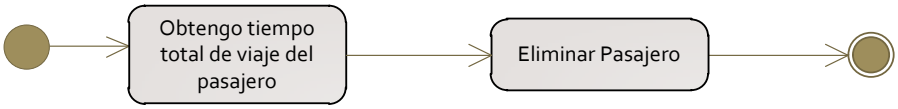

Figura 116: Evento Arribo Pasajero Centroide

- Fin Subida Pasajeros: Evento Fijo ***{B7}***

  Este evento simplemente indica al pasajero que ya subió al bus. La Figura 117 muestra la secuencia del evento.

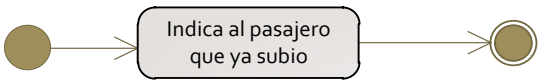

Figura 117: Evento Fin subida Pasajero

- Frecuencia Bus: Evento Fijo ***{B8}***

  Este evento es el encargado de controlar que los buses viajen por el sistema a la frecuencia indicada. Despachará los buses existentes que estén en espera o generará nuevos buses de tal forma de mantener al sistema “abastecido”, siempre respetando las frecuencias de cada línea. La Figura 118 muestra la secuencia de este evento.

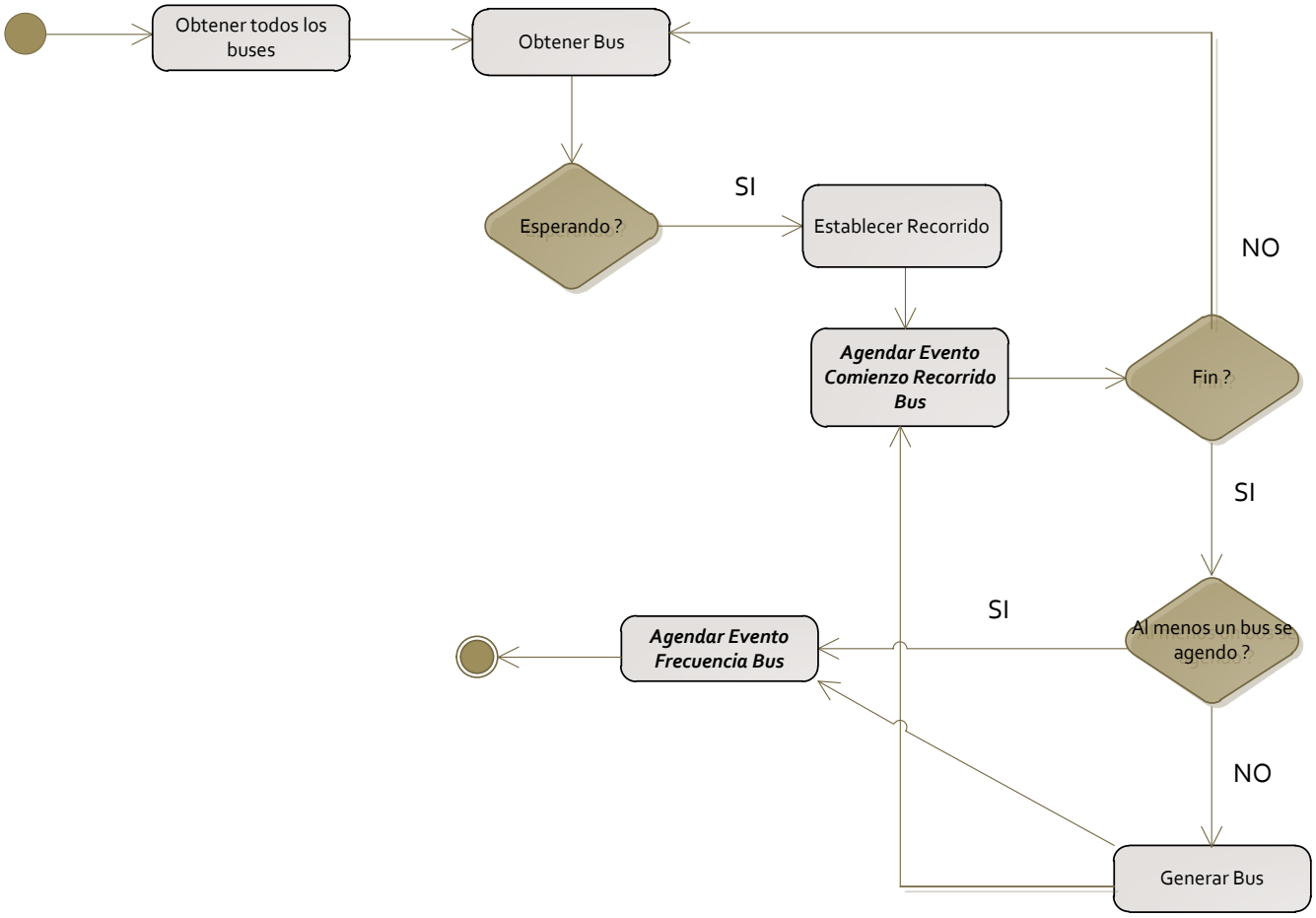

Figura 118: Evento Frecuencia Bus

- Cambio línea vuelta: Evento Fijo ***{B9}***

  Este evento es el encargado de establecer el recorrido de vuelta de un bus. La Figura 119 muestra la secuencia de este evento.

