UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Simulación de Control de Trenes Mediante Agentes JADE
Autores: Franco Eduardo Randazzo
Director: Fernando Gustavo Tinnetti
Codirector: 2
Asesor profesional: 2
Carrera: Licenciatura en Informática

Resumen

Los sistemas basados en CBTC (IEEE 1474) son aquellos que permiten realizar un seguimiento y control de las formaciones ferroviarias basados en el conocimiento de la posición de los trenes con alta disponibilidad y gran precisión brindando al conjunto del sistema un mayor grado de seguridad que los sistemas convencionales de señalización. A través de un modelo de agentes de software implementados en JADE, se construyó un prototipo de simulador inicial con la capacidad de representar detalladamente la cinemática del ambiente y proveer un alto grado de control a las formaciones que se simulan por medio de controladores diseñados e implementados para dicho propósito haciendo uso del sistema métrico decimal. El presente modelo es un ambiente distribuido, capaz de ser ejecutado en múltiples plataformas por estar desarrollado en JAVA. Se cuenta con la representación de formaciones ferroviarias, las cuales pueden desplazarse a distintas velocidades y dos modos de operación, cantón fijo o cantón móvil, siendo este último el que realiza un uso eficiente del circuito de vía, permitiendo que diversas unidades circulen con seguridad a distancias reducidas. CBTC es normalmente implementado en hardware, siendo este trabajo una visión innovadora basada en software.

Palabras Claves

Simulación, control, Agentes de software, JADE, JAVA, Sistema de tiempo real, Concurrencia, Trenes, Ferroviario, Simulador, CBTC, IEEE 1474, Communications-Based Trains Control, Java Agent DEvelopment, Cinemática, Sistema métrico decimal, Innovación.

Trabajos Realizados

Se realizó un estudio amplio sobre CBTC y su alcance.
Se diseñó un modelo basado en agentes móviles con la capacidad de simular un entorno de control de formaciones ferroviarias.
Se describió el modelo, la interacción que existe entre los agentes, que comportamientos presentan y la arquitectura propuesta.
Se cuantifico el volumen que supone el uso de la red de datos para el prototipo propuesto.
Se analizó el modelo con datos obtenidos a partir del sistema real y para luego demostrar la capacidad efectiva de control alcanzado.

Conclusiones

Se presentó un prototipo inicial de un simulador de control de trenes mediante agentes JADE. El mismo es capaz de representar distintos circuitos de vías, con un único sentido de circulación en un entorno distribuido. El prototipo utiliza principios de cinemática para simular el movimiento de las formaciones y llevar a cabo el control. El funcionamiento se basa en el conocimiento constante de la posición de los trenes con gran precisión. El prototipo es un modelo abierto, con capacidad de ser expandido a futuro.

Trabajos Futuros

Desarrollar una interface para el prototipo para que expanda el tipo de representaciones a simular, agregando señales, mayor detalle de accidentes geográficos, barreras, etc.
Crear una capa de seguridad que provea a los agentes el servicio de autenticación por medio de credenciales.
Implementar una interfaz gráfica agnóstica al prototipo que permita la división de los datos a visualizar.
Compatibilizar el simulador, aprovechando la capacidad de ejecutarse en múltiples plataformas, con arquitecturas de hardware libre para realizar test dentro de un entorno real modelado.

Fecha de la presentación: Febrero de 2017

Agradecimientos

Este trabajo no podría haberlo concluido, si no contara con un grupo humano y familiar sublime, que me ha brindado un apoyado incondicional a lo largo de mi trayectoria como alumno de Licenciatura en Informática. Silvina, la que compartió a mi lado, prácticamente toda mi carrera, le dejo esta especial mención. Mi madre, Susana, que tanto pujo para que concluya mis estudios ¡Hurra!!! Florencia, Hugo y Carolina, mis hermanos que han estado en buenas y malas. Mis suegros, Adela y Hector, por el consentimiento provisto este tiempo. Que decir de mis primos, Florencio y Pablo, que siempre me han facilitado un hueco en sus hogares.

Me es prácticamente imposible nombrar a todos, desde tíos, primos, familiares, amigos, vecinos y conocidos de ocasión, que de un modo u otro nos facilitan el hecho de sentir Humano. Enérgicamente, gracias.

Tabla de contenido

Índice de Ilustraciones

Índice de Ecuaciones

Índice de Tablas

1 INTRODUCCIÓN

Los términos agentes, simulación, control, trenes, JADE, serán empleados para referirse al presente proyecto informático. Dicho proyecto consistirá en el desarrollo e implementación de un prototipo de "simulador de control de formaciones ferroviarias" referido a un posible problema del mundo real. De acuerdo a la magnitud del problema, el simulador solo se centrará en el control de los trenes que circundan un circuito de vía, propiciando la seguridad desde la distancia entre las formaciones. ¿De qué modo? Mediante agentes de software desarrollados e implementados en la arquitectura de desarrollo provista por JADE.

1.1 MOTIVACIÓN

Los sistemas de tiempo real [Book8] de control automático de componentes móviles como los son automóviles, robots, sistemas de transporte, etc. están en boga. En gran parte, gracias a los avances y desarrollos de las grandes empresas. Estas nos muestran cómo un automóvil se conduce solo, un drone se traslada y realiza una acción determinada posiblemente en un lugar muy distante. Haciéndonos pensar en cuántos beneficios podría dejarnos como sociedad. En cualquier simple blog [Link1] de IT (Information Technology) nos muestran todos estos avances realizados por diminutos componentes que poseen una precisión mucho mayor en términos relativos a lo que se podía realizar algunos años atrás.

Por otro lado, existen soluciones que se encuentran "excluidas" de los sistemas informáticos. Excluidas no, porque les falte una solución informática posible, sino porque al frecuentarlas a diario las asumimos tal y como son. Pero si las analizamos en profundidad, se encuadran en problemas capaces de obtener una solución informática con un ámbito definido correctamente. Unas de estas soluciones es el sistema de control de ferrocarriles. Este sistema de control posee un ámbito bien definido y es denominado como Control de Trenes Basado en Comunicaciones (CBTC, por sus siglas en inglés Communications-Based Train Control) [Book2].

El sistema de Control de Trenes Basados en Comunicaciones es un entorno en crecimiento. Las primeras aproximaciones datan de principios de los años '90 cuando se comenzó a experimentar con la comunicación basada en ondas de radio [Book4]. Pasaron los años y los desarrollos de redes de alta velocidad, los sistemas de control y toma de datos por hardware incrementaron sus capacidades de procesamiento. Los servidores dedicados ya no eran

máquinas económicamente inaccesibles, lo que permitieron que en el año 2003 vieran la luz este tipo de plataformas en proyectos concretos como el que la empresa Bombardier llevó a cabo en el aeropuerto internacional de San Francisco [Link3].

En cierta forma, este proyecto surge con el interés de aportar tecnología y una herramienta innovadora a un sector que a nivel mundial sigue generando distintos episodios trágicos, mostrados con una frecuencia regular. Aprovechando el entorno de procesamiento distribuido y con diferentes posibilidades para comunicar y sincronizar entidades que provee JADE [Link12] se puede intentar en el avance del aporte mencionado antes. Toda la información que se puede obtener en cuanto a control ferroviario es bastante escasa y este trabajo se presenta como el puntapié inicial del tema en post de la complejidad que supone. En cuanto a contar con un simulador de nivel inicial, al cual luego, se le podrán incorporar gradualmente los múltiples objetos que modelen distintos detalles técnicos que rodean el problema, permitirá obtener datos e información valiosa para procesar soluciones a los pormenores que llevaron a los problemas que se han visto reflejados en las noticias de accidentes ferroviarios, puesto que el software puede testearse y actualizarse.

1.2 OBJETIVO

Este trabajo tiene como fin modelar y simular un sistema innovador de control de formaciones ferroviarias desarrollado en software. El modelado de la solución se fundamenta en el paradigma que propone la programación para sistemas distribuidos. El modelo cuenta con elementos independientes tanto físicamente como en funcionalidad, trenes y controladores. Donde cada una de las entidades que modelen los trenes serán coordinadas y supervisadas por controladores a través de canales de comunicación. Estos controladores procederán a realizar un seguimiento exhaustivo de las formaciones móviles dentro de tramos de vías férreas pre-asignados.

Como base tecnológica para el desarrollo, se empleó la arquitectura de agentes impulsada por JADE (Java Agent DEvelopment Framework). El sistema obtendrá los requerimientos y principios del estándar del Institute of Electrical and Electronic Engineers, IEEE 1474 para los sistemas de Control de Trenes Basados en Comunicaciones (CBTC, Communications-Based Train Control). El mencionado estándar es un sistema compuesto por formaciones ferroviarias y controladores, donde los controladores mantienen información enviada desde los trenes y estos responden a estímulos ordenados por los controladores.

El estándar IEEE 1474 es un desarrollo técnico basado en el control de formaciones ferroviarias mediante la comunicación bidireccional de información entre el equipamiento del tren y el equipamiento de la vía para gestionar el tráfico. El control del sistema es proporcionado por hardware especifico que implementa la comunicación continua y permite la alta capacidad de envío y recepción de datos entre el tren y el control de vía. Propone cuatro (4) niveles de automatización, que van desde el mero seguimiento de las formaciones hasta el control completo de las mismas. El desarrollo de esta norma en su totalidad, supera el alcance de la propuesta de esta tesina de grado, por consiguiente se propone recortar el área a conjunto de aspectos de la IEEE 1474 mediante los recursos provistos por los agentes y la información obtenida de la norma que encontremos disponible y libre para su uso. Todo ello, supondrá una innovación en la materia, puesto que actualmente, CBTC es un producto específico de hardware, y el enfoque de este trabajo es meramente el software de control.

Los agentes por sí mismos, como entidades autónomas, son capaces de seleccionar tareas, modificar las prioridades de las mismas, comportarse enfocados en un objetivo y tomar decisiones sin la intervención humana. Su persistencia, habilita a estos agentes a estar en continua ejecución como a seleccionar el comportamiento a realizar. Estos agentes, debido a que cuentan con la capacidad de relacionarse (sociabilizar) a través de mecanismos de comunicación, interactúan y realizan tareas coordinados. A su vez, brindan la posibilidad de colaborar en la culminación del objetivo que se persigue. Además, tienen la capacidad de reaccionar a estímulos del entorno en el que operan y responder a estos estímulos adecuadamente.

JADE como arquitectura de soporte de agentes nos proporciona todas las ventajas conocidas del lenguaje Java (en el que se basa ésta plataforma), tanto la capacidad de Java de poder ejecutarse en un gran espectro de dispositivos de uso específico como dispositivos multipropósito. Esto nos permite pensar en soluciones de bajo costo ante sistemas electrónicos dedicados a soluciones con requerimientos similares. Al tener soluciones con diversas cuantías, podemos optar por tener sistemas redundantes de control para cada procedimiento. Esta redundancia de control proveerá una mayor seguridad del sistema, otorgando fiabilidad y robustez a la solución, disminuyendo la posibilidad de que ocurran fallos imprevistos.

A su vez, los agentes cuentan con la habilidad de la movilidad. Esta característica otorga a un agente de software la posibilidad de migrar o realizar una copia de sí mismo (clon) dentro de la arquitectura JADE de un contenedor a otro cuando el agente lo requiera. Hacer uso de esta característica, permite construir la simulación por sectores con total independencia entre un sector y otro. Como así también, la interacción con sistemas o módulos construidos por

otros desarrolladores en base a protocolos preestablecidos. La movilidad soporta por JADE, no solamente proporciona que el código se traslade de un nodo a otro, sino que, además, su estado de ejecución no se pierde y puede continuar con su ejecución donde fue interrumpido.

La arquitectura de JADE facilita la utilización de canales de comunicación entre agentes por medio de mecanismo de conexión punto a punto. Para ello JADE propone un lenguaje de comunicación, FIPA-ACL [Link9]. Un mensaje FIPA-ACL cuenta con distintos tipos de actos comunicativos (acción que realiza el mensaje). Estos actos van desde la aceptación de una propuesta recibida previamente hasta la intención persistente de notificar al emisor de un determinado valor, y volver a notificarle cada vez que dicho valor cambie, pasando por actos como informar que no se entendió el mensaje o preguntarle a un agente si determinada proposición se cumplió.

1.3 ESTRUCTURA DE LA TESINA

Antes de comenzar con el presente trabajo, se realiza una breve descripción de las secciones/capítulos para que el lector pueda hacerse una visión global del documento.

El capítulo 2, es una revisión de las distintas tecnologías utilizadas como así también, de lineamientos que rigen para el sector ferroviario. En una primera etapa, se mencionan los sistemas de tiempo real, sistemas distribuidos y software concurrente. Luego se detalla que son los agentes de software y el porqué de su elección. Que es una simulación y que función cumple. Que sistemas simulados con agentes encontramos para luego compararlos con el presente proyecto.

Por último, se realiza un breve reseña sobre la arquitectura de JADE, para continuar con una introducción a los sistemas convencionales de señalización para ferrocarriles y para finalizar, que es CBTC y que propone.

En el capítulo 3, se explican los aspectos funcionales y técnicos del simulador. Que características del sistema ferroviario son de interés para el modelo propuesto para el prototipo de simulador, y el porqué de las características elegidas para el mismo. Se indica que rol desempeña cada agente junto con la interacción que tendrán entre ellos y la introducción de la cinemática al simulador.

También se indica cual es el formato que tiene el prototipo y como está distribuido en software. Y, por último, una descripción de la arquitectura

propuesta para el simulador, junto con el análisis de la información que debe circular entre los agentes para proporcionar la posición de las formaciones con alta precisión.

El capítulo siguiente, el número 4, se centra en la interfaz gráfica. Se detalla la información que será mostrada en la pantalla por medio de su agente. A si mismo, se detalla que funcionalidades otorga al usuario, como la detención de la simulación o la generación del archivo de la traza simulada y el formato elegido para el mismo.

Como se representan gráficamente los agentes que simulan los trenes y los controladores junto con los correspondientes paneles de control, al igual que, las zonas de detención o las estaciones de stop.

El en capítulo 5, se detalla la implementación del prototipo, desde el agente que grafica la interfaz, pasando por el controlador y finalizando en el agente que representa el tren. Conforme a que está constituida la librería que comparten los trenes y los controladores.

Se explica de qué modo se implementó la cinemática en el simulador y que formato tienen los objetos que se envían en cada evento determinado. Que comportamientos tienen definidos los controladores como los trenes.

Ya en el capítulo 6, se presentan dos fuentes distintas. Una fuente, es la que se utiliza para convalidar el modelo propuesto con el modelo real. La restante, es para realizar una valoración de la capacidad que tiene el prototipo para controlar las formaciones simuladas.

En séptimo capítulo, se arriba a la conclusión, dejando una breve descripción de posibles trabajos para continuar en un futuro y se detalla que problemas se tuvieron en la realización del actual trabajo.

2 ESTADO DEL ARTE

En este capítulo, se analizará qué son los agentes móviles de software. Qué ventajas tiene el uso de agentes de software en el desarrollo de software distribuido. Por su parte, qué permite JADE como plataforma para el desarrollo de agentes móviles. Se dará una descripción de los sistemas de tiempo real y se explicará porque este simulador se encuentra bajo tal descripción.

Asimismo, se hará una comparación de sistemas que utilizan como herramientas principal agentes de software, que persiguen como objetivos simular problemas del mundo real. También se abordará una explicación de que es una simulación y las diferencias que tiene con otros sistemas teóricos matemáticos. Y, por último, estudiar el entorno ferroviario y su estado general en el marco global.

2.1 SISTEMA DE TIEMPO REAL, CON PROGRAMACIÓN DISTRIBUIDA Y CONCURRENTE

El papel de la concurrencia [Book9] para la generación de datos y la toma de decisiones cumple un rol muy importante para acelerar los tiempos de procesamiento y disminuir los tiempos de respuestas de un software. Por ello, es que se desea utilizar plenamente el procesador. Las velocidades de los procesadores modernos son muy superiores a los dispositivos de entrada y salida con los que el sistema debe interactuar. La concurrencia permite a los programadores solapar actividades con otras que se ejecutan mientras el procesador está esperando a que se complete una actividad de entrada/salida.

Para poder censar la información generada por distintos elementos de procesamiento distribuidos geográficamente, se debe permitir que más de un procesador resuelva un problema. Un programa secuencial sólo puede ser ejecutado por un procesador (a menos que el compilador haya transformado el programa secuencial en uno concurrente). Un programa concurrente es capaz de explotar el paralelismo verdadero y obtener así una ejecución más rápida. Si a esto se suma que, la infraestructura de trabajo y los recursos no se encuentran en un mismo edificio o en un entorno cercano, se debe distribuir el procesamiento de la información entre los diferentes recursos.

Para modelar el paralelismo y la distribución de la información en el mundo real, los sistemas de tiempo real [Book8] y embebidos tienen que

controlar e interactuar con entidades del mundo real (robots, cintas transportadoras, etc.) que se encuentran distribuidos geográficamente y que son inherentemente paralelas. Reflejar la naturaleza paralela/distribuida de los sistemas modelados en las estructuras de los programas hace que las aplicaciones sean más legibles, menos costosas de mantener y confiables.

2.2 AGENTES DE SOFTWARE

¿Qué es un agente? En la bibliografía podemos encontrar varias definiciones de lo que es un agente de software. Algunas destacan ciertas capacidades o prestaciones por encima de otras.

Según la FIPA (Foundation for Intelligent Physical Agents) [Link9] tenemos que: “Un agente es una entidad de software encapsulado con su propio estado, conducta, hilo de control y la habilidad para interactuar y comunicarse con otras entidades (gente, otros agentes o sistemas legados).”; podemos ver que en esta definición se destaca la capacidad de un agente que no solo se comunica con el medio en el cual se encuentra, sino que además con otros agentes del sistema y entidades externas como usuarios.

Otra posible definición inherente al trabajo nos dice lo siguiente, “Un agente es cualquier cosa que puede ver en su entorno a través de sensores y actuar en su entorno a través de generadores de acciones” [Book5]; entonces esta definición describe una de las características de un agente computacional, que es poder percibir y actuar en un entorno determinado.

Entre otras definiciones también encontramos: “Un agente es un sistema computacional que está situado en algún ambiente, y que es capaz de actuar autónomamente en dicho ambiente con el fin de cumplir con sus objetivos” [Book6]; donde se destaca que un agente posee autonomía, y es capaz de cumplir con sus objetivos por sus propios medios.

Cuando se destaca que un agente es inteligente, podemos atribuir esta condición a la capacidad de reaccionar, activarse/desactivarse, comunicarse, etc., de los agentes ante los estímulos que le propicia el medio en el que se desarrolla. Dicha “capacidad” podemos dividirla en cuatro (4) características: Autonomía, Sociabilidad, Reactividad y Proactividad.

Decimos que un agente es **autónomo** cuando a partir de sus conocimientos es capaz de alcanzar sus objetivos sin la necesidad de que algún usuario lo guíe.

Los agentes deben ser capaces de **socializar** (comunicarse) y colaborar entre ellos para lograr un objetivo común, como también interactuar con entidades externas al propio sistema, como lo son los usuarios.

Ser **reactivos** al ambiente en el cual se desarrollan percibiendo estímulos tanto del ambiente como los que pueden ser enviados desde el exterior. Estos estímulos condicionan que los agentes puedan cumplir o no, las tareas encomendadas.

Y, por último, tendrán que ser capaces de tomar decisiones (ser **proactivos**), más allá de los estímulos recibidos del medio, con la información que ellos mismos obtengan.

Si bien estas cualidades no son menores para un agente, todas ellas nos dan la definición de agente débil. Para referirnos a agentes inteligentes, tenemos que hablar de que el agente posea los atributos de ser racional, coherente y adaptable, en mayor o menor medida. Un agente será más inteligente cuanto más desarrolladas tenga estas características de inteligencia, racionalidad, coherencia y adaptación [Book7].

Cuando nos hallamos ante un sistema en el cual todos los agentes trabajan conjuntamente para resolver problemas que trascienden el área de trabajo o conocimiento de un agente, nos referimos al sistema como un sistema multiagentes [Book31]. En este trabajo, todos los agentes individualmente y más aún en forma colectiva, brindaran colaboración en materia de seguridad a las formaciones ferroviarias, que tengan que emular.

El medio de comunicación es una parte imprescindible para la comunidad de agentes, debido a que, si no contaran con la capacidad de comunicarse, gran parte de sus fundamentos se verían afectados y quedarían degradados a simples procesos. Por todo ello, los agentes cuentan con un lenguaje de comunicación, FIPA-ACL. FIPA-ACL es un lenguaje que está asociado con la arquitectura abierta de FIPA, el cual se basa en los actos del habla. Posee una sintaxis particular y está diseñado para trabajar con cualquier lenguaje y especificación de ontología.