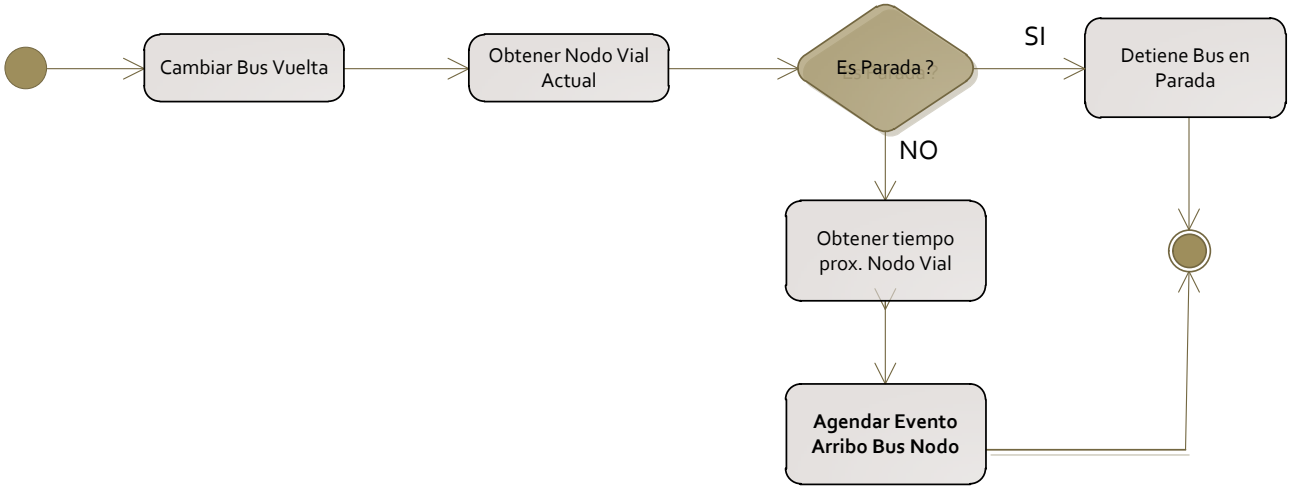

Figura 119: Evento Cambio Línea Vuelta

3. Invocación del Simulador

El proyecto simulador genera, cada vez que se compila exitosamente, un ejecutable llamado *PublicTransport.exe* que se encuentra dentro de la carpeta */bin* del proyecto de *Visual Studio* [31]. Al invocar dicho ejecutable, se desplegará una consola del sistema operativo *Windows* donde se llevará a cabo la simulación.

El simulador tiene un conjunto de argumentos obligatorios y un conjunto de banderas optativas. Para implementar la lectura de comandos se utilizó la biblioteca *Argument Helper* desarrollada por Daniel Russel [77], esta biblioteca permite que las banderas tengan más de un formato, uno de dichos formatos es compacto y el otro es largo. El formato compacto consta del carácter "-" seguido de una letra, el formato largo consta de dos caracteres "--" seguidos de una palabra. La ventaja del segundo formato sobre el primero es que es más fácil de recordar ya que las palabras son verbos o acciones a realizar por el simulador. La Tabla 38 indica cuales son los argumentos necesarios así como las banderas para ejecutar el visualizador:

Argumentos:	
Grafo	Archivo *grafo.txt* obtenido desde el módulo de manipulación de igoR-tp
Solución	Archivo *.solucion* generado por el módulo manipulación de igoR-tp
Demanda	Archivo de demanda generado desde el módulo de manipulación de igoR-tp
Lineas	Archivo que define lineas de buses y frecuencias a partir de recorridos.
Banderas(corto/largo):	
-h / --help	Ayuda en pantalla.
-e / --espera	Habilita tomar tiempos de espera en parada de Buses para cálculo de Z2.
-d / --default	Ignora todos los argumentos y ejecuta con valores por defecto.
-f / --sdlfast	Habilita la salida visual SDL original del simulador a máxima velocidad.
-l / --log	Habilita la generación del log de salida para la herramienta de visualización.
-s / --sdl	Habilita salida en pantalla de la simulación por SDL a velocidad normal.