2.3 MODELO DE SIMULACIÓN

Un modelo de simulación consiste en un conjunto de reglas que definen cómo cambia un sistema a lo largo del tiempo, dado su estado actual. A diferencia de los modelos analíticos, un modelo de simulación no se resuelve, sino que se ejecuta y los cambios de los estados del sistema se pueden observar

en cualquier momento. Esto proporciona una visión dinámica del sistema, en lugar de sólo predecir la salida de un sistema basado en insumos específicos.

La simulación no es una herramienta de toma de decisiones, sino una herramienta de apoyo a la decisión, que permite tomar decisiones mejor informadas. Debido a la complejidad del mundo real, un modelo de simulación sólo puede ser una aproximación del sistema objetivo. La esencia del arte del modelado de simulación es la abstracción y la simplificación. Sólo se deben incluir en el modelo de simulación aquellas características que son importantes para el estudio y análisis del sistema objetivo.

La simulación como modelo teórico de un problema del mundo real es una aplicación de gran potencia. Modelos de sistemas climáticos, como son los que modelan las mareas oceánicas, la predicción de huracanes, etc., son ampliamente conocidos y mencionados a diarios. Estos modelos, más allá de la importancia que ofrecen sus resultados, presentan limitaciones temporales para devolver los valores solicitados y se construyen basados en modelos de sistemas paralelos [Book9]. Los sistemas que podemos simular empleando agentes tienen prestaciones en áreas que evolucionan con requerimientos temporales más abiertos. Dichas áreas suelen ser para modelar sistemas de evoluciones comerciales, como puede ser el sistema bursátil [Link13], sistemas de simulación de evaluación de servicios [Link14], sistemas para simular el manejo de agentes reactivos en fluidos [Link15], etc.

2.4 SIMULACIÓN CON AGENTES DE SOFTWARE

Los agentes como elementos integrados de una comunidad con capacidades como las descriptas (autónomos, sociables, reactivos y proactivos) se convierten en modelos deseables para la simulación en distintas áreas. En contraste, todas sus cualidades quizás sean las mismas que provoquen que su expansión a múltiples áreas no sea masiva como otras tecnologías. No obstante ello, existen distintas empresas y comunidades que proveen frameworks basados en la arquitectura de agentes de software.

Algunas de las plataformas basadas en la arquitectura de agentes móviles son Mobile-C: A Multi-Agent Platform for Mobile C/C++ Agents [Link10], Open source project KATO for PHP and Java developers to write software agents [Link11], JADE Java Agent Developing Framework, an Open Source framework developed by Telecom Italia Labs [Link12], esta última es sobre la cual, se fundamenta el trabajo de la tesina de grado.

Los sistemas multi-agentes (MAS) son una de las tecnologías más interesantes que han surgido en la informática en los últimos 30 años. Como se dice en la página de inicio de ATAL [Book20], uno de los talleres más importantes del mundo a finales de los años 90,

"Agents are autonomous computer programs, capable of independent action in environments that are typically dynamic and unpredictable. Agents have proven to be of interest in many important application areas, such as electronic commerce on the Internet, the control of space probes on missions to the outer planets, the design of user interfaces, to industrial process control."

(Los agentes son programas informáticos autónomos, capaces de actuar independientemente en entornos típicamente dinámicos e impredecibles. Los agentes han demostrado ser de interés en muchas áreas de aplicación importantes, como el comercio electrónico en Internet, el control de sondas espaciales en misiones a los planetas exteriores, el diseño de interfaces de usuario y el control de procesos industriales).

Dentro de la comunidad informática, esta tecnología se utilizó especialmente en la resolución de problemas. Por otro lado, los modelos, arquitecturas de software e implementaciones publicados en este campo podrían ser muy útiles para otra disciplina científica: la simulación social. Los científicos sociales suelen necesitar algunos bancos de pruebas computacionales para probar sus teorías sobre la interacción social o el surgimiento de convenciones, entre otros. Existe una serie de talleres, llamados MABS (Multi-Agent-Based Simulation) [Book31] [Link32] que tienen como objetivo reunir a investigadores de inteligencia artificial, ciencias de la computación y ciencias sociales interesados en el uso de modelos y tecnología multi-agente en la simulación social.

En el trabajo presentado en “Agent-based Planning and Simulation of Combined Rail/Road Transport” [Sim1], se presenta un modelo de simulación del flujo de unidades terminales intermodales (ITU) entre terminales intermodales terrestres. Donde Las terminales intermodales están interconectadas por corredores ferroviarios.

Cada terminal sirve a una zona de captación de usuarios a través de una red de carreteras. El terminal está modelado como un conjunto de plataformas, que son atendidas por una serie de grúas pórtico y elevadores frontales. Dado el calendario de conexiones ferroviarias entre los terminales, un sistema basado en agentes, el Planificador de Transporte Intermodal (ITP), registra a las ITU en los trenes y asigna camiones para entregarlos al terminal de origen y recogerlos en el terminal de destino.

El software de simulación de corredores de ferrocarriles y terminales se ha implementado como un modelo de simulación de eventos discretos, utilizando el MODSIM III como herramienta de desarrollo. El ITP ha sido implementado sobre la base del sistema TELETRUCK. Esta investigación ha sido desarrollada dentro del proyecto PLATFORM, financiado por la Dirección General VII de la Comunidad Europea.

En la simulación con agentes propuesta en “Simulation and evaluation of urban rail transit network based on multi-agent approach” [Sim2] se observa el tránsito ferroviario urbano como un sistema complejo y dinámico, que es complejo de describir en un modelo matemático global para su escala e interacción.

Con el fin de analizar las características espaciales y temporales de la distribución del flujo de pasajeros y evaluar la efectividad de las estrategias de transporte, proponen dar un método nuevo e integral que represente dicho sistema dinámico. Por lo tanto, este estudio busca utilizar el enfoque de simulación para resolver este problema en la red de metro.

Por tal motivo, los autores proponen un modelo de simulación basado en el enfoque multi-agente, que es un método bien adaptado para diseñar sistemas complejos. El modelo incluye las especificidades de los comportamientos de viaje de los pasajeros y tiene en cuenta las interacciones entre viajeros y trenes.

El desarrollo obtuvo una herramienta de simulación de tránsito ferroviario urbano para la verificación de la validez y exactitud de este modelo, utilizando los datos reales de flujo de pasajeros de la red de metro de Beijing para tomar un estudio de caso. Para analizar las características de la distribución del flujo de pasajeros y evaluar bien las estrategias de operación.

Las principales implicaciones que obtuvieron de este trabajo son proporcionar apoyo a la toma de decisiones para la gestión del tráfico, hacer el plan de operación del tren y las medidas de despacho en caso de emergencia. Como resultado de ello, se desarrolló un método nuevo y completo para analizar y evaluar la red de metro, se ha confirmado la exactitud y eficiencia computacional del modelo y se ha encontrado con las necesidades reales de la red a gran escala.

Otro simulador, es el proyecto de investigación “Multi agent based train simulation” [Sim3] que contribuye al proceso de desarrollo de una herramienta, que se basa en la ubicación actual de los trenes en la red ferroviaria holandesa. Permite poder predecir la futura ubicación de los trenes en la red ferroviaria. Esta herramienta es un concepto de la empresa holandesa de desarrollo de software Ordina.

Para esta herramienta, se utiliza un enfoque basado en agentes para simular los escenarios ferroviarios. Para ello, se desarrolla un modelo conceptual del sistema ferroviario. Este modelo se simplifica para poder desarrollar un modelo ejecutable con el que poder planificar una simulación. Además, se explica qué datos deben estar actualizados del sistema real para poder hacer una buena predicción.

Se explica que al inicio de una simulación se deben fijar los valores de los parámetros que normalmente se establecerán mediante observaciones del entorno. Estos valores deben establecerse como si los agentes ya estuvieran activos en el pasado. Se evalúo el modelo simplificado y se obtuvieron recomendaciones para nuevas investigaciones.

2.5 COMPARACIÓN DE SIMULADORES EXISTENTE/PROPUESTOS CON EL PRESENTE PROTOTIPO DE TRABAJO

Hasta aquí, todos los trabajos mencionados, son simuladores basados en arquitecturas distintas de desarrollo de agentes móviles. También, se aprecia que están aplicados/diseñados para la industria del transporte con especial enfoque en la industria ferroviaria. En ellos los agentes se desarrollan/evolucionan generando información y reaccionando a dicha información tomando decisiones.

Las plataformas de desarrollo utilizadas para cada uno de los simuladores mencionados no son las mismas. Por ejemplo, en el trabajo “Agent-based Planning and Simulation of Combined Rail/Road Transport” la plataforma para el desarrollo de los agentes utilizada fue MODSIM III. Este es un lenguaje de simulación de eventos discretos orientado a objetos con extensas bibliotecas de tiempo de ejecución, interfaz gráfica de usuario y herramientas de presentación de resultados, acceso a bases de datos y enlaces con la HLA (High Level Architecture) [Link24].

“Simulation and evaluation of urban rail transit network based on multi-agent approach” utiliza para su desarrollo c#, mientras que el trabajo propuesto en “Multi agent based train simulation” solo es un prototipo conceptual para el desarrollo de un simulador. En todos los casos se tuvo una visión de múltiples agentes cooperando/trabajando para cumplir un rol común, simular una problemática de la vida real en concordancia con el trabajo presente desarrollado.

Por tanto, como es de esperar, el enfoque al cual apuntan cada uno de los trabajos es muy distinto. En unos de los trabajos, el enfoque está puesto en conocer la posición de las formaciones en el sistema holandés, en otro, se intenta operar la red de Beijing teniendo en cuenta los desplazamientos de los usuarios dentro de la red. Y en el restante, se plantea como enlazar dos medios de transporte, como es el medio de carreteras con el medio ferroviario.

Para concluir, los trabajos mencionados, poseen una cercana relación con el presente trabajo, utilizan agentes de software, simulan problemas del sector ferroviario para obtener información para una futura toma de decisiones, etc., pero descartan como objetivo, controlar el tránsito de las formaciones ferroviarias, el cual, es el enfoque principal de esta tesina de grado.

2.6 JADE (JAVA AGENT DEVELOPMENT ENVIRONMENT)

JADE (Java Agent Development Environment) es una plataforma de software completamente implementada y desarrollada en Java. Como requisito mínimo de sistema, debe contar con la versión 5 de JAVA (el entorno de tiempo de ejecución o el JDK). Su desarrollo y distribución está impulsado por los Laboratorios de Telecom en Italia, titular de los derechos de autor, en código abierto, bajo los términos y condiciones de la licencia LGPL (Lesser General Public License Versión 2). Esta plataforma JADE, facilita el desarrollo de Sistemas Multiagente a través de un framework que cumple con las especificaciones FIPA y una serie de herramientas gráficas para administrar y monitorear la ejecución de los agentes. El objetivo de JADE es simplificar el desarrollo de agentes y a su vez garantizar el cumplimiento del estándar FIPA [Book16].

Un sistema basado en agentes JADE puede distribuirse entre máquinas (que ni siquiera necesitan compartir el mismo sistema operativo) y la configuración puede controlarse a través de una GUI remota. La configuración puede cambiarse incluso en tiempo de ejecución moviendo agentes de una máquina a otra, según sea necesario. JADE está completamente implementado en lenguaje Java.

Hay un gran número de casos en los que este enfoque conduce a la construcción de redes con ventajas significativas con respecto a un enfoque clásico cliente-servidor. Los campos de aplicación en los que los sistemas JADE pueden ser diseñados y desempeñados de manera eficiente son prácticamente todos, desde servicios de Internet dirigidos tanto a consumidores como a personas corporativas, extendiéndose al entorno móvil y cubriendo, con óptimos

resultados también el sector de aplicaciones máquina-máquina. La proactividad de los agentes compañeros y la coordinación de tareas distribuidas son funcionalidades perfectamente adecuadas para las necesidades típicas de redes automatizadas.

En resumen, un paradigma de comunicación directa y sin fisuras entre agentes pares o compañeros, donde todos los elementos tienen las mismas capacidades, las misma posibilidad de iniciar una sesión y actuar independientemente, surge del análisis de la evolución en acto.

JADE es un middleware que apunta a apoyar el desarrollo de aplicaciones que abordan esta evolución como su fundamento es el enfoque de agente inteligente Peer-to-Peer. JADE permite el desarrollo de sistemas de pares capaces de:

- Trabajar de manera proactiva, de acuerdo a las reglas de uso que le den los propietarios.
- comunicarse y negociar con otros, directamente e independientemente de su papel y posición.
- coordinar para resolver problemas complejos de forma distribuida.

2.7 ACTUALIDAD DE LOS SISTEMAS FERROVIARIOS CONVENCIONALES

Los sistemas convencionales de señalización/control de tren [Book2] [Link26] se basan casi exclusivamente en circuitos de vía para detectar la presencia de trenes. Es decir, se proporciona información sobre el estado de la pista por delante a los operadores de trenes, a través de señales en tierra o señales de cabina de tren.

La seguridad en el cumplimiento de las señales se logra mediante procedimientos operativos, paradas de tren automáticas en la vía, o equipo de supervisión de tren vinculado al sistema de frenado del tren. Estos sistemas convencionales son eficaces para proporcionar protección de trenes (su colocación debe planificarse, debido a que los costos impiden su colocación a lo largo de un trayecto indiscriminadamente), pero no son particularmente eficientes para maximizar la utilización de la infraestructura de tránsito ferroviario, como resultado de una serie de limitaciones fundamentales, específicamente:

A) La ubicación de los trenes sólo puede determinarse para la resolución de los circuitos de vía; Si cualquier parte de un circuito de vía está ocupada por un tren, se supondrá que todo el circuito de vía está ocupado por el mismo. Los circuitos de vía se pueden hacer más cortos, pero cada circuito de vía adicional

requiere hardware adicional de vía, por lo que hay un límite económico y práctico para el número de circuitos de vía que se pueden proporcionar. Un tipo de sensor utilizado para este propósito es como el que a continuación se muestra en la Ilustración 2-1.

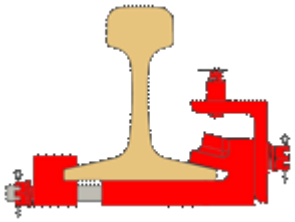


Ilustración 2-1 Detector para rueda ferroviaria Honeywell [Link28].

B) La información que se puede proporcionar a un tren se encuentra limitada a un número reducido de aspectos de la señal en vía, o un sucinto número de códigos de velocidad en un sistema de señal de cabina.

Ilustración 2-2 Sistemas de seguridad en redes ferroviarias españolas.

C) En el caso de un sistema de señalización de vía con paradas automáticas del tren, pero sin señalización continua de la cabina, la aplicación es intermitente.

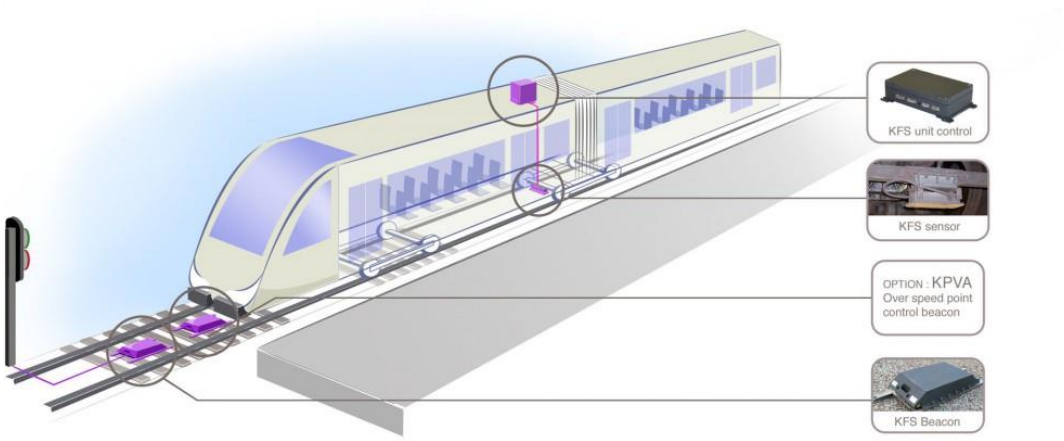


Ilustración 2-3 - Modelo de control de parada automática ferroviario KFS SIL2.

2.8 Sistemas de Control de Trenes Basado en Comunicaciones

Los sistemas de control de trenes basados en las comunicaciones (CBTC) superan las limitaciones fundamentales de los sistemas convencionales de circuitos de vía antes mencionadas y, por lo tanto, permiten una utilización más eficaz de la infraestructura de tránsito.

Esto se logra, por ejemplo, al permitir que los trenes operen con mayor seguridad de circulación con distancias de cercanías muy inferiores, permitiendo mayor flexibilidad y mayor precisión en el control del tren, proporcionando seguridad continua de separación entre las formaciones y proveyendo al tren protección contra el exceso de velocidad.

Beneficios adicionales de la tecnología CBTC incluyen el soporte económico de las operaciones de trenes automáticos (tanto en la línea principal como en los talleres de mantenimiento), mejora de la confiabilidad y la reducción de los costos de mantenimiento, a través de una reducción del equipamiento en tierra e información de diagnóstico en tiempo real.

2.8.1 Las características básicas de un sistema de CBTC incluyen lo siguiente:

A) Determinación de la ubicación del tren, con un alto grado de precisión, independiente de los circuitos de vía.

B) Una red de comunicaciones de datos distribuida en la geografía del sistema ferroviario y con comunicación continua con los trenes dentro de la vía y de vía con los controladores en tierra, que permiten la transferencia de información de control y de estado significativamente más controlada de lo que es posible con los sistemas convencionales.

C) Procesadores vitales de los datos del tren que es alcanzado y del tren alcanzante para poder generar y establecer qué datos de estado y de control deben proveerse para proporcionar protección automática continua de los trenes (ATP). También pueden proporcionarse funciones de operación automática del tren (ATO) y supervisión automática de los trenes (ATS), según lo requiera la aplicación en particular.

Aunque se reconocen los beneficios de la tecnología CBTC, actualmente no existen estándares independientes que definen el desempeño y los requisitos funcionales que los sistemas CBTC deben satisfacer para lograr un mejor desempeño, disponibilidad, capacitación en la flexibilidad operacional y protección del tren. Esta norma ha sido desarrollada para tratar esto.

Entonces, ¿Qué es CBTC? CBTC es el acrónimo de Communications-Based Train Control (Control de Trenes Basado en las Comunicaciones). Concretamente es una norma propuesta por la IEEE (The Institute of Electrical and Electronics Engineers, Instituto de Ingenieros Eléctricos y Electrónicos), la cual propone un sistema seguro y que su desempeño debe suministrar un sistema ferroviario con base en las comunicaciones.

2.8.2 Que establece el estándar

El estándar establece los requisitos de desempeño y funcionales para un sistema de control de trenes basado en las comunicaciones (CBTC). Los siguientes tips nos darán un rápido resumen:

1) El Alcance de la norma, establece un conjunto de requisitos de desempeño y funcionales necesarios para mejorar el rendimiento, disponibilidad, operaciones y la protección de los trenes mediante un sistema CBTC.

2) Su Propósito es definir y unificar en un único estándar, el conjunto de normas que describen requisitos de desempeño, funcionales de forma independiente que deben cumplir sistemas idénticos a CBTC. Esto mejorará el rendimiento, la disponibilidad, las operaciones y la protección de los trenes, facilitando la creación de nuevas aplicaciones sobre CBTC.
3) Las características principales de un sistema CBTC incluyen:
    a) poder determinar la posición de la formación ferroviaria en alta resolución, independientemente del trayecto en el que se encuentre;
    b) alta disponibilidad de comunicación de los datos bidireccional y continuamente entre los trenes y los controladores;
    c) capacidad de procesamiento en las unidades ferroviarias como en los circuitos de vías que desempeñan las funciones vitales del sistema global.
4) Esta norma reconoce que son posibles diferentes configuraciones de sistemas un CBTC, dependiendo de la aplicación específica. Por ejemplo, un sistema CBTC puede:
    a) proporcionar funciones ATP solamente, sin funciones ATO o ATS;
    b) proporcionar funciones ATP, así como ciertas funciones ATO o ATS, según sea necesario para satisfacer las necesidades operativas de la aplicación específica;
    c) ser el único sistema de control del tren en una aplicación dada, o puede utilizarse junto con otros sistemas auxiliares de vía.
5) Un sistema CBTC deberá ser capaz de soportar una variedad de configuraciones como:
    a) Trenes unidireccionales de longitud fija compuestos por una o más unidades operativas básicas;
    b) Trenes bidireccionales de longitud fija compuestos por una o más unidades operativas básicas;
    c) Trenes unidireccionales de longitud variable;
    d) Trenes bidireccionales de longitud variable.

    El sistema CBTC debe ser capaz de soportar una flota mixta de trenes, donde trenes específicos y/o clases de trenes, tienen diferentes características de desempeño.

6) Operaciones de trenes en modo a prueba de fallas:

Poder seguir operando el sistema con fallas de manera segura es un requisito operativo fundamental. Las formaciones deben continuar funcionando de forma segura en caso de fallo de los equipos CBTC y/o de la comunicación de datos. Posiblemente con velocidades de operación reducidas o aumentando los recorridos operativos en comparación con las operaciones normales de los trenes.

Como consecuencia de esto, CBTC prevé que se diseñará un sistema de CBTC de apoyo para los modos degradados de operación en caso de fallo, y para continuar proporcionando la protección automática del tren con una dependencia mínima en el cumplimiento de los procedimientos operativos. Esto se logrará a través de elementos funcionales del propio sistema CBTC, un sistema auxiliar de vía (si lo especifica la autoridad competente) o una combinación de ambos sistemas.

Un plan de contingencia, basado en el análisis de fallas y los procedimientos operativos, identificará los modos operativos en que los trenes aprovecharán los modos degradados de operación y capacidades de recuperación del sistema CBTC.

Concretamente, las operaciones de trenes en modo a prueba de fallas en el territorio de la CBTC se ocupan de las fallas del sistema CBTC que afecten:

A) Todos los trenes que operen dentro de un área particular de control;

B) Un tren en particular que opere dentro de cualquier área de control.

Es importante destacar, que este estándar, en la actualidad es un área donde los equipos que se utilizan para su implantación, son hardware específico y propietario.

3 MODELADO DEL SIMULADOR DE CONTROL DE TRENES MEDIANTE AGENTES JADE

En este capítulo, se especifican los aspectos funcionales y técnicos del simulador de control de trenes mediante agentes JADE. Dada su evolución en tiempo real, se detalla la interacción entre los distintos tipos de agentes (formaciones ferroviarias, controladores ferroviarios y el centro de recepción y visualización de datos), como así también las cuestiones generales de funcionamiento del simulador.

3.1 CARACTERÍSTICAS DE UN SISTEMA FERROVIARIO

Las características de un sistema ferroviario no son únicas. Cada país posee una legislación específica cuando se trata la materia del transporte. Construir una solución integradora que reúna todos los requisitos no es un enfoque posible. Se puede sesgar un poco la visión y obtener valores de referencia, parámetros de funcionamiento, características especiales, etc., que, sin ser estándar en todos los países, permita para realizar una simulación válida del problema.

En el documento Especificaciones Técnicas Básicas Equipamiento Electromecánico y Material Rodante Tramo: Villa El Salvador – AV. GRAU Tomo 2 septiembre 2007 [Link25] se ofrece una vista de un sistema ferroviario para Perú, con un gran nivel de detalles técnicos, que puede ser tomado como referencia para la implementación del simulador.

En este documento se especifica que la característica de aceleración [25, pág. 12] de una formación ferroviaria debe ser de 0,6 m/s^2 a 1,0 m/s^2 lo que, para el trabajo propuesto se establecerá en 1 m/s^2. También se describe cual es el valor para que el tren se detenga, al igual que la aceleración, lo que se establece en -1 m/s^2. A su vez, la longitud de la estación cuenta con su correspondiente especificación, siendo esta de 120 m [25, pág. 6].

La cantidad de formaciones que pueden circular en una línea ferroviaria para un circuito puntual convencional dependerá del sistema de señalización y control. Estos sistemas se basan en la presencia del tren en determinados puntos estratégicos que al detectar la presencia de una formación activan/desactivan señales indicadoras de tal acción. La granularidad en la que se fragmente el circuito de vía va a permitir mayor o menor cantidad de

formaciones en ella. Por contrapartida, a menor granularidad del circuito de vía, mayor es el costo de equipamiento necesario para la señalización o control.

3.2 CARACTERÍSTICAS DE LOS SIMULADORES

Un modelo de simulación consiste en un conjunto de reglas que definen cómo cambian los estados de un sistema a lo largo del tiempo, dado su estado actual. No se tiene una única fórmula analítica para resolver la simulación en su conjunto, porque en cada uno de sus estados, los datos son evaluados y valorados en base al estado actual. La dinámica con que el simulador actualiza sus estados conlleva a que, en función de los datos de entrada, no se permita realizar una predicción de la salida como en otros sistemas o en una función matemática.

Por ello, las simulaciones ayudan en la decisión de elecciones al momento de optar entre soluciones muy similares o antagonistas. Para que la simulación no se vea afectada por la complejidad del entorno, el modelado de la misma solo puede proveer una aproximación del problema original, evitando modelar situaciones o procesos que pueden alterar la información resultante. Una simulación conlleva dos propósitos, o bien para comprender mejor el funcionamiento de un sistema en particular, o bien para hacer predicciones sobre el rendimiento de dicho sistema. Puede ser visto como un whiteroom artificial que permite a uno adquirir conocimiento, pero también poder probar nuevas técnicas y prácticas sin interrumpir la rutina diaria del problema a analizar. La idea es, si la teoría que se ha enmarcado sobre el sistema analizado se mantiene, y si esta teoría se ha traducido adecuadamente en un modelo de computadora, esto facilitará las respuestas de algunas de las siguientes preguntas:

- ¿Qué tipo de comportamiento se puede esperar bajo combinaciones arbitrarias de parámetros y condiciones iniciales?
- ¿Qué tipo de comportamiento mostrará un sistema determinado en el futuro?
- ¿Qué estado alcanzará el sistema objetivo en el futuro?

Una simulación del modelo propuesto en el actual trabajo, consiste en la definición (configuración de los parámetros iniciales) de los controladores de los trenes y de la interfaz de visualización de los datos. Dicha interfaz, una vez disponible en ejecución, se le enlazaran los controladores.

Los controladores especifican un trayecto de la vía a ser recorrida por las formaciones en un solo sentido. Además, contarán con una estación en su extremo más alejado, independientemente de la longitud del circuito de vía, pudiendo precisar estaciones en la parte interior del trayecto. En caso de ser el primer controlador, implícitamente contará con una estación inicial, la cual no es visualizada y no debe ser especificada en la definición del controlador, puesto que se deduce su ubicación. Cada controlador se enlazará automáticamente con el último controlador definido. Si tal controlador es el primer controlador de la simulación, no tendrá un antecesor. Por lo tanto, toda esta información será enviada y mostrada por la interfaz de visualización de datos.

La información generada como la provista al simulador para su ejecución, deberá ser especificada en el sistema métrico, siendo el metro la unidad para referirnos a las longitudes de los andenes de las estaciones como a la distancia total de un trayecto, o las longitudes que se refieren a la caracterización de la formación. La unidad establecida para la velocidad de la formación es el kilómetro por hora. El tiempo promedio que se desea que una formación se detenga en una estación es en segundos. La definición de una estación simplemente se realizará indicando la distancia a la que se encuentra en metros desde el comienzo del circuito en cuestión. Toda estación, cuenta con una zona de detención de las formaciones, no necesariamente la formación comenzará a detenerse allí. Se recomienda que la longitud a la que se establezca dicha zona, sea una distancia a la que formación circulando a máxima velocidad pueda detenerse o mayor aún.

La secuencia necesaria para que toda la plataforma esté preparada para su ejecución, es la misma que existe para un sistema ferroviario real. Se debe contar con un área donde plasmar el circuito ferroviario, para ello en el simulador, iniciamos la interfaz visual. Se continua por definir un trayecto para que las formaciones se desplacen (instanciar todos los controladores que sean necesarios para simular el trayecto en cuestión) y luego montar las formaciones sobre los rieles correspondientes. Al igual que los trenes, si se instancia un tren primero o por delante de otro, mientras que el tren que fue instanciado primero y se encuentra en la cabecera no comience a operar, el tren que lo secunde no podrá hacerlo.

3.3 INTERACCIÓN DESDE LOS DISTINTOS ACTORES DE LA SIMULACIÓN

La simulación no es computada por un único proceso que organiza y distribuye las tares de la simulación. Esto es así, porque la simulación está

organizada de forma distribuida y por ello, cada uno de los participantes de la simulación cumple un rol o papel específico dentro de la misma. Por ello, a continuación, se realiza una breve descripción de la interacción entre los agentes dependiendo de qué rol cumple en la simulación.

- Desde la Interfaz de Visualización de información, el sistema de simulación es totalmente autónomo. Esta interfaz está compuesta por un agente de software que interactúa con todos los controladores involucrados en la simulación. Su única y especial función es la de poder aglomerar la información que cada uno de los controladores administra. Y luego mostrarla para obtener una comprensión visual de lo que está sucediendo en vivo durante la ejecución de una simulación.
- Los trenes. La concepción de las formaciones para este simulador es que estas sean subordinadas a las decisiones de los controladores. Por tanto, son agentes que reciben órdenes sobre qué tarea realizar y sobre qué tarea NO realizar. El porqué de este enfoque se basa en que, si los controladores y los trenes son capaces de interactuar con un alto grado de dependencia de uno sobre el otro, pero que a su vez el agente dependiente cumple holgadamente su cometido, podemos decir que tenemos un sistema altamente fiable y de gran simplicidad.

  Fiable porque al estar todo el control sobre los controladores, los trenes solo deberán obedecer las órdenes impartidas. De este modo, los trenes tendrán la capacidad solamente de ejecutar instrucciones simples. Lo que evitará la necesidad de chequear o verificar que la ordenes que le son impartidas puedan llevarse a cabo. Por todo ello, la simplicidad se verá expresada en la implementación de los agentes que simulan los trenes.
- Los controladores se encargan de todo. Por ser los encargados de enviar la información para que sea visualizada en la interfaz de visualización, controlan cual es la información que será mostrada. Y desde el punto meramente de control que aborda este trabajo, controlan a las formaciones mediante el análisis de la información que los trenes les suministran en cada momento.

  Los controladores deben poder detectar entre todas las formaciones que estén disponibles en el circuito los siguientes eventos:

    - puede ocurrir que una formación esté detenida en una estación o en un trayecto del circuito cualquiera y un tren que viene en alcance está operando a una velocidad elevada con el consecuente riesgo de colisión;
    - o que un tren se está desplazando a una velocidad inferior a la que se está moviendo la formación que está dando alcance.

- Si el tren se encuentra detenido:

    - en una estación esperando para poder continuar con su circuito de vía asignado;
    - en un lugar arbitrario del circuito porque delante suyo se encontraba una formación detenida.
- Si el tren se encuentra por delante con una estación:
    - el tren debe comenzar a disminuir la marcha para detenerse en el lugar correspondiente.
- Realizar la acción que corresponda cuando una formación ha llegado al final del circuito:
    - cambiar de controlador, dado que el siguiente circuito de vía lo gestiona el controlador precede;
    - retirar la formación del trayecto que administra porque el tren ha concluido su servicio.

3.4 La Cinemática del Simulador

En particular para este trabajo, será necesaria la comprensión de algunos aspectos teóricos y aplicados que dependen de la Cinemática [Book29] [Book30]. La Cinemática como rama de la física que estudia el movimiento de los cuerpos sólidos sin considerar las causas que lo producen, tiene gran impacto en el desarrollo del simulador. Las formaciones que recorren los circuitos de vías durante su ejecución realizan cálculos que son objeto de estudio de dicha área de la física aplicada. De la misma manera, los controladores, deberán conocer estos principios para desarrollar el control activo de los trenes.

Puntualmente, de la Cinemática se utilizan las ecuaciones que son aplicadas para Movimiento Rectilíneo Uniforme, Movimiento Rectilíneo Uniformemente Acelerado, y sus derivaciones como el cálculo del espacio recorrido, el cálculo de la aceleración y desaceleración, cálculos de velocidades, etc. Todas las fuerzas de rozamiento, de resistencia u otro tipo de fuerzas que integran el mundo real no son evaluadas, puesto que las especificaciones técnicas de las formaciones reales, entregan valores concretos y dentro de los cálculos de esos valores asumieron estos coeficientes.

3.5 Del Simulador

El simulador entendido como herramienta, se desarrolló por completo en JAVA comprendido de cuatros (4) proyectos Java que son, las formaciones

(agent2Train), los controladores (agent2Controller), el vocabulario compartido por todos los agentes (agent2Trainvocabulary) y la interfaz visual (agent2Visual).

En el proyecto agent2Train se encuentra toda la especificación requerida para modelar una formación ferroviaria. En una primera aproximación, se aprecia que los trenes tienen que poder desempeñar tareas como moverse, detenerse, avanzar a una determinada velocidad, entre otras cuestiones más técnicas que se irán detallando conforme se avance en la escritura del trabajo.

El proyecto agent2Controller puede verse como el manager o cerebro del sistema. En él se realiza la toma de decisión de qué formación avanza, se detiene o se demora. A su vez, en qué situaciones puede obtenerse la posibilidad de que dos (2) o más formaciones se aproximen demasiado o se encuentran libres para circular. Qué formaciones están en una parada programada como es una estación. Decidir que una formación necesita realizar el cambio de agente controlador por el siguiente controlador en el trayecto, cuando la formación ferroviaria se encuentran en los límites de control del Agent2Controller.

Agent2TrainVocabulary es la librería que permite que los distintos integrantes del sistema conversen el mismo idioma o protocolo. Y, por último, la interfaz visual, agent2Visual, se encarga de graficar los eventos que acontecen en todo momento del sistema a nivel global. Mostrando los distintos tramos, las estaciones y los inicio y fin del trayecto.

3.6 ARQUITECTURA DEL SIMULADOR

La arquitectura del simulador se compone de una red de interconexión de información, como lo es, una red LAN. Se utilizará un servidor JADE por cada uno de los controladores (agent2Controller) distribuidos y colocados estratégicamente por los usuarios/operarios del simulador a lo largo del trayecto en cuestión. Los agentes (agent2Train) que tienen que controlar las formaciones ferroviarias en movimiento, deberán simular a los trenes que asistan. El monitor global del sistema (agent2Visual), que también es un agente móvil, estará montado sobre la misma red de interconexión.

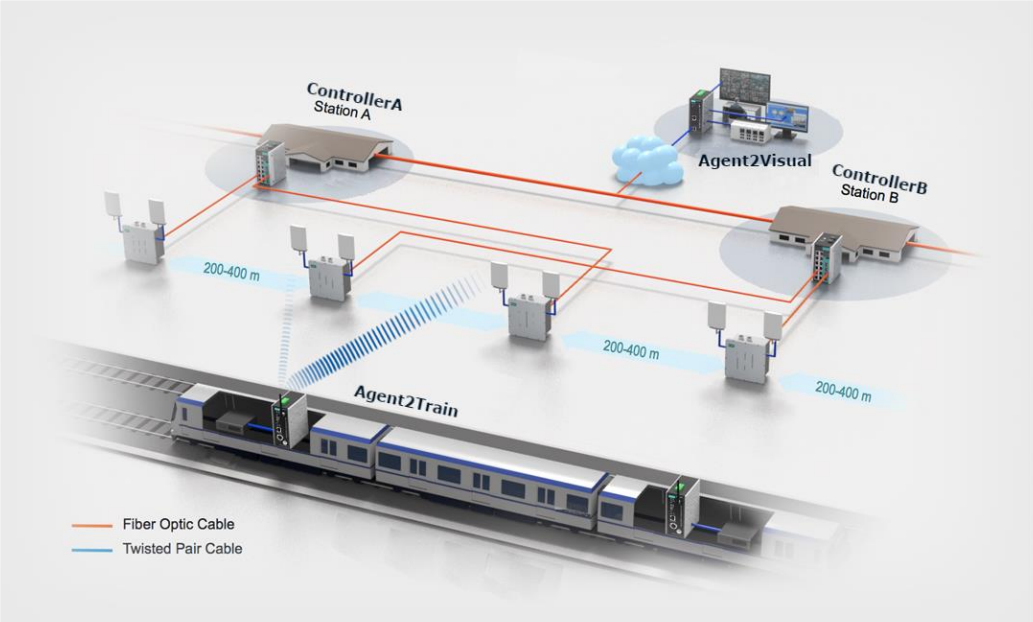


Ilustración 3-1 - Modelo de la arquitectura propuesta.

La arquitectura sobre la que se montó el simulador es como la que se muestra en la Ilustración 3-1. Cada controlador monitorea una sección del trayecto a completar. Uno o más trenes circulando en cada etapa del trayecto en un único sentido, reportando y recibiendo órdenes de un controlador ubicado en cada sección de control por la que transite. En la imagen, se observa un controlador en cada estación, pero puede colocarse más de una estación por controlador. El monitor global del sistema recibe datos de los controladores que le informan el avance de las formaciones. Todo ello montado sobre una red LAN o giga LAN que es acorde con las tasas de transferencia producido por la arquitectura global.

3.7 La tasa de datos durante la simulación

La información que las formaciones envían a los controladores es de vital importancia para el funcionamiento global del sistema. Cualquier impedimento que se anteponga en la comunicación entre el tren y el controlador correspondiente, arrojará un saldo desfavorable. Puesto que al no poder recibir la información requerida (posición de las formaciones, velocidad de las mismas, estado en que se encuentran, etc.), los controladores se encuentran sin la información para llevar a cabo los cálculos de control y seguimiento de las formaciones.

Si se tiene en cuenta que una tasa de mensajes aceptables está entre 7 y 13 mensajes por segundos, por lo tanto, el volumen contenido en cada mensaje no puede ser muy elevado. Quizás, para un sistema que ofrece servicios en la web, estas medidas de datos no repercuten en un inconveniente. Pero, en este caso, por tratarse de un simulador que modela un sistema de transporte, no sería aceptable.

Para lograr una visión más clara de la situación, el problema debe ser visto a través de cálculos, los cuales facilitaran su comprensión. Se tiene una formación que se desplaza a una velocidad media de 100 km/h. Esto indica que estará atravesando en un segundo algo más de 27.7778 metros en dicho instante. Estos datos ponen a prueba constantemente al controlador que debe reaccionar, no sólo a las peticiones de una formación, sino a todas las formaciones que se encuentren en su circuito de administración y control.

La infraestructura de comunicación junto con el hardware que brinden soporte a un esquema como el propuesto contarán con alto grado de disponibilidad. Para este proyecto, una tasa de seguimiento de las formaciones será considerada confiable cuando el registro del avance de la formación para el trayecto asignado comunique valores inferiores a 5 metros. En este simulador esta tasa puede ser condicionada, tanto por la velocidad máxima a la que se desplaza la formación o por la tasa de mensajes seteadas en el timeLatch de los agentes que emulan los trenes.

El simulador envía distintos valores de tamaño de mensajes dependiendo del tipo de mensaje que se envía. Se tiene mensajes que indican el inicio del servicio con un peso de hasta 2.600 Bytes, pero este tipo de mensaje puede ser tomado como excepcional o tope superior, porque sólo se registra cuando las formaciones se registran en el correspondiente controlador. Si en una simulación se tiene 5 controladores, dicha cantidad será la que se envíe un mensaje con ese tamaño. Los mensajes que mayor frecuencia tienen, son los que dan cuenta de la posición de la formación. Estos mensajes tienen un valor de 1.800 Bytes como valor máximo por mensaje.

Con estas tasas de volumen de información, recordar que se necesitan tasa de envío de los mensajes de entre 7 a 13 mensajes por segundo, esto se traducirá en 12.600 Bytes o en 23.400 Bytes dependiendo de las tasas elegidas. Siempre se podrá variar la cantidad de mensajes que se envíen para obtener valores acordes a cada infraestructura. Este volumen de información, solo abarca a una formación con su correspondiente controlador. La información que se da como respuesta desde el controlador al tren se desprecia porque tiene un valor tope de 10 Bytes. Al cabo de transcurrir una hora de simulación, para una tasa de envío de 10 mensajes por segundo, se contará con un volumen de información enviado de 61.8 Megabytes, pero si consideramos enviar a tasas de

15 mensajes por segundos, obtendremos un volumen de 92.7 Megabytes, por tren simulado.