Tabla 38: Argumentos y banderas del simulador *PublicTransport*

A continuación se detalla una salida de la consola en pantalla al momento de ejecutar el simulador mediante la bandera -h o --help:

```
>PublicTransport –help

Uso:
PublicTransport grafo solución demanda lineas [repeticiones] [estrategia]
[tiempo]

Argumentos:
 grafo          Archivo grafo.txt que se encuentra en manipulación
 solucion       Archivo .solución generado por manipulación
 demanda        Archivo de demanda generado por construcción
 lineas         Archivo de líneas para recorridos
 repeticiones   Cantidad de experimentos (por defecto es 1)
 estrategia     0-Original, 1-Trasbordo (por defecto es 1)
 tiempo         Segundos de simulación (por defecto es 21600 = 6hs)


Banderas:
 -d/--default   Permite ignorar todos los argumentos y ejecutar con valores
por defecto
```

```
 -e/--espera    Habilita espera de Buses en parada (Solo para calcular Z2)
 -f/--sdlfast   Habilita la salida visual de la simulación a máxima
velocidad
 -l/--log       Habilita la generación de un log para el visor
 -s/--sdl       Habilita la salida visual de la simulación
```

Como se observa en el modo de uso, el simulador necesita como argumentos las rutas de cuatro archivos (los tres primeros se obtienen desde un modelo de manipulación de igoR-tp y el cuarto se debe crear en forma manual), pero también se puede especificar argumentos opcionales (los que están entre corchetes) cantidad de repeticiones, estrategia a usar y tiempo de simulación. Si alguno de estos tres se ignora, se cargan valores por defecto.

En caso que falte alguna ruta de los archivos, se muestra la ayuda indicando el archivo faltante. Si se ingresan los cuatro archivos y alguno no existe o no es accesible, indica cual o cuales no existen y termina la ejecución. Se pueden omitir todos los argumentos e indicar que los cargue por defecto con **-d** o **--default**. Los archivos por defecto se encuentran en la carpeta */entrada* que debe estar al mismo nivel de directorio que el archivo ejecutable del simulador.

Una ventaja del uso de banderas desde la línea de comandos, es que estas se pueden invocar de forma intercambiable. A continuación se presentarán algunos ejemplos de invocación del simulador:

- Para ejecutar una simulación con salida *SDL* a velocidad de simulación normal, utilizando los archivos de entrada alojados en la carpeta */entrada*, sin log de simulación, se realiza utilizando cualquiera de las combinaciones siguientes:

```
publictransport --default --sdl
publictransport -d -s
publictransport -d --sdl
publictransport --default –s
```

- Para ejecutar una simulación con salida visual *SDL* a velocidad máxima de simulación, generando log de simulación, accediendo a los archivos de entrada por defecto, se debe invocar:

```
publictransport --default --sdlfast –log
```

- Para ejecutar una simulación, con dos repeticiones de 30000 segundos, con estrategia de trasbordo de pasajeros, indicando explícitamente los caminos de los archivos a, b, c y d, se debe invocar de la siguiente forma:

```
publictransport a b c d 2 1 30000
```

- Ejecutar una simulación indicando explícitamente los caminos de los archivos a, b, c y d, generando salida de log de simulación, se invoca de cualquiera de las siguientes formas:

```
publictransport a b c d --log
publictransport a b c d –l
```

Apéndice F – Archivo para visualización de la simulación

El presente apéndice describe el nuevo archivo de salida del simulador (archivo log o simplemente log de aquí en más), el cual es el responsable de persistir una sola corrida de simulación. Este archivo surge de la necesidad de disponer de un medio de comunicación entre el simulador *PublicTransport* y la nueva *Herramienta de Visualización* de simulaciones.

El objetivo del archivo de log es registrar los cambios de estado de los objetos que son visibles y animados en la salida original *SDL* del simulador *PublicTransport*, para que posteriormente pueda ser reconstruida por otras aplicaciones (en nuestro caso la *Herramienta de Visualización)*.

El archivo de log es un archivo de texto en formato *XML*. Este formato tiene la ventaja de ser un estándar para el intercambio de datos, fácil de comprender por parte de un usuario o desarrollador, es ampliamente difundido e interpretado por muchos sistemas [33], además cuenta con bibliotecas robustas para la mayoría de los lenguajes de programación. En el caso del simulador *PublicTransport*, la biblioteca utilizada para generar el log es *XMLParser Library* [78] .

Este apéndice maneja términos de *simulación a eventos discretos* y del simulador *PublicTransport* por lo que se recomienda la lectura previa del *Apéndice E*. Conceptos tales como *evento*, *entidad* y *calendario* se asumen conocidos por parte del lector.

1. Diseño del archivo de log

Los únicos objetos animados durante una simulación en la salida original *SDL* del simulador *PublicTransport* son las entidades *Bus* y *Pasajero*, el resto de los objetos visibles son estáticos (calles, paradas, caminatas, centroides, etc.) y no cambian su estado visual durante toda la simulación, es por esto que los únicos datos incluidos en el archivo de log son los relevantes a las animaciones de Buses y Pasajero.

Los eventos implementados en el simulador *PublicTransport* se utilizan para definir en que momento una entidad debe realizar la acción asociada a los mismos.

Por ejemplo el evento *ComienzoViajePasajero* se dispara en un cierto tiempo *t* del calendario para una cierta entidad *e* (en este caso *e* es del tipo *Pasajero*). En el momento que se ejecuta el código asociado a este evento, el pasajero *e* es inicializado, se calcula cuanto tiempo le llevará desplazarse caminando a una parada, se agenda su siguiente evento que es *ArriboPasajeroParada* y se muestra en la salida *SDL* como un nuevo punto sobre el mapa.