La interfaz visual, no es ajena a la recepción de información. Por este motivo y por constitución del modelo del simulador, el agente visual recibe un duplicado de la información que es recibida por los controladores. Siguiendo con los cálculos realizados, encontramos que la tasa se duplica. Podría suponerse que la información hacia la interfaz visual podría reducirse, pero debido a que la interfaz es el agente que unifica la información de toda la arquitectura para poder realizar el resguardo de la Información generada en la simulación. Volviendo a los valores de la tasa de transmisión de mensajes, estaremos en 14 o 26 mensajes para los valores de 7 o 13 mensajes respectivamente, configurados para la transferencia de los trenes. También, en el volumen de datos, se verá reflejado la misma circunstancia, elevando a 86.5 Megabytes hasta 160.6 Megabytes para los casos descritos en el lapso de una hora.

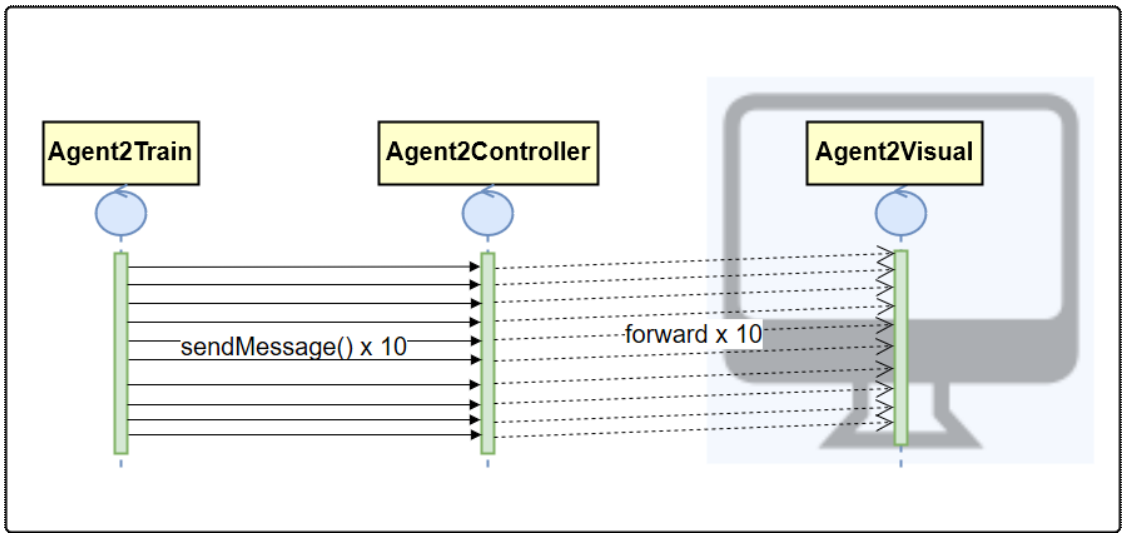


Ilustración 3-2 - Comunicación desde el Agent2Train hasta el Agent2Visual.

4 DETALLES DE LA IMPLEMENTACIÓN DE LA INTERFAZ VISUAL DEL SIMULADOR

Para una mejor compresión del funcionamiento del simulador, sus prestaciones y sus limitaciones, se dividirá esta información en base a qué tarea realiza cada componente. El simulador posee componentes como los controladores, los trenes y el agente que provee la interfaz visual. Este último agente es que a continuación se describe.

Ocurre que una interfaz para este tipo de problemas, no suelen ser de lo más amigables o user friendly interface por sus características técnicas y que necesitan ocultar información que puede producir ambigüedad. Este tipo de paneles técnicos no muestran los trenes o estaciones como cualquier persona ajena a este ámbito puede pretender que se visualice. Por ello, se ha optado un por una interfaz no tan técnica, pero sí, más auto explicativa y en la que se pueda apreciar todo el detalle de la información que se va generando.

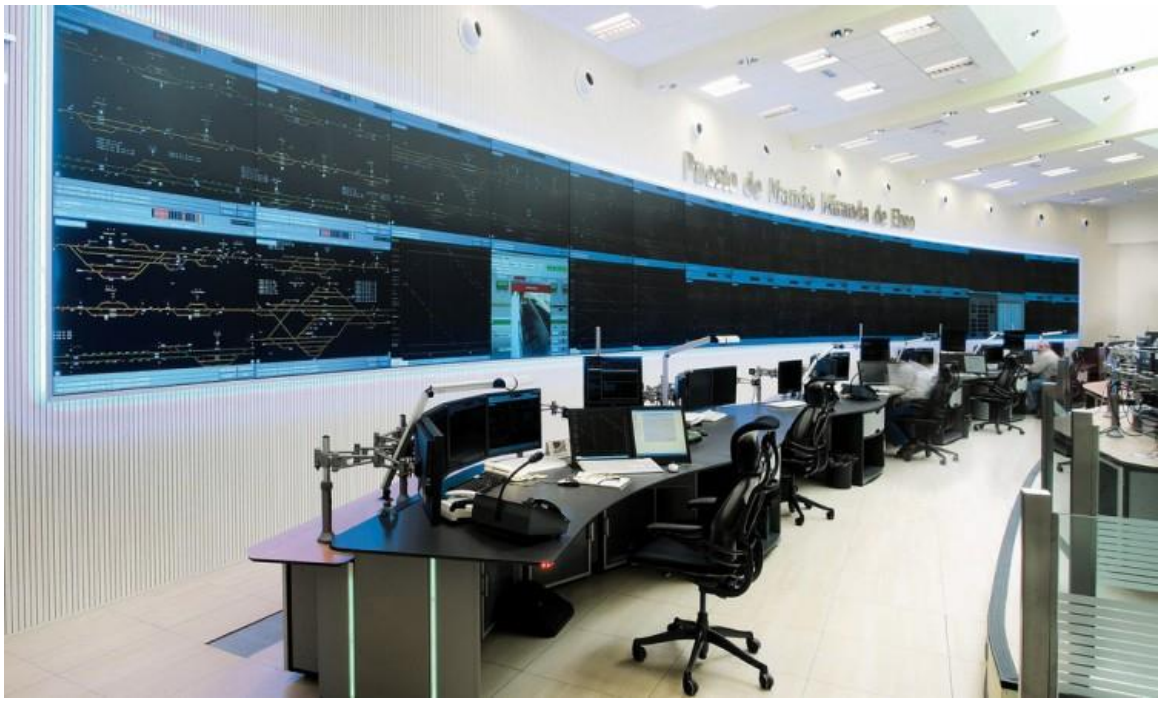


Ilustración 4-1 - Imagen de un panel de control ferroviario de Miranda de Ebro.

4.1 AGENTE DE LA INTERFAZ VISUAL

El agente que provee la funcionalidad de interfaz visual es el agente que inicia el simulador como así también, el entorno de JADE. Este agente, además de mostrar la información que va aconteciendo a medida que evoluciona la simulación, instancia la arquitectura de JADE y da visibilidad a todos los agentes que a continuación se instancian para que puedan conectarse e ir formando la arquitectura correspondiente que se ha optado por simular.

La correspondiente interfaz es muy sencilla, permitiendo que la que información que vaya a ser visualizada se interprete rápida e intuitivamente. Se reitera, que para una persona que esté familiarizada con el entorno ferroviario, puede no estar de acuerdo con la gráfica seleccionada. La ventana de la interfaz gráfica del simulador está dividida en 4 partes. La primera parte de la pantalla de izquierda a derecha, se encuentra un marco en blanco que será el panel donde se irán cargando los circuitos de vías correspondientes de cada controlador. En el margen derecho, se encuentra un panel más pequeño. En este marco se cargarán los datos técnicos y de estado de los controladores y los trenes, información como qué longitud posee un circuito de vía para un controlador o la velocidad a la que circula una formación en un instante dado, la cantidad de trenes que está controlando un controlador, etc.

Las dos (2) partes restantes que quedan por describir, son la sección de funcionalidades o acciones que provee el simulador para el usuario y la sección de notificaciones. Las funcionalidades o acciones que esta primera versión de la interfaz visual provee a un usuario del mismo, es la capacidad de poder realizar un historial de todos los eventos ocurridos durante la simulación que se encuentra en ejecución y la función de detener/continuar (“HALT”) la simulación. En la sección de notificaciones, se muestra la información que resulta de la ejecución de una de las acciones. Luego de realizar un backup se indica si se éste se realizó correctamente u ocurrió un error y lo propio para cuando se ejecuta la detención del simulador, indicando si pudo detenerse la simulación o no.

Ilustración 4-2 - Muestra las cuatro (4) zonas en que se divide la interfaz gráfica.

La funcionalidad HALT permite detener la simulación. Para reanudar la simulación se debe volver a presionar el botón mencionando. Esta funcionalidad resulta muy útil para analizar los datos que se van generando instantáneamente. De lo contrario, se debería terminar de correr la ejecución y analizar al consecuente archivo XML que contiene la traza de la simulación.

4.1.1 Archivo de historial de la simulación

El archivo que resulta de la ejecución de presionar el botón “SAVE” como se visualiza en la interfaz, es un archivo XML que almacena la información enviada a la interfaz visual por los agentes controladores. Recordar que esta información es la misma que envió una formación a su correspondiente controlador. La información estará contenida dentro de las etiquetas de marcado <simulationTrack>.

Dentro de la etiqueta <simulationTrack> se almacenarán los elementos <registerTrain>, <controller>, <trainPosition> y <unregisterTrain>. Todos estos elementos cuentan con su correspondiente marca de tiempo definida como <!ATTLIST timeStamp> para su posterior análisis.

El elemento <registerTrain> se utiliza para indicar el registro de la formación en el controlador. Está constituido por los siguientes campos de información:

- <name> nombre que identifica unívocamente al tren.
- <trainId> identificador o número del tren.
- <length> es la longitud del tren.

- <maxSpeed> es la velocidad máxima a la que la formación puede circular.
- <modo> es el modo elegido para la simulación de la formación y que se refleja en el funcionamiento de las "nubes" B frontales que trabajan de forma estática o dinámica.
- <cloudAFront> es el valor que se definió para la formación como nube más cercana a la formación.
- <cloudARear> análogamente con la cloudAFront, pero ubicada en la parte posterior de la formación.
- <cloudBFront> es la nube más alejada de la formación en la parte frontal. Esta nube puede ser dinámica o estática.
- <cluodBRear> es la nube más alejada de la parte trasera.

El elemento <controller> indica la instanciación de un controlador en la arquitectura global del simulador. El controller es la representación de un circuito de vía y como tal, los valores que representa son los que definen un tramo de vía. Está constituido por los siguientes campos de información:

- <name> nombre del tren.
- <trackId> identificador del tren,
- <trackLength> es la longitud del tren,
- <stationsLength> es la longitud total de las estaciones que representa este circuito de vía.
- <stations quantity="<cantidad de estaciones>"> este elemento indica la cantidad de estaciones que provee el circuito de vía y lo almacena en el atributo quantity. A su vez, este elemento contiene <station> que describen a continuación.
- <station index="<posición de la estación en el circuito actual>"> indica la distancia a la que se encuentra desde el inicio del tramo actual.
- <detentionZones quantity="<cantidad de zonas de detención que contiene el tramo de vía actual>"> indica la cantidad de zonas de detención que posee el circuito definido. Contiene <detentionZone> en su interior.
- <detentionZone index="<posición de la zona de detención dentro del track>"> indica la distancia desde el inicio de circuito de vía a la que se encuentra la zona de detención.

El elemento <trainPosition> es el elemento que más se utiliza y eso es porque todo el simulador está basado en registrar con precisión la posición de la formación. Está constituido por los siguientes campos de información:

- <name> nombre del tren. Dicho nombre es el mismo utilizado para el registro del tren como también cuando se registra la finalización del servicio por parte del tren.

- <position> representa la posición dentro del controlador o tramo de vía correspondiente.
- <speed> es la velocidad a la que se registró el registro correspondiente.
- <state> es el estado al momento de generar el registro de la formación.
- <controller> indica bajo qué controlador o track se encuentra circulando.

El elemento <unregisterTrain> es el elemento que indicará que la formación ha concluido con el servicio y por tanto solicita que se quite del seguimiento del controlador que lo estaba gestionando porque va a concluir su ejecución, tanto la formación como el agente que la representa. Está constituido por los siguientes campos de información:

- <controller> nombre del controlador que es el último por el que circulo y finalizó su servicio.
- <trainId> identificador del tren.
- <trainState> es el último estado registrado por el tren al abandonar la supervisión del controlador.

```
  </trainPosition>
- <trainPosition timeStamp="1485186372961">
    <train>AAA2</train>
    <position>00000</position>
    <speed>000,324</speed>
    <state>2</state>
    <controller>controller1@192.168.0.35:1099/JADE</controller>
  </trainPosition>
- <trainPosition timeStamp="1485186372871">
    <train>AAA2</train>
    <position>00000</position>
    <speed>000,000</speed>
    <state>2</state>
    <controller>controller1@192.168.0.35:1099/JADE</controller>
  </trainPosition>
- <trainRegister timeStamp="1485186371925">
    <name>train2</name>
    <length>100</length>
    <id>AAA2</id>
    <modo>1</modo>
    <cloudAFront>60</cloudAFront>
    <cloudBFront>60</cloudBFront>
    <cloudARear>50</cloudARear>
    <cloudBRear>75</cloudBRear>
  </trainRegister>
- <controller timeStamp="1485186370086">
    <name>controller1</name>
    <trackId>13871301650</trackId>
    <trackLength>1650</trackLength>
    <trackStationLength>100</trackStationLength>
  - <stations quantity="1">
      <station index="0">1650</station>
    </stations>
  - <detentionZones quantity="1">
      <detentionZone index="0">1050</detentionZone>
    </detentionZones>
  </controller>
</simulationTrack>
```

Ilustración 4-3 - Muestra un ejemplo del archivo generado de resguardo de la traza.

4.1.2 La interfaz visual

Las dos imágenes que se muestran a continuación, son el simulador recientemente iniciado y el simulador ejecutándose con tres (3) controladores y cuatro (4) formaciones.

Ilustración 4-4 - Simulador inicializado esperando recibir información de los controladores.

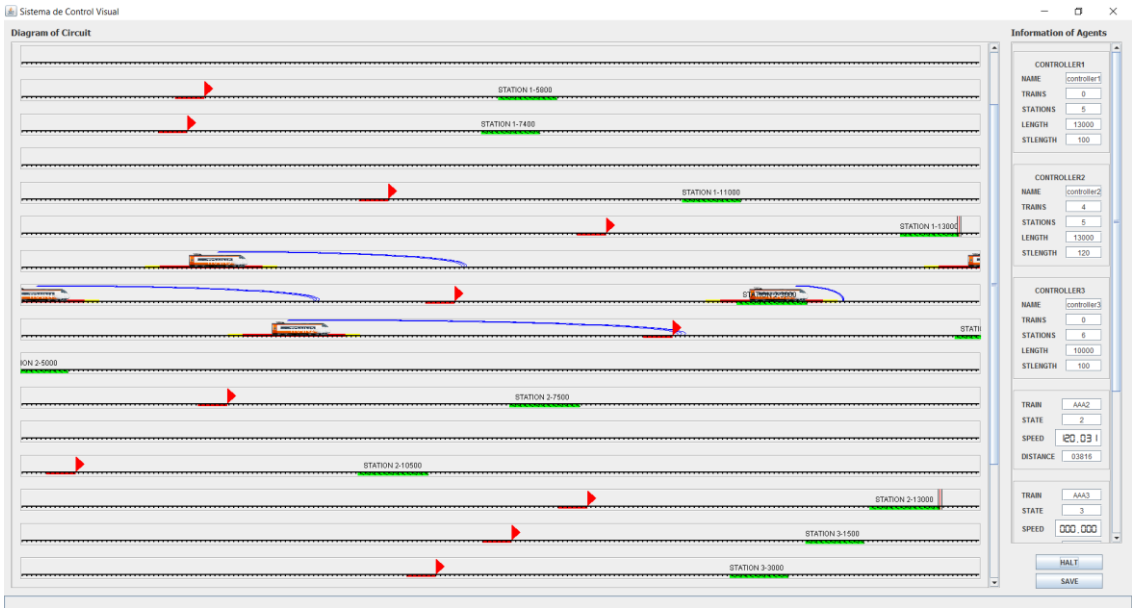


Ilustración 4-5 - Simulador funcionando con cuatro (4) formaciones.

4.1.3 Representación gráfica de los agentes de los controladores

En la Ilustración 4.5, que corresponde al simulador funcionando con varias formaciones y controladores representados. Los controladores se representan de dos maneras al mismo tiempo, al igual que las formaciones ferroviarias. En el panel principal, el panel de los circuitos de vías, los controladores están representados por una línea continua negra con puntos en su parte inferior, emulando los durmientes de las vías férreas. La longitud del circuito está a escala de un metro igual a un pixel. La longitud máxima que se representa en un solo panel completo de es 1629 metros o píxeles en el simulador.

El primer controlador de una simulación, comienza en el margen superior izquierdo y termina donde se ven las tres líneas verticales paralelas, dos (2) líneas de color negro y una (1) línea de color rojo en medio de las líneas negras. Esta última línea, es el verdadero final de circuito. Los restantes controladores representados comenzarán donde termina el primer controlador (a partir de la línea roja del controlador que lo antecede) y finalizará de la misma manera. La segunda representación, se encuentran en el panel del lateral derecho. Ahí se ve en forma resumida el controlador, con su nombre, con la cantidad de formaciones que están bajo su control, la cantidad de estaciones que contiene, la longitud del circuito de vía que representa y la longitud de las estaciones de ascenso y descenso de pasajeros.

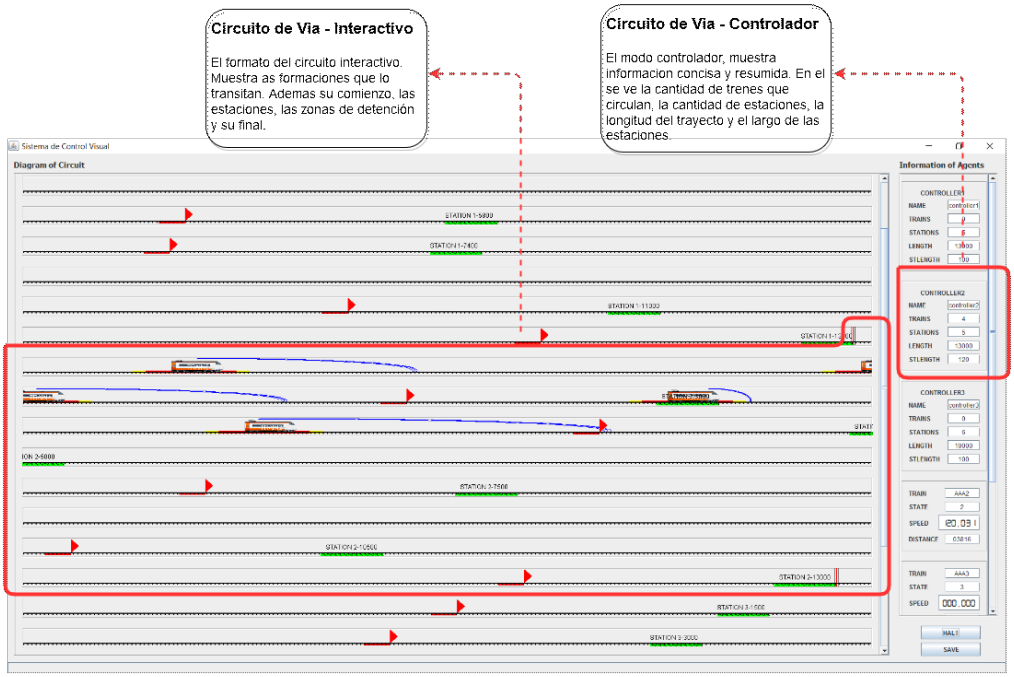


Ilustración 4-6 - Detalles de las visualizaciones de los controladores.

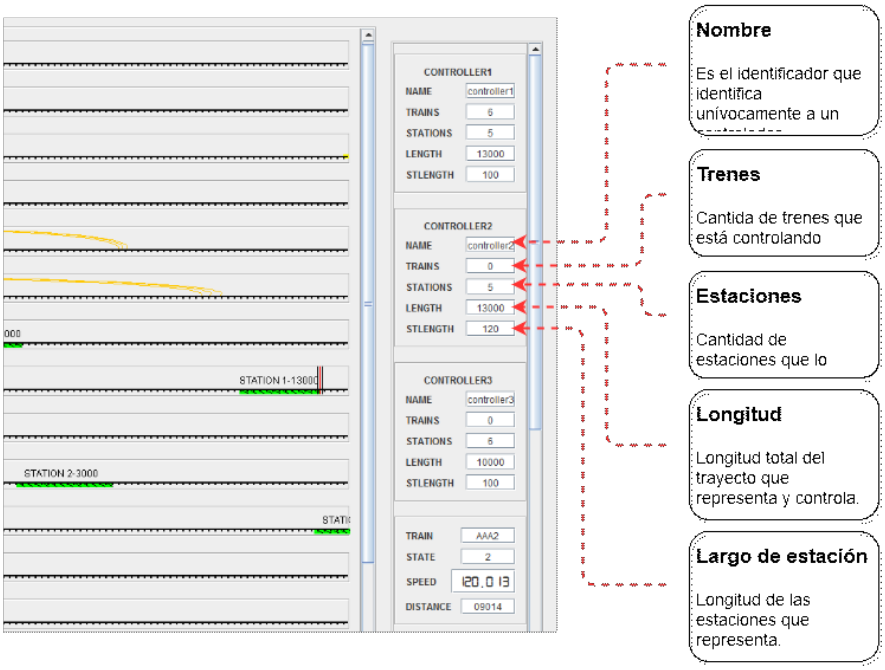


Ilustración 4-7 - Detalles de los controles de los controladores.

4.1.4 Estaciones de ascenso y descenso junto con las zonas de detención.

Dentro del controlador, se encuentran graficadas las estaciones y las zonas de detención de las formaciones que contiene el circuito de vía representado. Las estaciones tienen una longitud que puede ser definida en el agente controlador y las cuales no pueden tener una longitud de menos de 100 metros o pixeles. Están representadas en color verde por sobre el circuito de vía del controlador. Sobre ellas se encuentra sobreimpreso el texto “STATION <# de trayecto o controlador> - <distancia a la que se encuentra dentro del circuito>”. El <# de trayecto o controlador> condice con el número o posición del circuito dentro del circuito global de simulación (entiéndase el circuito compuesto por todos los controladores). El final de un tramo o trayecto se indica con tres (3) líneas verticales, dos (2) de color negro y la tercera de color rojo que es el final propiamente indicado del tramo.

Las zonas de detención se encuentran siempre por delante de las estaciones, por delante, realizando una lectura de izquierda a derecha, la zona de frenado está a la izquierda de la estación correspondiente. Se visualizan en color rojo y tiene la forma de un triángulo de lado. Estas zonas de detención indican a los controladores que los trenes que circulen en su interior próximamente se encontrarán con una estación. Para una utilización eficaz de las zonas de frenado, es conveniente colocarles a una distancia mayor que la

necesaria por la formación para detenerse a máxima velocidad. Para ejemplificar, si una formación circula a 100 km/h estará recorriendo 27.778 m/s. El tiempo necesario para detenerse con una desaceleración de -1 m/s^2 es de Tiempo = (VelocidadFinal - VelocidadInicial) * Aceleración, Tiempo = (0 m/s - 27.778 m/s) * -1 m/s^2 = 27.778 s. Para su detención con una desaceleración de -1 m/s^2 deberá emplear una Distancia = 27,778 m/s * Tiempo - 0.5 * Tiempo^2, donde Tiempo = 27,778 s, quedando Distancia = 385,80 metros.

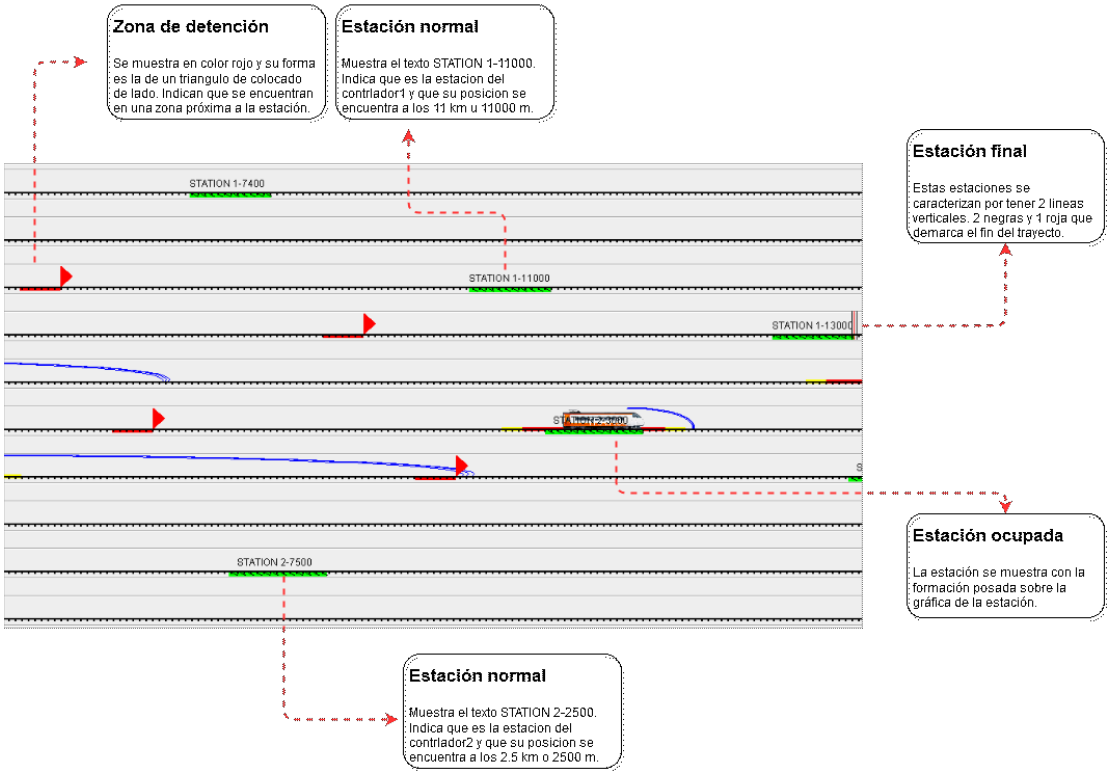


Ilustración 4-8 - Detalles de las distintas estaciones posibles en el simulador y las zonas de detención.

4.1.5 Detalles gráficos de las formaciones ferroviarias.

El panel de control que representa a un tren muestra los datos que lo identifican como el nombre, el estado en el que se encuentra en determinado instante al igual que la velocidad de movimiento, la cual está indicada en kilómetros por hora para una mejor comprensión. Justo por debajo, se muestra la distancia en metros que lleva recorrida dentro de un trayecto del circuito bajo la supervisión de un controlador. Cuando la formación cambia de controlador la distancia recorrida se reinicia a 0 metros recorridos. El cambio que muestra la distancia recorrida, cuando se aprecia que la formación se está deteniendo

prácticamente a cero (0), produce un efecto visual contradictorio, debido a que se ve disminuir la velocidad y el avance no parece reflejarse. Esto se debe a que la unidad de representación elegida para visualizar es el metro y la velocidad es demasiado baja. Recuerde, que el velocímetro de un vehículo su marcación comienza en 20 km/h. Y, por último, se muestra el nombre del controlador que lo está supervisando.

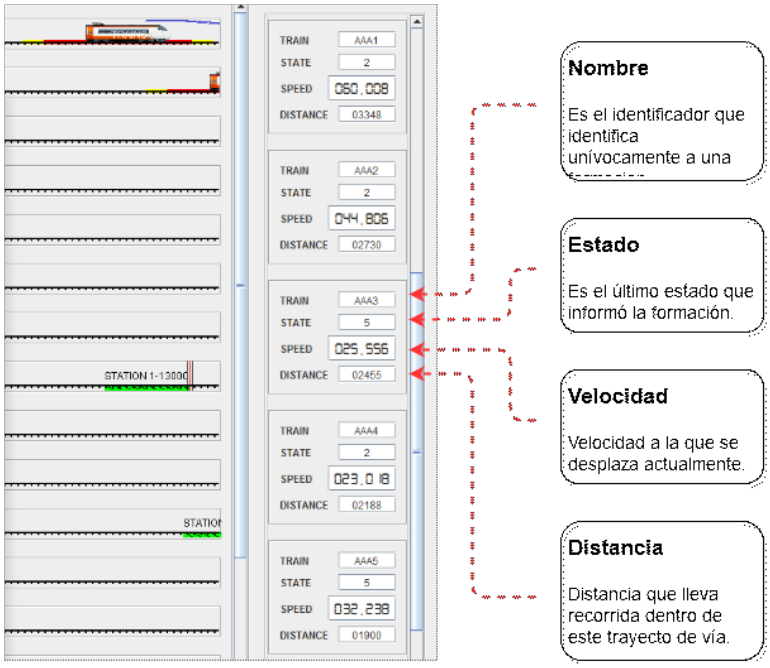


Ilustración 4-9 - Detalle de la información de control de las formaciones.

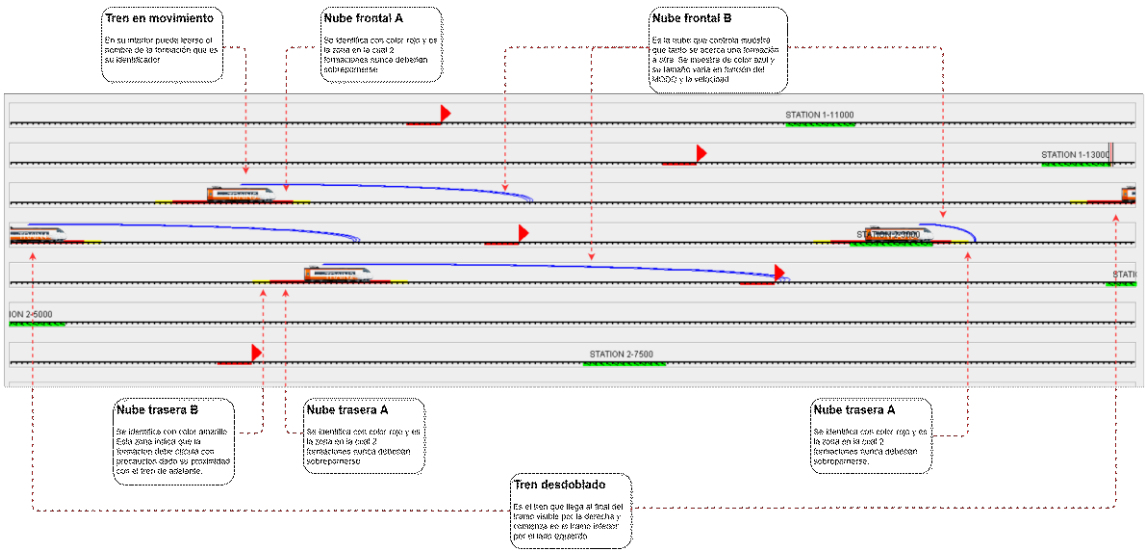


Ilustración 4-10 - Detalles de los trenes que se visualizan en el simulador.

La representación gráfica de las formaciones en el panel interactivo, se puede apreciar que existen distintas representaciones para distintas condiciones. En primer lugar, la formación se representa con la imagen de un tren en cuyo interior se encuentra el nombre de que lo identifica. Debajo de esta imagen, se dibuja una línea de color rojo que indica el tamaño que tienen las cloudAFront y cloudARear (nube A de adelante y nube A de la parte posterior). En ningún caso, una formación con otra, deberían entrar en contacto con dichas zonas o nubes. Las zonas que continúan a las nubes detalladas son las cloudBFront mínima y cloudBRear (nubes B frontal mínima y trasera). La cloudBFront mínima es fija y es a partir de la cual, la cloudBFront dinámica evoluciona indicando la distancia mínima de frenado de la formación. Estas zonas se utilizan para indicar que una formación se está acercando peligrosamente a otro que se encuentra por delante. En las simulaciones donde se carguen los parámetros correctamente para una ejecución, dichas zonas nunca harían contacto.

Por último, la se aprecia un arco de color azul. Este arco representa la evolución de la cloudBFront. La cloudBFront dinámica cuando es mínima es igual a su representación mínima de color amarilla. A medida que la formación avanza y gana velocidad se va anticipando a la formación para indicar la distancia de seguridad y frenado con respecto a una formación que se encuentre por delante.

Como detalle, cuando un tren que está circulando por un trayecto llega al final gráfico del mismo, pasa a dibujar en el panel que se encuentra inmediatamente por debajo. A medida que va terminando el tramo visual, va incrementando su representación en el panel de abajo. La representación en el panel termina cuando la cloudBRear termina de pasar por el último valor del tramo representado en ese tramo.

5 DISEÑO E IMPLEMENTACIÓN DEL SIMULADOR

El diseño del simulador está basado en el modelo de agentes móviles propuesto por la plataforma para desarrollo de agentes JADE. Esta plataforma, además, provee todos los mecanismos para la sincronización, la concurrencia y el control del tiempo de ejecución de los agentes generados. De un modo de vista amplio, el modelo está compuesto por cuatro (4) proyectos JAVA. Estos proyectos son: Agent2Train, Agent2Controller, Agent2Visual y Agent2TrainVocabulary. Todos los proyectos están enlazados por medio de Agent2TrainVocabulary qué es el proyecto que tiene los objetos que se van a utilizar en los restantes proyectos a modo de librería y funcionalidades.

5.1 LA INTERFAZ VISUAL COMO AGENTE

La interfaz visual como Agente, es el agente que cuenta con la menor complejidad en cuanto a su participación en el control de las formaciones. Su funcionalidad se reduce a recibir la información remitida por los controladores informando las posiciones y demás información generada por las formaciones rodantes. La siguiente ilustración muestra el diseño del objeto interfaz gráfica.

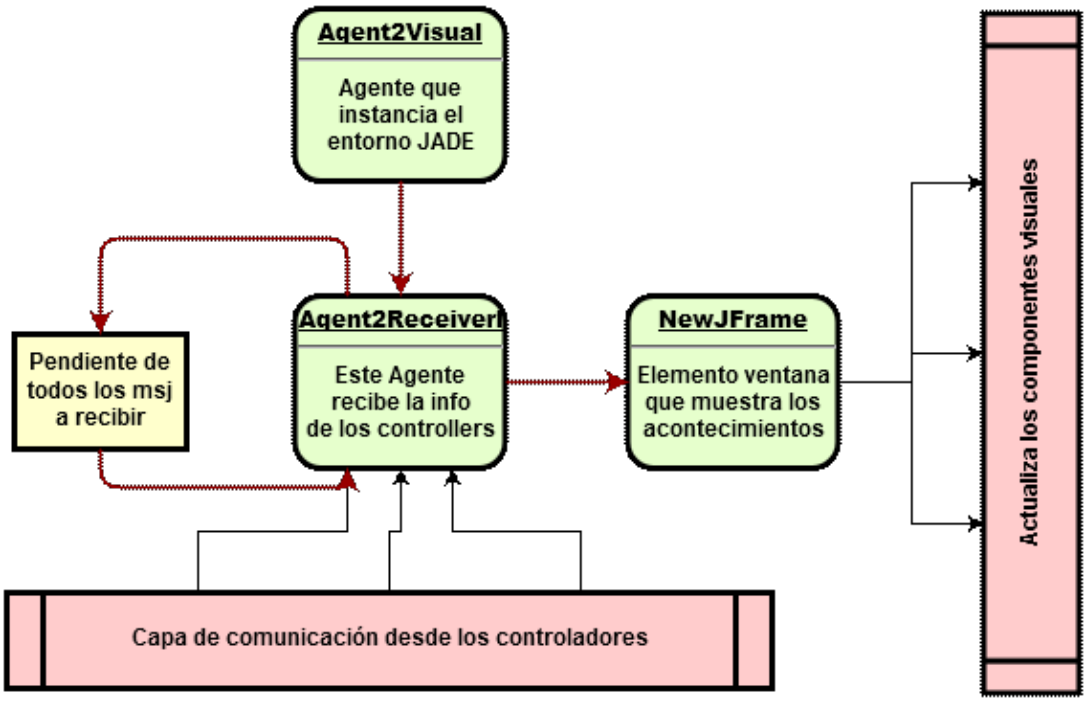


Ilustración 5-1 - Diseño del agente Agent2Visual.

El agente Agent2Receiver se encuentra activamente esperando para recibir datos desde los agentes Controladores. Cada controlador que es agregado al simulador tiene que registrarse con su par inmediato como controlador enlazado y con el agente de la interfaz gráfica para que la interfaz grafique su representación, indicando su comienzo, su fin, qué longitud posee, cuantas estaciones contiene, etc. Para evitar un uso excesivo de la red y del recurso primario como lo es el procesador, los controladores no enviaran información al agente de la interfaz hasta que no se registre actividad en el circuito de vía.

Todo lo mencionado hace parecer que dicho agente no tiene una funcionalidad imprescindible. Pero su importancia radica en traducir rápidamente la información que es informada a nivel numérico a información visual. Siguiendo con los modelos de patrones propuestos por Gamma [Book27] se obtiene un modelo Model-View-Controller.

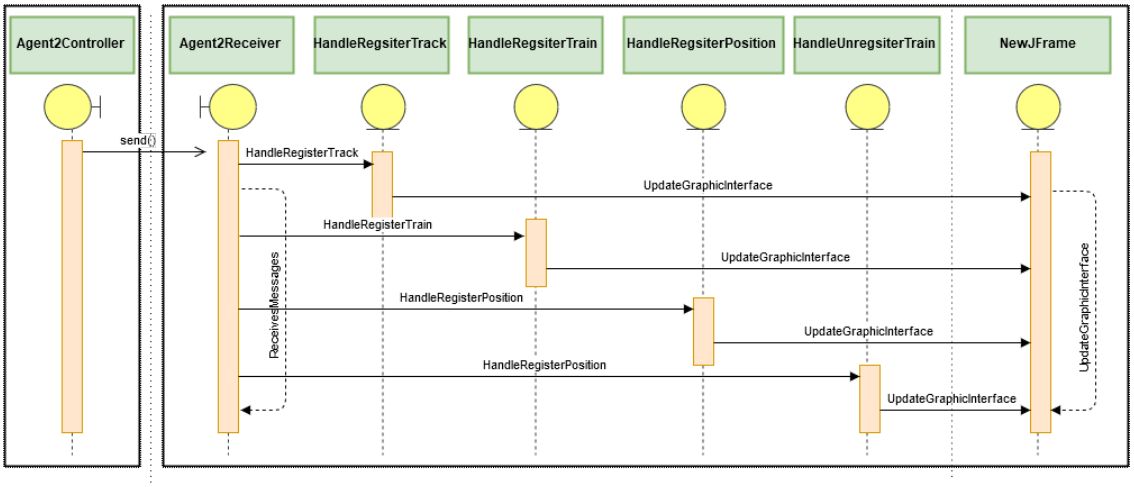


Ilustración 5-2 - Modelo de Interfaz Gráfica.

Una vez que la interfaz gráfica que se encuentra en ejecución, el Agent2Receiver está disponible para recibir las notificaciones de los controladores y dependiendo del tipo que sea la notificación, selecciona el comportamiento con el que dará tratamiento a la notificación para ser representada visualmente. Los comportamientos definidos para el agente Agent2Receiver, como se muestra en la Ilustración 5.2, son HandleRegisterTrack, HandleRegisterTrain, HandleRegisterPosition y HandleUnregisterTrain.

Para que las formaciones puedan circular, es necesario que tengan definido un circuito de vía en el que poder circular. Esto se genera al cargar los controladores o lo que es lo mismo, al ejecutar los Agent2Controller. Estos

agentes notificarán a la interface gráfica con la información del Track, donde el Agent2Receiver que opera la recepción de los mensajes permitirá que el comportamiento HandleRegisterTrack atienda la notificación. El HandleRegisterTrack almacena la información del trayecto, que puede ser todo o una parte del circuito a recorrer, en la colección tracks que se encuentra la clase que implementa el listener que atiende a la interfaz visual, ReceiverInterface.

Para el registro de una formación, el Agent2Receiver recibe nuevamente del agente Agent2Controller los detalles de la formación que se ha registrado en dicho agente controlador y que necesita que se visualice. Previamente, el Agent2Train ha notificado al agente controlador y este ha aceptado el registro del agente tren. Agent2Receiver gestiona la petición al comportamiento HandleRegisterTrain. A su vez, el comportamiento delega en el objeto ReceiverInterface para que almacene el tren en la colección trains y realice la gestión del objeto para que cuando modifique su estado, se notifique y visualice en la pantalla.

Para registrar la posición de las formaciones o el borrado de la formación de la interfaz visual, se realizan los mismos mecanismos descritos para el registro de un tren. La diferencia radica en que, para el caso de registrar la posición, el comportamiento que refleja esta acción es HandleRegisterPosition y para el caso de borrar el tren, el comportamiento es HandleUnregisterTrain. Ambos manejadores, a través del TrainsPositionInterfaceManagerEventListener, notificarán el avance de la formación o provocan la eliminación de la misma del sistema.