Resulta natural modificar el código asociado al evento *ComienzoViajePasajero* para que además escriba una nueva línea en el log dejando registrado que en el tiempo *t* del calendario, el pasajero *e* comenzó su viaje desde las coordenadas *(x, y)* del centroide hasta las coordenadas *(x2, y2)* de la parada.

Si generalizamos este razonamiento, también se podría modificar *ArriboPasajeroParada* para que escriba en el log de simulación el tiempo *t'* en el que se dispara el evento y las coordenadas *(x2, y2)* de la parada a la que se está arribando.

Resumiendo, para construir el log de simulación es necesario identificar los eventos que alteran el estado visual de los *Buses* y los *Pasajeros* de forma que escriban una nueva línea en el log

indicando, la entidad afectada, el evento y el tiempo en el que se ejecuta dicho evento. Analizando el código del simulador se observa que los eventos que modifican el estado visual en la salida original *SDL* del simulador son, para Pasajeros:

- ComienzoViajePasajero
- ArriboPasajeroParada
- FinSubidaPasajero
- BajaPasajeroBus
- ArriboPasajeroCentroide

Para Buses:

- ComienzoRecorridoBus
- ArriboBusNodo
- CambioLineaVuelta

2. Formato del archivo de log

Como primera medida es necesario codificar los eventos para poder escribirlos en el log. La Tabla 39 muestra todos los códigos reconocidos y su relación con los eventos.

Se puede observar que algunos códigos están relacionados uno a uno con un evento, por ejemplo el código COMIENZO_RECORRIDO y el evento *ComienzoRecorridoBus*, sin embargo otros tienen múltiples relaciones como BAJA_PARADA_INTERMEDIA y ARRIBO_PARADA_FINAL que está relacionada a *BajaPasajeroBus*.

Este último caso se podría haber resuelto con un solo código, pero como el interés es reconstruir la animación considerando la nueva estrategia de trasbordo e interesa destacar con distintos colores cuando un pasajero espera en una parada de trasbordo o arriba a la parada destino, se añade complejidad al evento *BajaPasajeroBus* para que según el estado de la entidad distinga cada caso al escribir el log.

CODIGO	EVENTO
Bus	
COMIENZO_RECORRIDO	*ComienzoRecorridoBus*
ARRIBO_NODO_VIAL_LLENO	*ArriboBusNodo, CambioLineaVuelta*
ARRIBO_NODO_VIAL	*ArriboBusNodo, CambioLineaVuelta*
FIN_RECORRIDO	*ArriboBusNodo*
Pasajero	
COMIENZO_VIAJE	*ComienzoViajePasajero*
ARRIBO_PARADA	*ArriboPasajeroParada*
BAJA_PARADA_INTERMEDIA	*BajaPasajeroBus*
ARRIBO_PARADA_FINAL	*BajaPasajeroBus*
SUBE_BUS	*FinSubidaPasajero*
LLEGA_CENTROIDE	*ArriboPasajeroCentroide*

Tabla 39: Relación entre eventos del simulador y códigos de log

Como segunda medida es necesario identificar la entidad que se está registrando en el log, esto se resuelve fácilmente pues ambas entidades están identificadas por un número entero que se puede escribir en el log.

Por último es necesario registrar el tiempo en el que sucede el evento para la entidad, esto es trivial, pues el simulador ofrece un mecanismo para consultar el tiempo de simulación en cualquier evento, basta con consultar en el momento de escribir la línea en el log y añadir el valor de la consulta.

Se propone entonces, el siguiente formato de etiquetas *XML* para persistir los datos relevantes de la simulación asociados a las entidades *Bus* y *Pasajero*.

Es importante destacar que la solución propuesta utiliza una sola línea por entidad y parametriza los valores almacenados según el código de evento registrado.

2.1. Bus

El formato de etiqueta *XML* propuesto para registrar una línea de log con información relevante a una entidad *Bus* es la siguiente:

```
<Bus>
      <Tiempo>          </Tiempo>
      <Id>              </Id>
      <Estado>          </Estado>
      <X>               </X>
      <Y>               </Y>
      <X2>              </X2>
      <Y2>              </Y2>
      <TiempoViaje>     </TiempoViaje>
      <Linea>           </Linea>
</Bus>
```

Las primeras tres etiquetas del nivel interno de `<Bus>`, permiten persistir la hora, el identificador de la entidad y un código (obtenido de la Tabla 40) que identifica el evento que la escribió.

Las etiquetas `X` e `Y` almacenan la coordenada del Bus en el momento que se escribe la nueva entrada en el log, los demás valores luego de `X` e `Y` están parametrizados según la etiqueta de estado.

Las etiquetas `X2`, `Y2` y `TiempoViaje`, almacenan la coordenada destino de la siguiente parada o nodo vial y el tiempo que le lleva recorrer ese trayecto entre los dos puntos.

La etiqueta `Linea` almacena un código de línea y permite establecer a que línea de la simulación (identificada por el código) pertenece el *Bus*.