El objeto ReceiverInterface, a través del mecanismo que provee Java, implementado con por medio de los EventListener para notificar a objetos dependientes, actualiza la información mostrada en pantalla, mediante el uso de los manejadores definidos como ReceiverInterfaceManagerEventeListener y TrainsPositionInterfaceManagerEventListener. El primer manejador se encargará de las notificaciones que corresponden a los Track u objetos que representan el circuito de vía. El segundo manejador atenderá las notificaciones que indican que una formación se ha movido y necesita que se refresque su representación gráfica.

Ambos manejadores o EventListener tienen asociadas las clases que entienden cómo representar un tren o un tramo de vía. Los objetos JPanelWithStation representan gráficamente al circuito de vía con las estaciones, las zonas de detención y los finales de trayecto. Por otro lado, los objeto GraphicTrain representan visualmente a las formaciones, con su nubes frontales y traseras para indicar donde se encuentran dichas nubes y verificar visualmente las distancias predefinidas de seguridad necesarias.

Para las visualizaciones que se encuentran bajo el título de “Information of Agents”, directamente, los comportamientos definidos realizan la gestión de dicha visualización. Esto es así, porque los objetos que reflejan los datos que son actualizados constantemente, no necesitan ser resueltos de forma compleja. Simplemente, cuando el HandleRegisterTrain o el HandleRegisterTrack obtiene la información enviada, dibujan las estructuras gráficas correspondientes. La información de la posición, que es la información que más abunda, es tratada por HandleRegisterPosition y como este tiene acceso a los JFieldText definidos para mostrar la velocidad y la posición, mediante el método setText() los actualiza in situ.

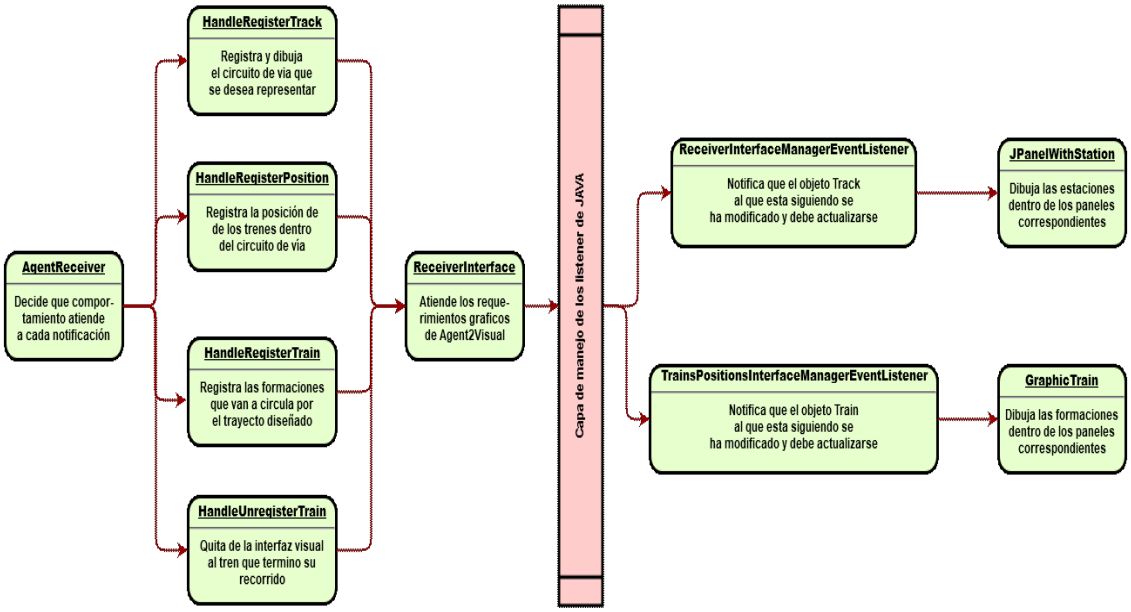


Ilustración 5-3 - Modelo de atención de requerimiento de la interfaz gráfica.

5.2 El proyecto Agent2TrainVocabulary como librería compartida

En este proyecto se encuentran desde objetos como trenes (Train) hasta los códigos de operación de los mensajes que se intercambian entre los controladores y los trenes. En la Ilustración 5.4 tenemos una visión global de su contenido.

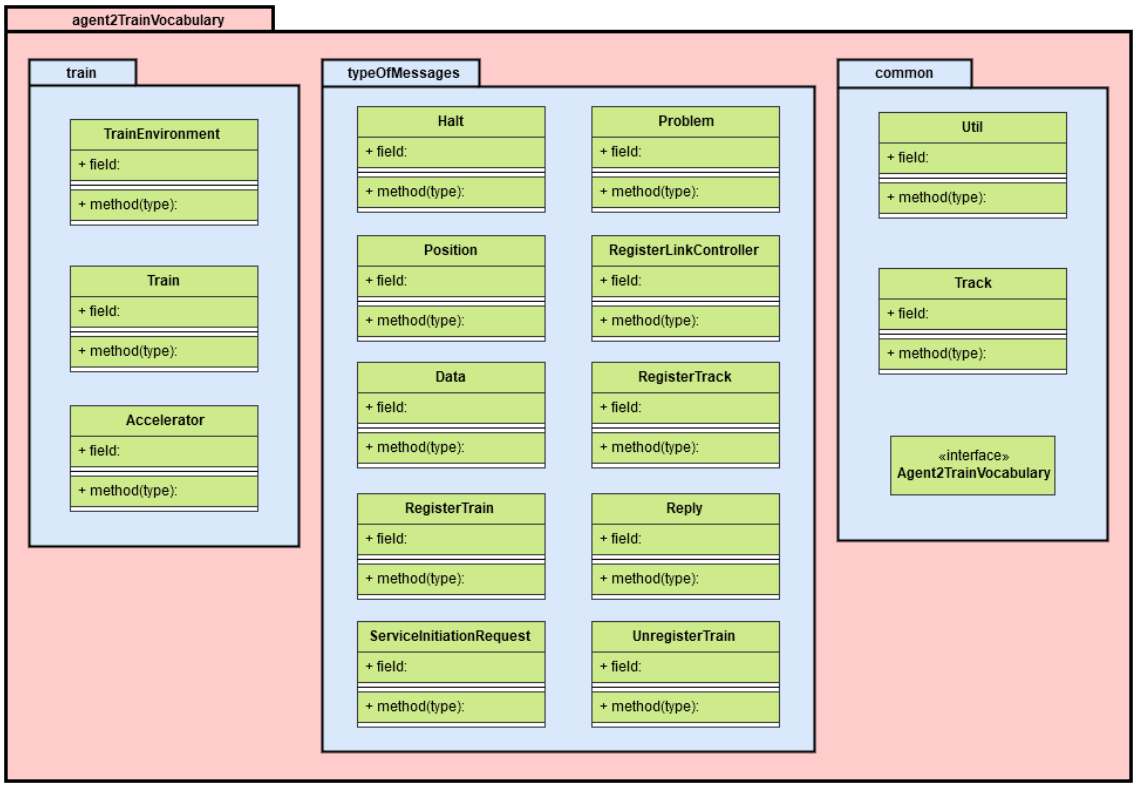


Ilustración 5-4 - Objetos definidos en Agent2TrainVocabulary.

Gran parte de todos estos objetos no tienen relación directa entre sí como se puede apreciar en la Ilustración 5.4. Por ello, trataremos esta sección por los paquetes que contiene y donde los objetos definidos tienen una relación más estrecha que con el resto. Este proyecto tiene definidos tres paquetes, train que contiene la clase Train, Accelerator y TrainEnvironment; el package common que tiene las definiciones de clase o interfaces utilizadas por todos o gran parte de los objetos dentro del simulador y para finalizar, typeOfMessages que define objetos como tipos de mensajes o clases para manejar los mismos.

5.2.1 La clase Train

La clase Train, como su nombre lo indica representa a las formaciones a simular. Esta clase contiene el nombre de la formación, la longitud de la formación, la longitud de cada una de las zonas de control anticipado o “nubes” frontales y traseras como las denominamos dentro de la implementación. También, se pueden definir parámetros como la tasa de frenado, la velocidad máxima a la que puede circular una formación, el tiempo que el tren espera para realizar la comunicación con el controlador asignado, como también el modo de operación que deseamos (cantón fijo o cantón dinámico) e incluso se puede

definir si se desea que el tren espere un tiempo random para continuar o reanude el servicio mediante la pulsación de la tecla ENTER.

Las zonas de control o "nubes" son utilizadas para anticipar la posición del ferrocarril. Tanto para las nubes traseras como delanteras, no incluida la nube frontal B o primaria, su cálculo se hace mediante la suma del valor definido para dicha zona de control sumado o restado a la posición actual de la formación dependiendo de si nos encontramos en el frente del tren o en la parte posterior.

$$\boldsymbol{cloudAFront} = [\boldsymbol{posición\ del\ tren}] + [\boldsymbol{longitud\ de\ cloudAFront}]$$

$$\boldsymbol{cloudARear} = [\boldsymbol{posición\ del\ tren}] - [\boldsymbol{longitud\ de\ la\ formación}] - [\boldsymbol{longitud\ de\ cloudARear}]$$

$$\boldsymbol{cloudBRear} = [\boldsymbol{posición\ del\ tren}] - [\boldsymbol{longitud\ de\ la\ formación}] - [\boldsymbol{longitud\ de\ cloudBFront}]$$

Ecuación 5-1 - Fórmulas de cálculo de la posición de las zonas de control.

Para la nube de control frontal primaria o B, el cálculo es más complicado, dado que involucra la velocidad a la que se mueve la formación y que solo es calculado por el tren, sino que también debe realizarlo es el controller para garantizar la seguridad en la circulación de las formaciones. Por tanto, la velocidad que le es comunicada al controlador puede tener pequeñas variaciones de la velocidad a real a la que se está moviendo la formación, o porque la formación se encuentra en realizando un movimiento rectilíneo uniforme variado [Book29] o porque ocurrió un retraso en las comunicaciones. Recordar que la distancia que puede recorrer un objeto que se mueve a una velocidad de 100 km/h equivale a que recorre un total de 27,7778 metros por segundo.

Para la construcción de la nube frontal primaria y determinar su posición en el circuito de vía, se computa la distancia de frenado necesaria para detener el tren sumado a la nube frontal secundaria con la posición del tren en cuestión. Veamos cómo sería la función resultante.

Para comenzar con el cálculo, necesitamos cuantificar el tiempo que le supondría a la formación detenerse por completo con base a la velocidad que se encuentra circulando.

$$\boldsymbol{cloudBFront} = (\boldsymbol{tiempo\ para\ detenerse}^2) + \boldsymbol{cloudAFront}$$

Ecuación 5-2 - Fórmula para el cálculo de la nube primaria frontal.

Donde <$tiempo\ para\ detenerse$> es igual a la velocidad que circula la formación en metros por segundos dividido la aceleración con la que se esta

deteniendo. Para este cálculo, utilizamos la fórmula matemática que calcula la velocidad de un objeto con una aceleración constante.

$$\boldsymbol{Velocidad\ final\ =\ Aceleración(\frac{m}{s^2})\ \times\ deltaTiempo}$$

Ecuación 5-3 - Formula para cálculo de la velocidad final con aceleración constante.

Haciendo el correspondiente pasaje de términos obtenemos que:

$$\boldsymbol{deltaTiempo\ =\ Velocidad\ final\ \div\ Aceleración}$$

Ecuación 5-4 - Fórmula para calcular el tiempo transcurrido luego de acelerar constantemente.

Una vez obtenido el tiempo necesario para realizar la detención de la formación, se pasa a calcular la distancia necesaria para la detención.

$$\boldsymbol{EspacioParaDetenerse\ =}$$

$$\boldsymbol{Velocidad\ \times\ deltaTiempo\ \pm\left(\frac{1}{2}\times Aceleración\ \times\ deltaTiempo^2\right)}$$

Ecuación 5-5 - Fórmula para calcular el espacio recorrido.

A partir de la Fórmula 5.5, se obtiene la Fórmula 5.3. La formación por constitución puede acelerar o detenerse a con una aceleración de $\pm\,\boldsymbol{1\frac{m}{s^2}}$. Por este motivo, podemos simplificar la fórmula 5.5 así:

$$\boldsymbol{EspacioParaDetenerse\ =\ Velocidad\ \times\ deltaTiempo\ \pm\left(\frac{1}{2}\times\ deltaTiempo^2\right)}$$

Ecuación 5-6 - Fórmula para calcular el espacio recorrido con aceleración de ± 1 m/s².

Esto es posible debido a que la aceleración o desaceleración de la formación es de $\pm\,\boldsymbol{1\frac{m}{s^2}}$ para todo momento y puede anularse el término porque no aporta variación al resultado. En este punto, podemos ver que la velocidad, expresada en metros por segundo es igual a $\boldsymbol{deltaTiempo}$, dejando la ecuación como

$$\boldsymbol{EspacioParaDetenerse\ =\ deltaTiempo\ \times\ deltaTiempo\ \pm\frac{1}{2}\times\left(deltaTiempo^2\right)}$$

Ecuación 5-7 - Fórmula que asume que la aceleración es ± 1 m/s².

Ahora, fácilmente podemos apreciar que la fórmula puede reducirse quedando así:

$$\boldsymbol{EspacioParaDetenerse\ =\ deltaTiempo^2\ \times\frac{1}{2}}$$

Ecuación 5-8 - Fórmula resultado para realizar el cálculo de la nube frontal primaria.

5.2.2 El simulador del movimiento, la clase Accelerator

La clase Accelerator implementa el thread que es el responsable de calcular la cinemática del movimiento del tren. Por esto, es totalmente independiente del agente, permitiendo realizar una simulación con un alto grado de afinidad con el problema real. El thread tiene definidos métodos que permiten controlarlo al igual que puede ocurrir con una formación. Los métodos definidos hacen que el thread se detenga, avance, mantenga una velocidad estable, entre otras funcionalidades.

Este objeto funciona en base a las leyes que rigen para el movimiento rectilíneo uniformemente acelerado. Constantemente, el thread está ejecutando una resolviendo una función de movimiento. Para poder alterar la ejecución de una función, existe otro hilo que modifica el estado del thread principal dentro de una sección crítica.

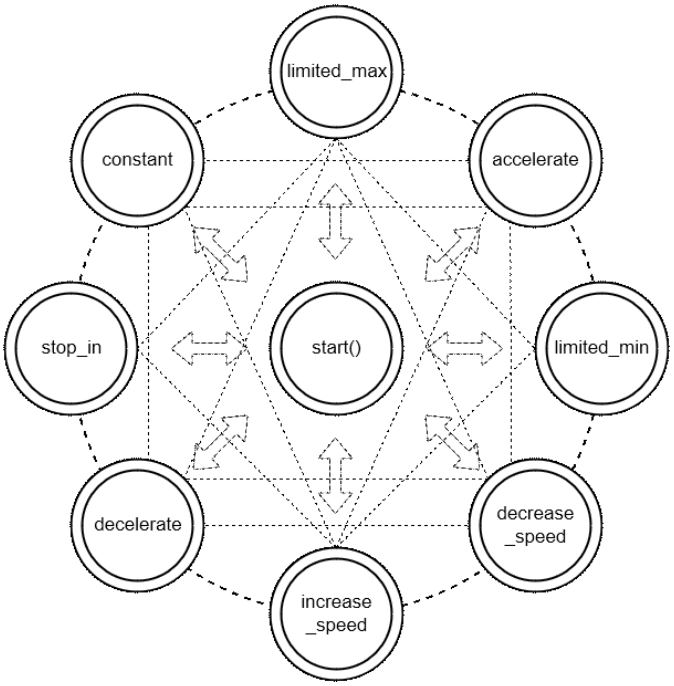


Ilustración 5-5 - Diagrama de estados del thread Accelerator.

Como se puede apreciar en la Ilustración 5.5, una vez que el thread ha sido iniciado, puede pasar de un estado a otro indistintamente. Las únicas restricciones con las que se encuentra, son con las limitaciones que hereda del tren, como ser la velocidad máxima y que no puede desplazarse en sentido contrario al que se encuentra circulando. Esta última restricción se fundamenta, en la precondición del simulador que sólo simula a las formaciones rondando en un sentido. Si esta condición fuese necesaria, debería proveerse una zona similar a la estación, en que una vez detenida la formación, realice un cambio de

vía. Esta vía, debería estar representada en una simulación paralela, a la cual el Agent2Train puede migrar y de esa forma poder rodar en sentido contrario, en visión de los agentes controladores.

Esta acción, motivaría a que el controlador que se encuentra administrando la circulación de la formación en cuestión deje de hacerlo, para cederle el control a otro controlador. Esto sería similar a un cambio de controlador como se ha descrito en el presente trabajo, con la salvedad que, en lugar de hacerse en los finales del recorrido, deba realizarse en un punto medio. Dicho punto debería ser precargado, al igual que una estación o una zona de detección.

Para tener una mejor comprensión del thread Accelerator, a continuación, se describen los estados a los que puede acceder:

- Constant: es el estado inicial del thread. Este estado ejecutará la función de cinemática que calcula la distancia para una velocidad y aceleración constantes. En ambos extremos, cuando al tren está detenido o a máxima velocidad, el tren se encontrará en este estado. Lo mismo ocurre si la formación se encuentra detrás de otra que va a una menor velocidad y se necesita que la formación no traspase los límites de seguridad.
- Accelerate: cuando la formación tiene que comenzar a rodar, el thread debe pasar a este estado. Este estado provocará que la formación registre el avance en metros, pero también, su velocidad comenzará a registrar un incremento con una tasa de aceleración de $\mathbf{1}\,\boldsymbol{m/s^2}$. Este incremento de la velocidad se registrará hasta que la formación llegue a su máxima velocidad y cambie al estado Constant o hasta que ocurra un cambio de estado.
- Decelerate: es el estado opuesto a Accelerate. La tasa de aceleración para este estado es negativa de $-\mathbf{1}\frac{\boldsymbol{m}}{\boldsymbol{s^2}}$. Si la formación llega a una velocidad de 0 km/h el thread cambiará de estado a Constant.
- Stop_in: este estado conoce de antemano la distancia a la que se debe detener siempre con una tasa de detención igual o superior a $-\mathbf{1}\frac{\boldsymbol{m}}{\boldsymbol{s^2}}$. El valor de desaceleración podrá ser variado cuando el estado detecte que necesita aumentar la presión de frenado o disminuirla.
- Increase_speed: modifica la velocidad en una unidad. La unidad por elegida por diseño es de 1 km/h o lo que es lo mismo, 0.27778 m/s. Una vez que el tren se encuentra en la velocidad solicitada pasa al estado Constant.
- Decrease_speed: al igual que el estado anterior, pero en vez de aumentar, disminuye la velocidad en una unidad o 0.027778 m/s. También, una vez que alcanza el valor solicitado, pasa al estado de Constant.

- Limited_max: este estado realiza una variación en la velocidad máxima a la que puede circular la formación. Nunca será superior a la velocidad máxima a la que puede circular el tren por constitución del modelo. Su uso es útil cuando se conoce la velocidad de la formación que le precede en el circuito de vía.
- Limited_min: es el estado contrario a Limited_max. Este estado, en primer lugar, evita que la velocidad de la formación llegue a 0 km/h, obligando a la formación a que siga rodando sin detenerse.

La generación de datos de este thread es a instantes parciales y no totales, lo que produce una mejor precisión, pero a una carga de procesamiento mayor. El agente train que lo contiene se comunica con el Accelerator a través de la modificación de datos en una sección crítica. El funcionamiento de Accelerator es simple, registra una marca de tiempo inicial, chequea si su estado fue modificado, sin importar que se seleccionó o modificó, el thread registra el tiempo nuevamente y realiza el cálculo de la cinemática correspondiente, para recién en ese punto cambiar de estado si fuese indicado. El último tiempo registrado, ahora pasa a ser el tiempo inicial y continua con el estado que le fuere asignado.

El tiempo de Latch o lapso de retorno para la realización de acciones es de 10 milisegundos. Este lapso de tiempo es lo suficientemente corto y necesario para no perder precisión en los cálculos. Visto de otra forma, la medición de los valores como la distancia, la velocidad, etc., se va realizando a intervalos de 27,778 milímetros si consideramos que la formación se está moviendo a 100 km/h.

5.2.3 Las clases de tipos de mensajes de información

Estas clases se definen para determinar con qué comportamiento, el agente que las recibe, atenderá la solicitud de servicio recibida. Están definidas dentro del package typeOfMessages. No definen un comportamiento más allá del que se puede proveer para realizar la petición de información o modificar la información que representan. Todas estas clases implementan la interface java.io.Serializable que permite que puedan ser enviadas como un stream de bytes y ser reconstruidos en el receptor.

5.2.4 La clase Data

La clase Data, es la clase que se utiliza a modo de wrapper para enviar en su interior objetos del tipo Position, RegisterLinkController, RegisterTrain, RegisterTrack, ServiceInitiationRequest y UnregisterTrain. El atributo trainId identifica a la formación que produce el objeto Data para ser enviado al controlador. En el campo object, como se mencionó anteriormente, se envían los objetos que se utilizan como mensaje de comunicación. El atributo speed contiene la velocidad a la que se encuentra la formación. En todos los casos, la velocidad es enviada, puesto que es un atributo central de la simulación. Puesto que no sería deseable que la formación solicite iniciar un servicio y, que, como contrapartida, su velocidad al momento de la petición es de 20 km/h.

state_service indica el estado de la formación al momento de realizar la solicitud. El atributo register está colocado para la situación en la que se necesite una confirmación, por parte del tren, de que el mensaje ha sido registrado por el controlador. En esta etapa se descartó su uso, dejando esta responsabilidad en JADE que utiliza el mismo mensaje que recibe para devolver como respuesta.

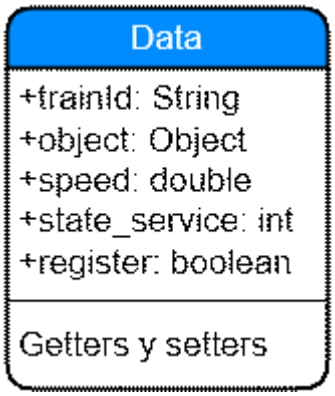


Ilustración 5-6 - Clase Data utilizada para enviar información.

5.2.5 La clase Position

La clase Position se utiliza para realizar el envío de la posición de las formaciones dentro del simulador a los controladores. La clase contiene la última posición registrada por la formación en el atributo last. En el atributo current guarda la posición actual. El ID es el número de movimientos de posiciones que son solicitada y se incrementa cuando se setea la posición actual por medio del setter setCurrent().

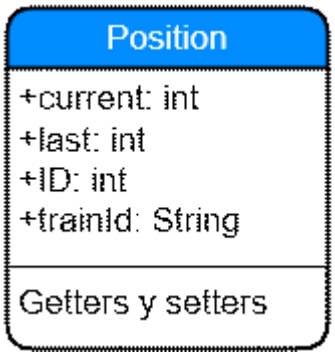


Ilustración 5-7 - Clase Position.

5.2.6 La clase RegisterLinkController

Cuando un controlador recibe un mensaje que contiene este objeto, procederá a registrar al emisor del mensaje como un controlador adyacente y continuo a él. Esto le estará indicando que al controlador que recibe el mensaje que no es el último controlador del trayecto que se está configurando. El tipo AID es el objeto que define JADE como objeto identificador de un agente en la plataforma o Agent IDentifier.

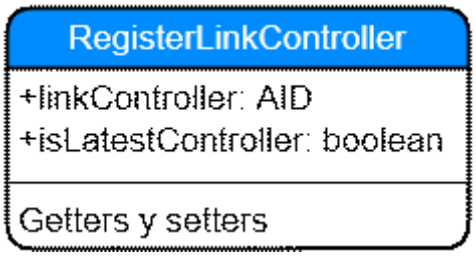


Ilustración 5-8 - Clase RegisterLinkController.

5.2.7 La clase RegisterTrain

Los agentes que representan a los trenes, al momento de ser instanciados tienen que registrarse en el primer controlador del circuito de vía propuesto. Para ello, envía el mensaje con el objeto Data que en su interior contiene un objeto del tipo RegisterTrain. Para realizar esta acción, el objeto RegisterTrain envía un objeto Train y el nombre del mismo. El controlador, al recibir este objeto, devuelve al tren la información correspondiente al track que representa y,

además, le envía el TYPE_REGISTER_INIT_SERVICE indicándole que puede iniciar el servicio.

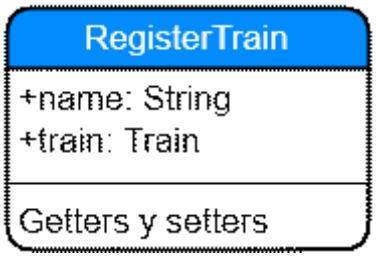


Ilustración 5-9 - Clase RegisterTrain.

5.2.8 La clase RegisterTrack

Esta clase es utilizada por los agentes controladores para registrar en el agente de la interfaz visual la representación del Track sobre el cual realizan el control de la simulación. Esta clase envía un objeto Track en su interior. El agente de la interfaz visual al recibir este objeto dispara el comportamiento para registrar el trayecto, HandleRegisterTrack.

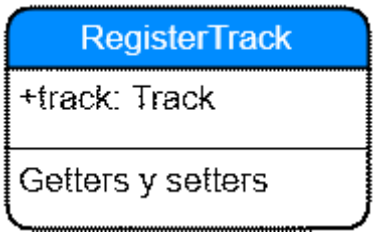


Ilustración 5-10 - Clase RegisterTrack.

5.2.9 La clase Reply

Es la clase que se utiliza como respuesta. El agente controlador responde a los mensajes que le envían las formaciones con esta clase. Los atributos type y msg se corresponden con los definidos en la interface Agent2TrainVocabulary. Recordar que los tipos definen los estados en los que se pueden encontrar una formación. Por ello, esta clase contiene los sendos setters para los estados con sus mensajes, obligando de esta manera a que los agentes trenes realicen la

traducción correspondiente. Estos setters tienen la particularidad de comenzar su definición como setType<nombre del valor que setean>.

Continuando con la descripción de los atributos, se tiene slowlyFlag que le indica a la formación que debe aminorar la marcha. El atributo approachFlag indica que la formación se está acercando a una formación que se encuentra adelante. nextController le remite al Agent2Train el siguiente controlador. speedRatioLimit informa al agente tren que disminuya la velocidad un porcentaje definido de la velocidad a la que circula. maxSpeed limita al tren que no supere la velocidad indicada. Por último, la clase posee el atributo object, que permite que, en caso de ser necesario, se agregue un objeto adicional en el mensaje.

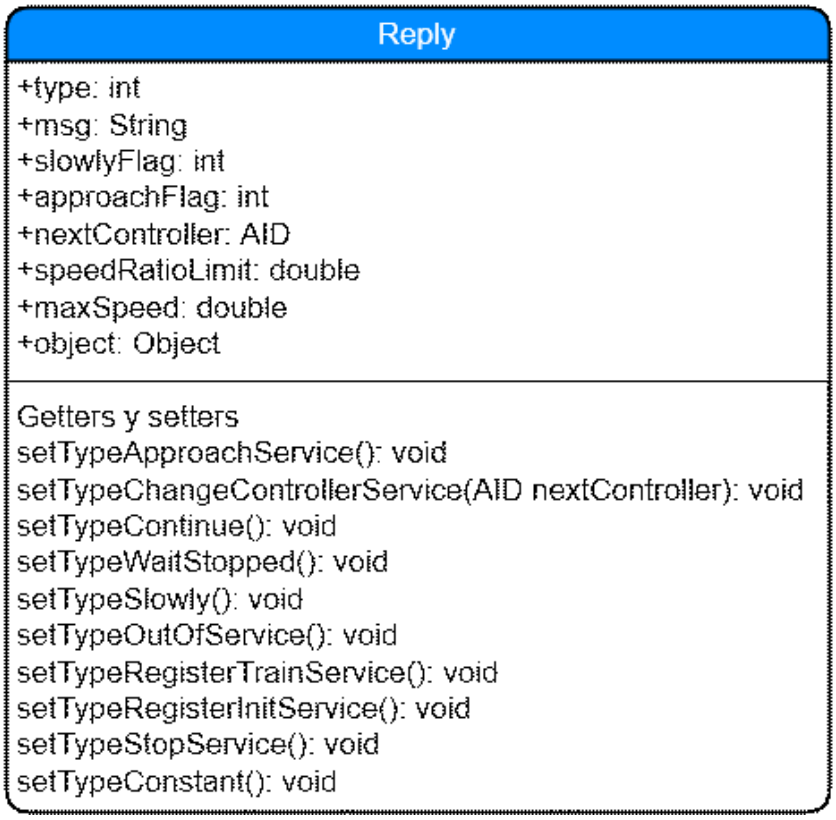


Ilustración 5-11 - La clase Reply utilizada para respuestas desde los controladores.

5.2.10 La clase ServiceInitiationRequest

Esta clase provoca que el controlador autorice al tren a iniciar el servicio. Solo tiene un atributo que es uno de los estados disponible en la interface Agent2TrainVocabulary, en caso de ser necesario puede setearse para confirmar lo que ya, de por sí, provoca la recepción de esta clase. El

Agent2Controller receptor de esta clase evalúa si retorna el permiso para que la formación inicie el servicio. Si las condiciones para que el tren avance están dadas, el agente controlador devuelve dentro del objeto Reply el tipo de mensaje TYPE_CONTINUE. De no ser así, el controlador informa al tren dentro del mismo Reply que vuelva al estado TYPE_REGISTER_INIT_SERVICE para que vuelva a solicitar la autorización para iniciar el servicio.

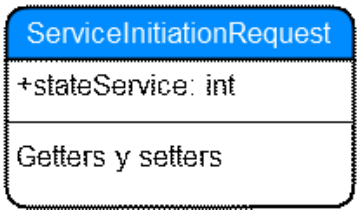


Ilustración 5-12 - Clase ServiceInitiationRequest.

5.2.11 La clase UnRegisterTrain

Esta clase es utilizada por el Agent2Train para solicitar que se lo elimine del control del Agent2Controller. Es objeto es enviado cuando la formación termino de salir por completo del track en el que se encontraba. Esto significa que no solo la longitud total del tren atravesó el ultimo metro del tramo de vía, sino que necesita que la nube más alejada de la parte trasera del tren, también haya pasado.

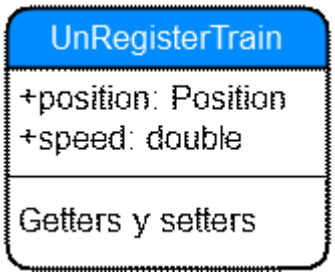


Ilustración 5-13 - Clase UnRegisterTrain.

5.2.12 Las clases Halt y Problem

Estas dos últimas clases se utilizan en el simulador para indicar la detención del agente o informar un problema en la recepción en el agente que

responde con este mensaje. Para la clase Halt se definen dos atributos: time y type. El atributo time marca el tiempo en el que se realiza la detención del simulador y type indica qué acción se debe realizar ante la recepción de un objeto Halt. Los valores definidos para type se encuentran definidos en la interface Agent2TrainVocabulary como HALT o RESUME.

El objeto Problem, al ser detectado, estará dando cuenta de que un evento no definido ha acontecido.

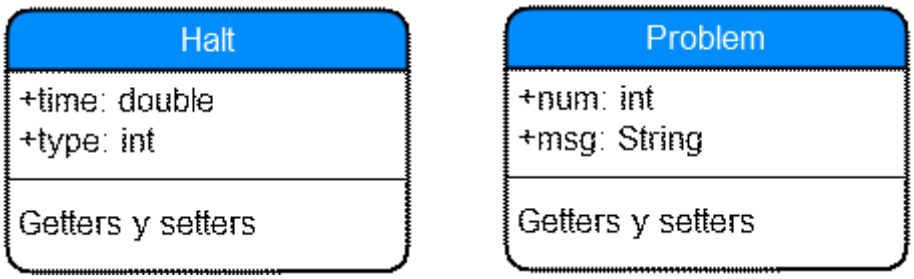


Ilustración 5-14 - Clases Halt y Problem respectivamente.

5.2.13 La interface Agent2TrainVocabulary

Esta interfaz contiene todos los estados posibles que puede encontrarse un agente tren y que estados puede modificar el agente controlador. Ambos agentes utilizan distinta codificación para los estados. Esto es así, para evitar que dos agentes de distinta clase al manejar los mismos códigos produzcan algún tipo de ruido en la comunicación de los estados, obligando a que cada agente tenga un traductor de los códigos enviados a cada agente.

Los atributos que se definen a continuación son globales y utilizados por todos los agentes directa o indirectamente.

- CONTROLLER_AGENT = "Server agent"; que es el tipo utilizado para identificar a los agentes controladores de los agentes trenes.
- VISUAL_CONTROLLER_AGENT = "Visual Server agent"; es el atributo definido para identificar entre los agentes al agente que representa a la interfaz visual.

Estos atributos que a continuación se detallan son utilizados solo por las formaciones trenes o Agent2Train.

- OUT_OF_SERVICE = 0; Indica que el tren ha terminado el recorrido

- REGISTER_TRAIN_SERVICE = 1; Registra el tren en un controlador, cada vez que se cambia de controlador, se debe registrar enviando este estado.
- RUNNING_SERVICE = 2; El tren se encuentra en movimiento por la vía cuando esta en este estado.
- WAITING_TO_CONTINUE_SERVICE = 3; El tren está detenido esperando por continuar el servicio.
- SLOWLY_SERVICE = 4; El tren se encuentra aminorando la marcha.
- APPROACH_SERVICE = 5; Se ha detectado un posible estado en el que puede producirse un abordaje.
- CHANGE_CONTROLLER_SERVICE = 6; El tren dejará un controlador para pasar a la órbita de control de otro controlador.
- REGISTER_INIT_SERVICE = 7; Se registra que está comenzando a operar el servicio, para esto ya se registró el tren.
- STOPPING_SERVICE = 8; Detener el servicio como sea posible.
- CONSTANT_SPEED_SERVICE = 9; El tren debe circular a velocidad constante.
- STOP_CONFIRMATION = 10; El tren permanece detenido.

Desde el lado de los Agent2Controller los atributos definidos para los distintos estados contienen mensajes textuales, a continuación, describimos las duplas como (tipo, mensaje):

- TYPE_CONTINUE = 100; MSG_TYPE_CONTINUE = "CONTINUE"; Indicará al tren que continúe circulando.
- TYPE_WAIT_STOPPED = 101; MSG_TYPE_WAIT_STOPPED = "WAIT STOPPED"; Indica al tren que aguarde detenido.
- TYPE_SLOW = 102; MSG_TYPE_SLOW = "SLOWLY"; Indica al tren que comience a detenerse lentamente.
- TYPE_FREE = 103; MSG_TYPE_FREE = "FREE TRACK"; Indica al tren que tiene libre circulación.
- TYPE_OUT_OF_SERVICE = 104; MSG_TYPE_OUT_OF_SERVICE = "OUT OF SERVICE"; Indica que el tren ha concluído el servicio.
- TYPE_APPROACH_SERVICE = 105; MSG_TYPE_APPROACH_SERVICE = "APPROACH SERVICE"; Se está informando que el tren que está siendo controlado se está aproximando a un tren que se encuentra por delante.
- TYPE_CHANGE_CONTROLLER_SERVICE = 106; MSG_TYPE_CHANGE_CONTROLLER_SERVICE = "CHANGE CONTROLLER SERVICE"; El controlador utilizará este mensaje para que la formación cambie de controlador.

- TYPE_REGISTER_INIT_SERVICE = 107; MSG_TYPE_REGISTER_INIT_SERVICE = "REGISTER INIT SERVICE"; Es utilizado cuando el tren desea comenzar a circular por el circuito de vía.
- TYPE_REGISTER_TRAIN_SERVICE = 108; MSG_TYPE_REGISTER_TRAIN_SERVICE = "REGISTER TRAIN SERVICE"; Registra que el tren ha comenzado el servicio.
- TYPE_STOP_SERVICE = 109; MSG_TYPE_STOP_SERVICE = "STOPPING TRAIN SERVICE"; El controlador le indica al tren que se mantenga detenido.
- TYPE_CONSTANT_SPEED_SERVICE = 110; MSG_TYPE_CONSTANT_SPEED_SERVICE = "SPEED CONSTANT TRAIN SERVICE"; Le indica a la formación que continúe circulando a una velocidad constante.

Y, por último, los siguientes atributos:

- RATE_DECREASE_CONTROLLER = 1; Este valor se usa para realizar la suma del número de los controladores.
- HALT = 9001; Es el código utilizado para indicar a un tren que su estado será de detención.
- RESUME = 9002; Este código es utilizado luego del código HALT para que la formación continúe como estaba funcionando.
- STOP_CONFIRMATION = 10; Este valor es utilizado para contabilizar la cantidad de confirmaciones que la formación debe realizar cuando se detiene en una estación. Este conteo es gestionado por el controlador que controla al tren en cuestión. Alcanzado dicho valor de confirmaciones de stop, se habilita a la formación a continuar desde el controlador.

5.3 El proyecto Agent2Controller

Este agente carga con la responsabilidad de controlar todos los sucesos de su entorno. Los sucesos que actualmente puede resolver son, enlazarse a sí mismo con el controlador visual, enlazar un controlador con su antecesor, permitir que una formación se registre en su circuito de vía, que la formación solicite iniciar su circuito de trabajo, finalizar su circuito de trabajo o cambiar de controlador, también permitir que las formaciones circulen con total seguridad, indicándoles donde detenerse o porque se encuentran en una estación de ascenso/descenso de pasajero o como porque alguna formación está detenida delante suyo, obviamente impidiendo el paso.

5.3.1 Ejemplo de definición de un controlador para su ejecución

controller1:agent2controller.Agent2Controller(

NombreDelTrack, //por defecto se utiliza la palabra “Track”

LongitudDelTrack,//distancia total del tramo que representa este controlador

LongitudEstación,//longitud de las estaciones

"**stops#**>", //palabra reservada para indicar el comienzo de la definición de las zonas de detención, # indica cuantas zonas de detención contendrá.

“ZonasDeDetención”, // 100, 1452, etc., valores enteros que representan la ubicación de las zonas de detención.

"<stops", //para cerrar la definición de las zonas de detención, utilizamos la palabra reservada <stops.

"**stations#>**", //al igual que para las zonas de detención, pero para las estaciones.

“PosiciónDeEstaciones”//100, 1563, etc., valores enteros que representan la ubicación de las estaciones, debe evitarse que se solapen con las zonas de Detención.

,**"<stations"**) //la palabra reservada <stations da fin a la definición de las estaciones.

5.3.2 El modelo del proyecto Agent2Controller

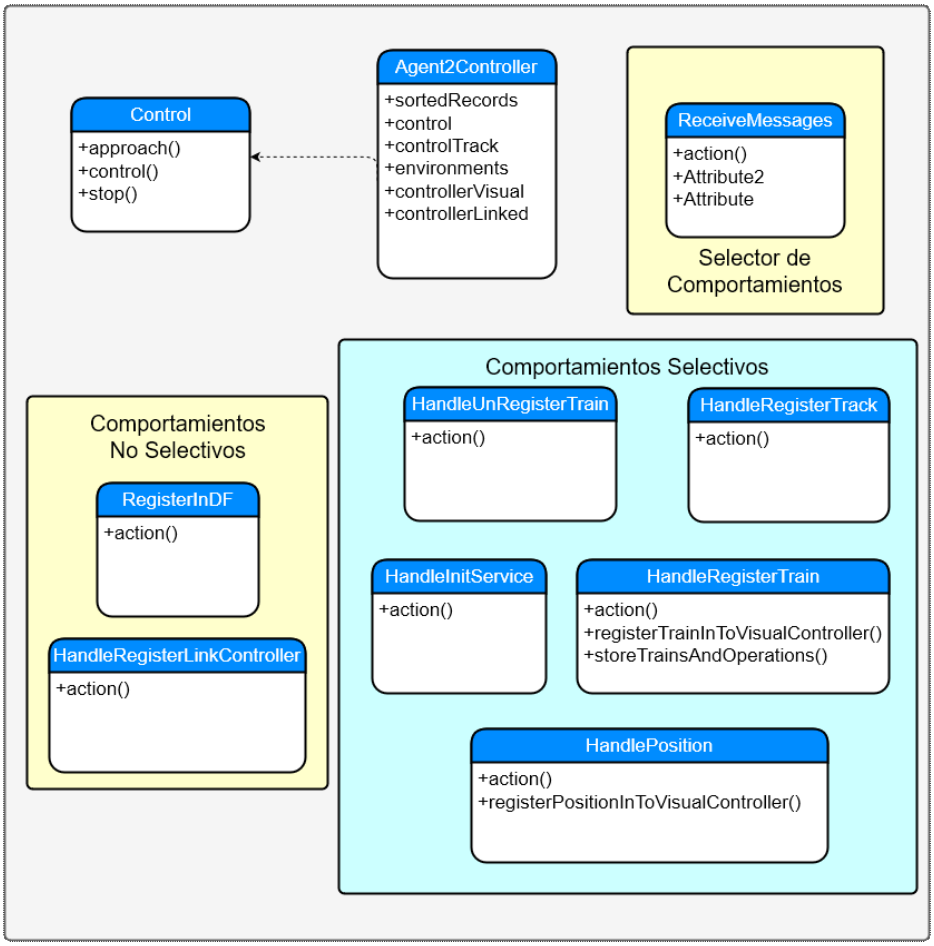


Ilustración 5-15 - Proyecto agent2Controller con sus Behaviours.