Como se mencionó anteriormente, los valores desde la etiqueta `X2` hasta `Linea` son utilizadas de acuerdo al código reconocido en la etiqueta `Estado`, por ejemplo si el valor de la etiqueta de estado fuera FIN_RECORRIDO, los valores que almacena `X2`, `Y2`, `TiempoViaje` y `Linea` son

inválidos y no deben ser tenidos en cuenta, pues carecen de sentido porque el Bus completó su viaje.

La Tabla 40 muestra las etiquetas con valores válidos (marcados como *x*) de acuerdo al código de evento para los registros de Buses.

	Tiempo	Id	Estado	X	Y	X2	Y2	TiempoViaje	Linea
COMIENZO_RECORRIDO	x	x	x	x	x	x	x	x	x
ARRIBO_NODO_VIAL_LLENO	x	x	x	x	x	x	x	x	x
ARRIBO_NODO_VIAL	x	x	x	x	x	x	x	x	x
FIN_RECORRIDO	x	x	x	x	x				

Tabla 40: Etiquetas válidas según el código de evento asociado a un Bus

2.2. Pasajero

El formato de etiqueta *XML* propuesto para registrar una línea de log con información relevante a una entidad *Pasajero* es la siguiente:

```
<Pasajero>
      <Tiempo>          </Tiempo>
      <Id>              </Id>
      <Estado>          </Estado>
      <X>               </X>
      <Y>               </Y>
      <X2>              </X2>
      <Y2>              </Y2>
      <TiempoViaje>     </TiempoViaje>
      <IdBus>           </IdBus>
</Pasajero>
```

Al igual que en el Bus, las primeras tres etiquetas del nivel interno de `<Pasajero>`, permiten persistir la hora, el identificador de la entidad y un código que identifica el evento que escribió la misma obtenido de la Tabla 41 (en este caso en la región de códigos de *Pasajeros*).

Las etiquetas `X` y `Y` almacenan la coordenada del Pasajero en el momento que se escribe la nueva entrada en el log, la etiquetas `X2`, `Y2` y `TiempoViaje`, almacenan (en algunos casos) la coordenada destino de la siguiente parada o centroide y el tiempo que le lleva recorrer ese trayecto entre los dos puntos.

La etiqueta `IdBus` se sobrecarga para no añadir más etiquetas, en caso del evento SUBE_BUS almacena un código de *Bus* en el que está siendo transportado el pasajero, en caso de los eventos BAJA_PARADA_INTERMEDIA y ARRIBO_PARADA almacena el identificador de la parada origen y para el caso de ARRIBO_PARADA_FINAL almacena el identificador de la parada destino.

Evidentemente `IdBus` tiene valores inválidos cuando el pasajero comienza su viaje o cuando camina entre paradas y centroides.

	Tiempo	Id	Estado	X	Y	X2	Y2	TiempoViaje	IdBus
COMIENZO_VIAJE	x	x	x	x	x	x	x	x	
ARRIBO_PARADA	x	x	x	x	x				x
BAJA_PARADA_INTERMEDIA	x	x	x	x	x				x
ARRIBO_PARADA_FINAL	x	x	x	x	x	x	x	x	x
SUBE_BUS	x	x	x						x
LLEGA_CENTROIDE	x	x	x	x	x				

Tabla 41: Etiquetas válidas según el código de evento asociado a un Pasajero

Es importante que las aplicaciones que realizan la lectura e interpretación del log (por ejemplo la *Herramienta de Visualización)* respeten las tablas (Tabla 40 y Tabla 41) que indican valores de etiquetas válidos de acuerdo al código de evento, pues el valor de la misma puede estar sobrecargado o puede ser inválido según el evento que la escribe.

2.3. Log XML final

Finalmente el archivo de log tiene la siguiente estructura:

```
<?xml version="1.0" encoding="UTF-8"?>
<Iteracion>
      <MaxPasajerosBus>20</MaxPasajerosBus>
      …
      <Bus> …
      …
      <Pasajero> …
      …
</Iteracion>
```

Se observa que este formato permite que las etiquetas *Bus* y *Pasajero* se puedan repetir e intercalar indistintamente con los formatos descritos en los apartados anteriores.

La etiqueta `Iteracion` simplemente se debe ignorar en la lectura, actúa como nodo raíz del documento *XML*, es decir todas las demás etiquetas están contenidas obligatoriamente en la raíz del documento. Si en un futuro surge la necesidad de persistir más de una iteración por archivo de log basta con repetir la estructura del nodo `Iteracion` y definir un nuevo nodo raíz.

Las pruebas realizadas con el simulador *PublicTransport* generando archivos de log para una simulación de 24 horas con el caso de la ciudad de *Rivera* resultaron en archivos de aproximadamente 38 mega bytes. El tamaño de los archivos generados motivó la decisión de no persistir más de una corrida de simulación por archivo de log.

Finalmente, existe una etiqueta con información adicional sobre el modelo de red vial, dicha etiqueta almacena la cantidad máxima de pasajeros por bus, este valor se incluye como una alternativa para detectar buses llenos llevando la cuenta de pasajeros. Actualmente la *Herramienta de Visualización* la ignora ya que las etiquetas de bus indican en el `Estado` cuando el mismo está lleno.

Bibliografía

[1] A. Mauttone, «Models and algorithms for the optimal design of bus routes in public transportation systems,» [En línea]. Available: http://www.fing.edu.uy/inco/pedeciba/bibliote/tesis/tesisd-mauttone.pdf. [Último acceso: 21 08 2012].

[2] D. Gawenda y H. M. Martínez, «Interfaz para herramienta de planificación de recorridos para transporte público,» 2009.

[3] P. Aldaz y G. De León, «Simulador de transporte público urbano colectivo,» Montevideo, 2010.

[4] J.-P. Rodrigue, C. Comtois y B. Slack, The Geography of Transport Systems, Segunda ed., Routledge, 2009, p. 368.

[5] A. Mauttone, «Optimización de Recorridos y Frecuencias en Sistemas de Transporte Público Urbano Colectivo,» Montevideo, 2005.

[6] A. Mauttone, H. Cancela y M. Ulquhart, «Diseño y optimización de rutas y frecuencias de transporte colectivo urbano, modelos y algoritmos.,» [En línea]. Available: http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0307.pdf. [Último acceso: 29 04 2012].

[7] M. D. H. Guy Desaulniers, «Chapter 2 Public Transit Review,» *Handbooks in Operations Research and Management Science,* vol. 14, pp. 69-127, 2007.

[8] S. Alaggia, A. Mauttone y M. Urquhart, «Curso: Simulación a Eventos Discretos,» Facultad de Ingeniería, Universidad de la República Oriental del Uruguay, [En línea]. Available: http://eva.fing.edu.uy/course/view.php?id=191. [Último acceso: 05 08 2012].

[9] H. Spiess y M. Florian, «Optimal strategies: A new assignment model for transit networks,» de *Transportation Research Part B: Methodological*, Elsevier, 1989, pp. 83-102.

[10] B. Stroustrup, The C++ Programming Language, Tercera ed., Addison–Wesley, 1997, p. 1040.

[11] S. Alaggia, A. Mauttone y M. Urquhart, «EOSimulator Documentation 1.1.1,» [En línea]. Available: http://www.fing.edu.uy/inco/cursos/simulacion/eosim_html/index.html. [Último acceso: 26 06 2012].

[12] E. Gamma, R. Helm, R. Johnson y J. Vlissides, Patrones de diseño. Elementos de software orientado a objetos reutilizable, 1 ed., Madrid: Pearson Education S.A., 2003, p. 364.

[13] C. Matthews, «Simple DirectMedia Layer,» [En línea]. Available: http://www.libsdl.org/. [Último acceso: 22 05 2012].

[14] M. Zeiler, Modeling our World, The ESRI Guide to Geodatabase Design, 1 ed., ESRI Press, 1999, p. 200.

[15] S. L. Pfleeger, Ingeniería de software, teoría y práctica, 1 ed., Buenos Aires: Pearson Education, 2002, p. 759.

[16] C. Larman, UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado., 2 ed., Madrid: Pearson Educación S.A., 2003, p. 624.

[17] O. M. Group, «Object Management Group - UML,» [En línea]. Available: http://www.uml.org/. [Último acceso: 02 08 2012].

[18] INCO, «Introducción a la computación gráfica,» Facultad de Ingeniería, Universidad de la República Oriental del Uruguay, [En línea]. Available: http://www.fing.edu.uy/inco/cursos/compgraf/. [Último acceso: 10 06 2012].

[19] INCO, «Introducción a los sistemas de infomación geografica,» Facultad de Ingeniería, Universidad de la República Oriental del Uruguay, [En línea]. Available: http://www.fing.edu.uy/inco/cursos/sig/. [Último acceso: 21 05 2012].

[20] Microsoft, «DirectX Developer Center,» [En línea]. Available: http://msdn.microsoft.com/en-us/directx/default. [Último acceso: 10 06 2012].

[21] K. Consortium, «OpenGL,» [En línea]. Available: http://www.opengl.org/. [Último acceso: 10 06 2012].

[22] Esri, «ArcGIS API for Silverlight,» Esri, [En línea]. Available: http://www.esri.com/software/arcgis/web-mapping/silverlight.html. [Último acceso: 21 05 2012].

[23] Codeplex, «DotSpatial,» [En línea]. Available: http://dotspatial.codeplex.com/. [Último acceso: 21 05 2012].

[24] Microsoft, «About Silverlight,» [En línea]. Available: http://www.microsoft.com/silverlight/what-is-silverlight/. [Último acceso: 05 06 2012].

[25] Microsoft, «Silverlight Architecture,» [En línea]. Available: http://msdn.microsoft.com/en-us/library/bb404713%28v=vs.95%29. [Último acceso: 10 06 2012].

[26] Esri, «What is ArcGIS Server?,» Esri, [En línea]. Available: http://webhelp.esri.com/arcgisserver/9.2/dotnet/manager/concepts/whats_server.htm. [Último acceso: 21 05 2012].

[27] Codeplex, «ESRI Shapefile Reader,» [En línea]. Available: http://shapefile.codeplex.com/. [Último acceso: 21 05 2012].

[28] Microsoft, «Windows Presentation Foundation,» [En línea]. Available: http://msdn.microsoft.com/es-es/library/ms754130.aspx. [Último acceso: 22 05 2012].

[29] Microsoft, «.Net Framework,» [En línea]. Available: http://www.microsoft.com/net. [Último acceso: 22 05 2012].

[30] ECMA, «C# Language Specification,» 06 2006. [En línea]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf. [Último acceso: 22 05 2012].

[31] Microsoft, «Visual Studio MSDN,» [En línea]. Available: http://msdn.microsoft.com/es-es/vstudio/ff431702. [Último acceso: 22 05 2012].

[32] F. S. Foundation, «GNU Library General Public License, version 2.0,» [En línea]. Available: http://www.gnu.org/licenses/old-licenses/lgpl-2.0.html. [Último acceso: 10 06 2012].

[33] W3C, «Extensible Markup Language (XML) 1.0,» [En línea]. Available: http://www.w3.org/TR/REC-xml/. [Último acceso: 12 06 2012].

[34] Esri, «ESRI Shapefile Technical Description,» ESRI, 07 1998. [En línea]. Available: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf. [Último acceso: 21 05 2012].

[35] M. Fowler, Patterns of Enterprise Application Architecture, 1 ed., Boston: Addison-Wesley, 2002, p. 560.

[36] Microsoft, «Windows.Forms reference documentation,» MSDN, [En línea]. Available: http://msdn.microsoft.com/en-us/library/dd30h2yb.aspx. [Último acceso: 24 06 2012].

[37] MapWindow.org, «MapWinGIS Activex Control,» [En línea]. Available: http://mapwingis.codeplex.com/. [Último acceso: 24 06 2012].

[38] C. Burrows, «New C# Features in the .NET Framework 4,» MSDN, [En línea]. Available: http://msdn.microsoft.com/en-us/magazine/ff796223.aspx. [Último acceso: 25 06 2012].

[39] Microsoft, «Common Language Runtime,» MSDN, [En línea]. Available: http://msdn.microsoft.com/en-us/library/8bs2ecf4.aspx. [Último acceso: 24 06 2012].

[40] Microsoft, «Windows Installer,» MSDN, [En línea]. Available: http://msdn.microsoft.com/es-ES/library/cc185688.aspx. [Último acceso: 25 06 2012].

[41] Microsoft, «Microsoft Visual C++ 2008 SP1 Redistributable Package (x86),» [En línea]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=5582&WT.mc_id=MSCOM_EN_US_DLC_DETAILS_121LSUS007998. [Último acceso: 25 06 2012].

[42] C. Peters, «Minimalist GNU for Windows,» [En línea]. Available: http://www.mingw.org/.

[Último acceso: 26 06 2012].

[43] B. Software, «Bloodshed Dev-C++,» [En línea]. Available: http://www.bloodshed.net/devcpp.html. [Último acceso: 26 06 2012].

[44] S. Lantinga y M. Engdegård, «SDL_image,» [En línea]. Available: http://www.libsdl.org/projects/SDL_image/. [Último acceso: 24 06 2012].

[45] M. Shemanarev, «Anti-Grain Geometry,» [En línea]. Available: http://www.antigrain.com/. [Último acceso: 26 06 2012].

[46] D. Turner, R. Wilhelm y W. Lemberg, «The FreeType Project,» [En línea]. Available: http://www.freetype.org/index2.html. [Último acceso: 27 06 2012].

[47] B. Dawes, D. Abrahams y R. Rivera, «Boost C++ Libraries,» [En línea]. Available: http://www.boost.org/. [Último acceso: 24 06 2012].

[48] J. J. Swain, «Simulation Reloaded Sixth biennial survey of discrete-event simulation software tools that empower users to imagine new systems, and study and compare alternative designs,» *OR/MS Today,* Agosto 2003.

[49] J. J. Swain, «Simulation Software Survey,» Octubre 2011. [En línea]. Available: http://www.orms-today.org/surveys/Simulation/Simulation.html.

[50] J. J. Swain, «Software Survey: Simulation — Back to the future,» Octubre 2011. [En línea]. Available: http://www.orms-today.org/surveys/Simulation/Simulation.html.

[51] Imagine That Inc, «ExtendSim,» [En línea]. Available: http://www.extendsim.com/. [Último acceso: 28 04 2012].

[52] R. A. Q. Tepaz, «Manual Introductorio de utilización de Bloques Básicos ExtendSim 7 Lt,» 2012.

[53] The Alion MA&D Operation, «Micro Saint Sharp Product Tour,» [En línea]. Available: http://www.maad.com/index.pl/product_tour#apps. [Último acceso: 25 04 2012].

[54] SAS Institute Inc., «SAS® Simulation Studio 1.6 User's Guide,» [En línea]. Available: http://support.sas.com/rnd/app/or/SimulationStudio15.html. [Último acceso: 22 03 2012].

[55] ProModel Corporation, «ServiceModel Home,» [En línea]. Available: http://www.promodel.com/products/servicemodel/. [Último acceso: 09 06 2012].

[56] Promodel MX, «Simrunner,» [En línea]. Available: http://www.promodel.com.mx/simrunner.php. [Último acceso: 09 06 2012].

[57] OptTek Systems, Inc, «OptQues,» [En línea]. Available: http://www.opttek.com/OptQuest. [Último acceso: 09 06 2012].

[58] Geer Mountain Software Corporation, «Stat::Fit,» [En línea]. Available: http://www.geerms.com/index.htm. [Último acceso: 09 06 2012].

[59] Legion Limited, «Legion Studio,» [En línea]. Available: http://legion.com/legion-studio. [Último acceso: 15 04 2012].

[60] PTV Group, «VISUM,» [En línea]. Available: http://www.english.ptv.de/software/transportation-planning-traffic-engineering/software-system-solutions/visum/. [Último acceso: 02 04 2012].

[61] ADDLINK, «VISSIM,» [En línea]. Available: http://www.addlink.es/productos.asp?pid=101. [Último acceso: 03 04 2012].

[62] SIAS, «S-Paramics,» [En línea]. Available: http://www.sias.com/ng/sparamicshome/sparamicshome.htm. [Último acceso: 15 05 2012].

[63] «MapWinGIS ActiveX Map and GIS Component,» [En línea]. Available: http://mapwingis.codeplex.com/. [Último acceso: 25 05 2012].

[64] Mozilla, «Mozilla Public License Version 1.1,» [En línea]. Available: https://www.mozilla.org/MPL/1.1/#section-8. [Último acceso: 12 06 2012].

[65] Microsoft, «ActiveX,» [En línea]. Available: http://www.microsoft.com/security/resources/activex-whatis.aspx. [Último acceso: 12 06 2012].

[66] ESRI, «ESRI Developers | Overview,» [En línea]. Available: http://www.esri.com/getting-started/developers/index.html. [Último acceso: 25 05 2012].

[67] ESRI, «ArcGIS API for Silverlight,» [En línea]. Available: http://help.arcgis.com/en/webapi/silverlight/index.html. [Último acceso: 01 06 2012].

[68] ESRI, «ArcGIS Online Map,» [En línea]. Available: http://www.esri.com/software/arcgis/arcgis-online-map-and-geoservices/map-services.html. [Último acceso: 01 06 2012].

[69] ESRI, «ArcGIS Online Image Services,» [En línea]. Available: http://www.esri.com/software/arcgis/arcgis-online-map-and-geoservices/image-services.html. [Último acceso: 01 06 2012].

[70] ESRI, «ArcGIS Server REST API,» [En línea]. Available: http://denverdemo.esri.com/ArcGIS/SDK/REST/index.html. [Último acceso: 01 06 2012].

[71] «DotSpatial,» [En línea]. Available: http://dotspatial.codeplex.com/. [Último acceso: 05 06 2012].

[72] Microsoft, [En línea]. Available: http://www.microsoft.com/maps/. [Último acceso: 06 06 2012].

[73] Google, [En línea]. Available: https://developers.google.com/maps/. [Último acceso: 06 06 2012].

[74] «OpenStreetMap,» [En línea]. Available: http://www.openstreetmap.org/. [Último acceso: 04 06 2012].

[75] F. d. Ingeniería, «Introducción a los sistemas de infomación geográfica,» [En línea]. Available: http://www.fing.edu.uy/inco/cursos/sig/contenidoCurso.htm. [Último acceso: 11 11 2011].

[76] R. M. Davies y . R. M. O'Keefe, Simulation Modelling with PASCAL, Primera ed., Prentice-Hall, 1989, p. 300.

[77] D. Russel, «A Simple C++ Argument Parser,» Stanford University, [En línea]. Available: http://graphics.stanford.edu/~drussel/Argument_helper/. [Último acceso: 05 08 2012].

[78] F. V. Berghen, «Small, simple, cross-platform, free and fast C++ XML Parser,» [En línea]. Available: http://www.applied-mathematics.net/tools/xmlParser.html. [Último acceso: 16 06 2012].

[79] ESRI, «Esri - Understanding Our World,» [En línea]. Available: http://www.esri.com/. [Último acceso: 22 05 2012].

[80] INCONTROL, «ED Transport - The Transport and Logistics Simulator,» [En línea]. Available: http://www.incontrolsim.com/en/ed-transport/enterprise-dynamics-transport.html. [Último acceso: 02 05 2012].

[81] ESRI, «ArcGIS Extension for the Google Maps API,» [En línea]. Available: http://help.arcgis.com/en/webapi/javascript/gmaps/index.html. [Último acceso: 29 05 2012].