La Ilustración 5.15 muestra el esquema del proyecto agent2Controller. El proyecto contiene la clase Control que es la encargada de realizar el control y seguimiento de las formaciones que está controlando el controller. Los behaviours que son No Selectivos, son comportamientos que se han de ejecutar una única vez en el inicio del controlador y para luego quedar en desuso. Por otra parte, los behaviours que son Selectivos pasan a depender del comportamiento que se encarga de la recepción de los mensajes que es ReceiveMessages.

ReceiveMessages es un comportamiento cíclico, esto quiere decir que se ejecuta indefinidamente hasta que el agente finalice su ejecución. Este behaviour realiza la recepción de todos los mensajes que tengan como destino el controller que lo está ejecutando. Obtiene el mensaje que le fue enviado y selecciona qué comportamiento es el adecuado para atender la solicitud. Al igual que un servidor web, este comportamiento se ejecuta de forma paralela en un modelo cliente-

servidor, donde a cada solicitud recibida, el comportamiento que la gestiona responde con la información de control adecuada para mantener la seguridad e integridad de la simulación.

5.3.3 Los comportamientos definidos en el controlador

A continuación, se describen los comportamientos o behaviours que los agentes controladores implementan. Algunos de ellos son utilizados a lo largo de la ejecución de la simulación y otros solo se utilizan para una función especifica

- ***RegisterInDF***: el agent2Controller se registra en el directorio de la arquitectura JADE para que otros agentes se informen de su presencia. Esto es necesario porque de esta manera, JADE permite que los agentes puedan comunicarse e intercambiar información. Los agentes registran en el directorio los comportamientos que ofrece para que otros agentes puedan consumirlos.

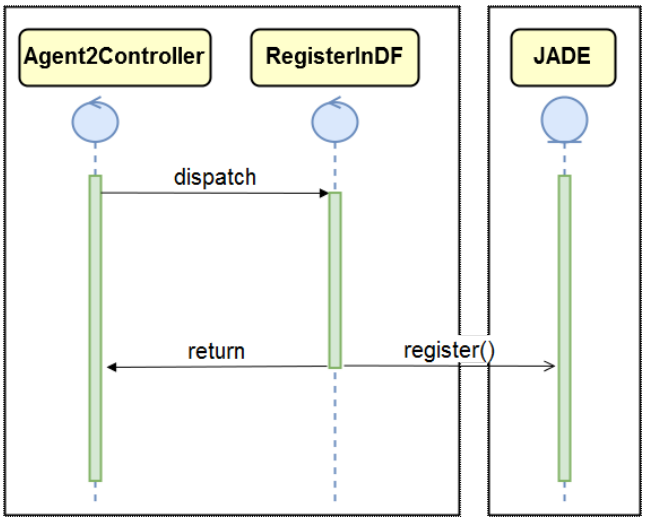


Ilustración 5-16 - Diagrama de secuencia de registro en JADE.

- ***HandleRegisterLinkController***: este comportamiento quita la responsabilidad de ser el último agente del trayecto de vía. De esta manera, el agente que se instancio último, le indicará a su antecesor que ya no es el último de la lista y que ahora el recorrido termina en él. Para esta operación, cuando se registró el nuevo controlador y formaliza la conexión con su antecesor por medio de este comportamiento configura la variable de instancia aimLatestController en true. El agente nuevo, luego de haberse registrado con su antecesor, guarda los datos de su antecesor en controllerLinked.

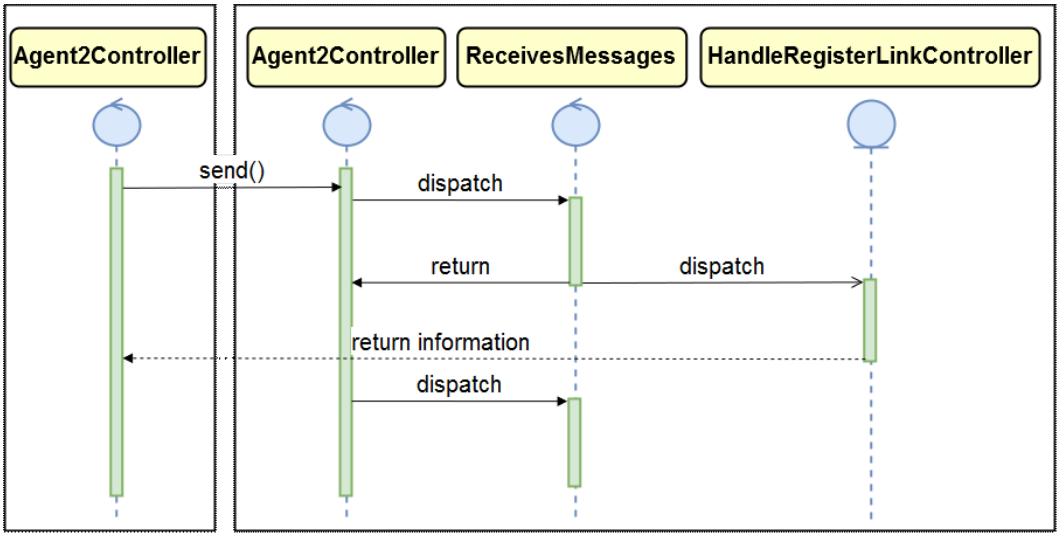


Ilustración 5-17 - Diagrama de secuencia de enlace de controladores.

- ***HandleRegisterTrain***: una vez que un tren se registra en un controlador, este envía la notificación al controlador visual de la presencia de una nueva unidad en el circuito de vía. Al tren que solicitó el registro se le devuelve la información del recorrido. La notificación al agente que muestra la información es en un solo sentido y no se espera respuesta alguna. Para tener un control de las formaciones registradas, se almacena la posición del tren en un arreglo sortedRecords para una rápida consulta como también en la colección environments. Esta colección contiene la información de cada formación de la última comunicación realizada.

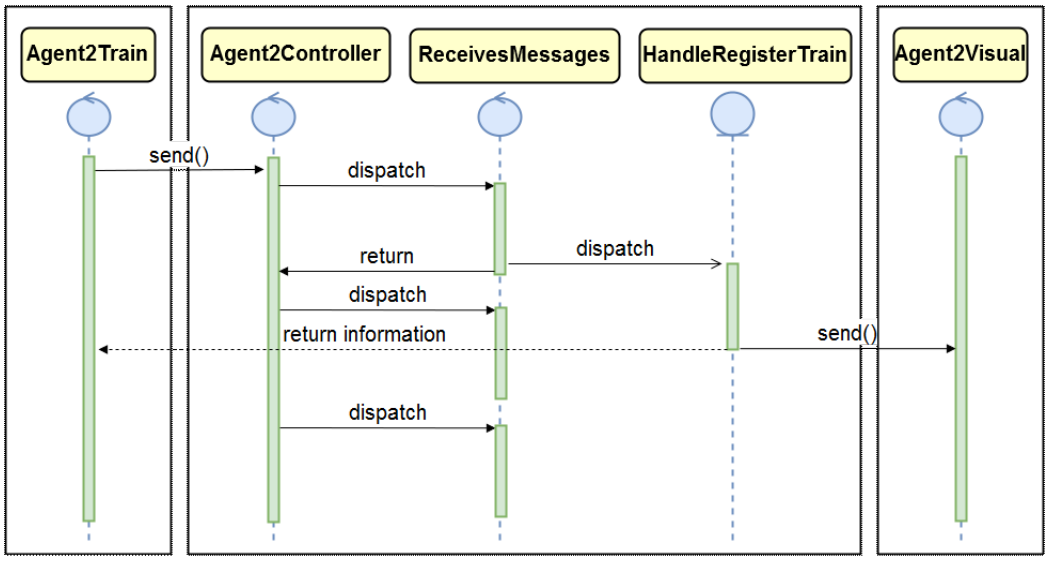


Ilustración 5-18 - Diagrama de secuencia de registro de una formación.

- ***HandleInitService***: el tren que se encuentra ya registrado y detenido a la espera de poder iniciar el trayecto. Este comportamiento evaluará si el circuito de vía se encuentra apto para que la formación comience su recorrido. Si esta comprobación detecte que una formación se encuentra en los límites aceptables para que inicie el recorrido, devolverá al tren el estado que corresponde al inicio del servicio, TYPE_REGISTER_INIT_SERVICE. Por otro lado, si en la vía se halla una formación detenida por el motivo que fuere o comenzando su recorrido, le enviara que siga detenido, STOPPING_SERVICE. Esta información, no corresponde notificar a la interfaz visual, porque cualquiera se la acción enviada, no provocara un efecto que deba ser visualizado.

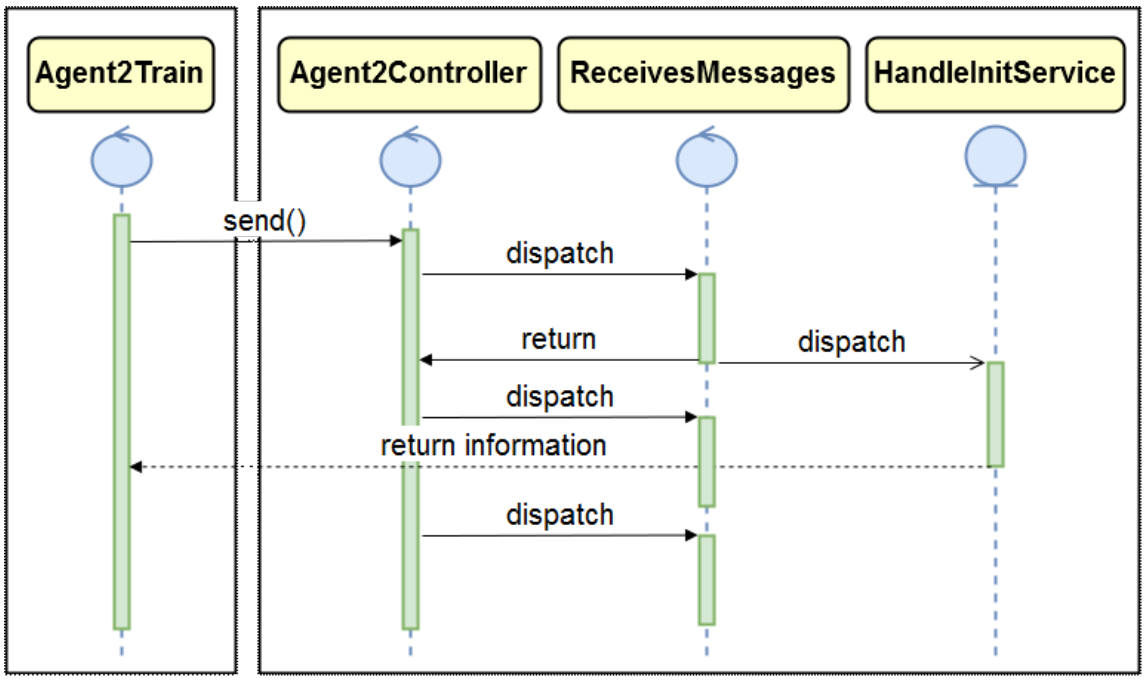


Ilustración 5-19 - Diagrama de secuencia de solicitud de inicio de servicio.

- ***HandlePosition***: es el comportamiento que más utiliza el controller, debido a que se utiliza para registrar la posición de la formación en la interfaz visual y controlar la formación en base a la información recibida.

  Una vez que se recibe la información, inmediatamente se envía la información a la interfaz gráfica para que registre el avance. A su vez, se envía a la instancia del objeto Control y evalúe qué información de control se le devolverá a la formación para que no colisione y tampoco pase por alto una estación de detención.

  Los valores que puede retornar son TYPE_APPROACH_SERVICE que indica que la formación se está acercando a otra formación que se encuentra por delante. Este valor devuelto, en general irá acompañado

por la velocidad a la que debe circular o en su defecto que se detenga por completo. Otro valor que puede retornar es TYPE_CONTINUE que le indicará a la formación que siga su funcionamiento normalmente. Los mensajes anteriormente mencionados, tienen su correspondiente en el control de aproximación, Control.approach().

Los siguientes mensajes son retornados por el método Control.stop() que evalúa si la formación se deberá detener o continuar circulando. TYPE_SLOW le indicará a la formación que comience a detenerse. TYPE_OUT_OF_SERVICE será devuelto indicando que la formación termino el recorrido. Si no es el fin del recorrido, estamos en presencia de un cambio de controlador o tramo de vía. Para indicar ello, el control de detención devolverá el valor TYPE_CHANGE_CONTROLLER_SERVICE. Si nada de ello ocurrió, devolverá el mismo mensaje que retorno el control de aproximación, TYPE_CONTINUE.

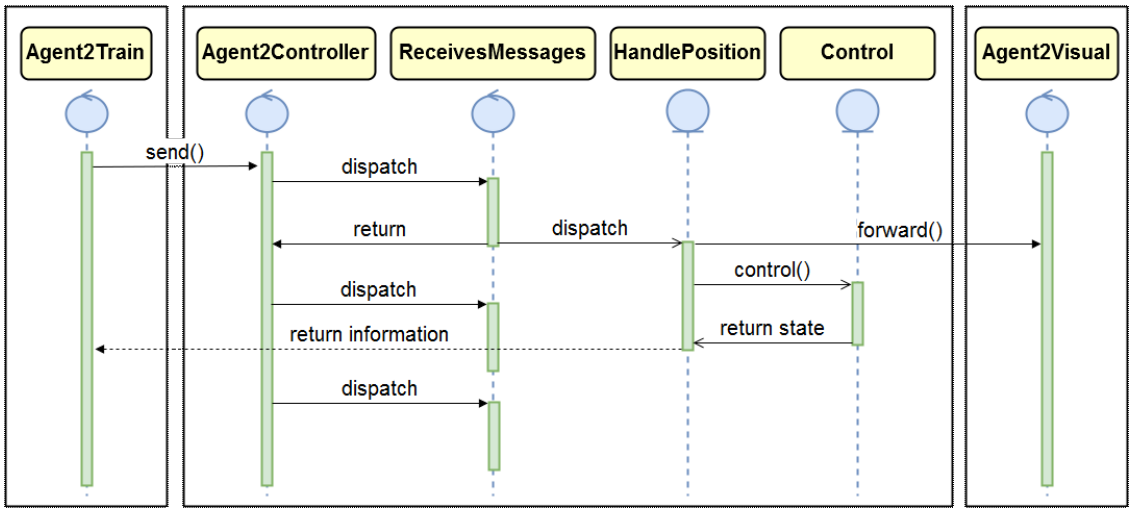


Ilustración 5-20 - Diagrama de secuencia de HandlePosition.

- ***HandleUnRegisterTrain***: este comportamiento quita al tren que lo solicita de la supervisión del controlador que lo recibe, también lo quita de la interfaz visual. El tren será removido de las estructuras sortedRecords y de environments. Luego se le notifica al agente tren que su solicitud se completó y este termina su ejecución.

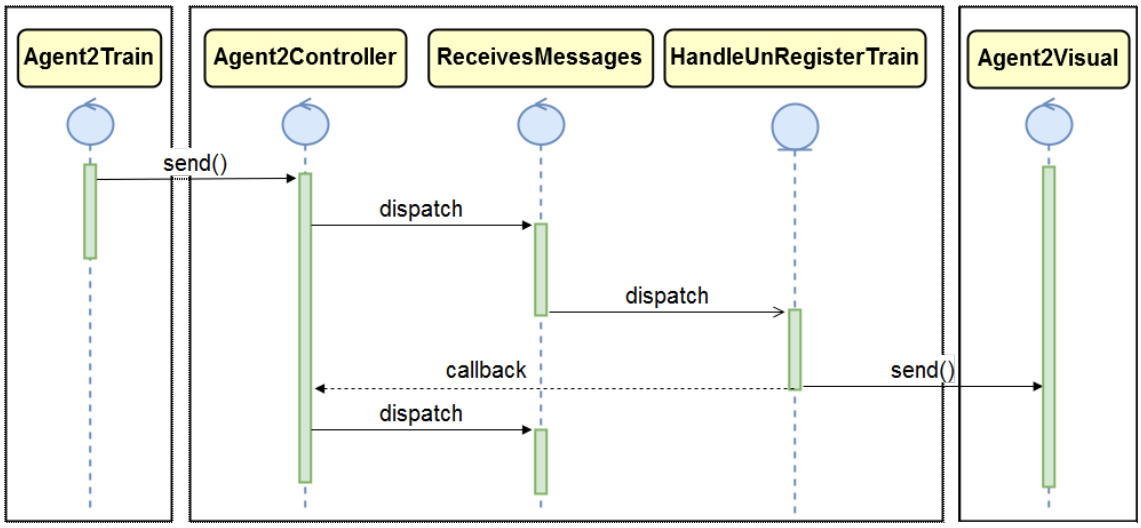


Ilustración 5-21- Diagrama de secuencia de borrado de la formación.

5.3.4 Enlace con la interfaz Visual

Es la primera operación que realiza un agent2Controller. Para ello envía la información del circuito que representa. Esta información contiene el nombre del controlador, la distancia total del recorrido en metros, la cantidad de estaciones que representa y la posición de las mismas dentro del trayecto al igual que las zonas de frenado.

Esta tarea es realizada por el comportamiento del agente linkToAgentVisual. Este comportamiento busca en el DF al agente que corresponde con el agente que representa la interfaz visual para luego enviarle el mensaje con el objeto RegisterController para que proceda al registro y graficar en pantalla la vía, las estaciones, las zonas de frenado y el final del trayecto correspondiente.

De no encontrarse activa la interfaz visual, el agente controlador dará por finalizada su ejecución indicando que no se ha podido conectar al agent2Visual. Recordar que, para ejecutar dicho agente, como cualquier otro de la arquitectura de JADE, debemos tener instanciada la plataforma de JADE, en caso contrario, recibiremos un mensaje definido por JADE indicando que la plataforma no se encuentra en ejecución en la dirección de red indicada.

5.3.5 Enlace con el controlador que lo antecede

Esta tarea es otro comportamiento que tiene el agente controlador luego de haber obtenido la conexión con el Agent2Visual. Nuevamente, el controlador realiza una búsqueda en el DF en busca de un controlador que se encuentre registrado inmediatamente antes que él. Puede ocurrir que existan varios agent2Controller registrados previamente, pero debido a la nomenclatura definida para los nombres de los agent2Controller, ("controlador<número de controlador dentro del trayecto>") rápidamente es identificado. Por ello, la nomenclatura que, para esta versión del simulador, se les otorgó a los controladores, no es un punto de poca importancia.

Si el controlador puede registrar al controlador que envía la solicitud, el controlador receptor, se quita la marca de último controlador y registra que posee un controlador que lo sucede en el circuito conformado.

5.3.6 Registro de formaciones

El registro de formaciones es la primera interacción que se registra entre un controlador y las formaciones ferroviarias. Al crearse los trenes, buscan en la arquitectura JADE la presencia de los controladores, en concreto del primer controlador que se encuentre registrado por su identificación numérica que se encuentra en el nombre del controlador.

Por medio del comportamiento establecido para registrar las formaciones, HandleRegisterTrain, el controlador dará servicio a esta solicitud proveniente del Agent2Train. El controlador crea un objeto de la clase TrainEnvironment donde une al tren que solicita el registro con el Track que este representa. Si bien el controlador representa a un trayecto del circuito, este conjunto, se forma para evitar múltiples accesos simultáneos al objeto Track que contiene el agente controlador. De no ser así, el acceso a la información del Track supondría un posible cuello de botella, con el consiguiente detrimento del sistema en general. Para dar fin al registro de la formación, el controlador, remite al agente tren que realizó la solicitud la información del tramo que controla, en definitiva, el objeto Track, Para completar el registro, se notifica al agent2Visual que una nueva formación se ha registrado.

5.3.7 El objeto controlador o Control

Este objeto es quien verdaderamente realiza la tarea de controlar y garantizar la circulación de las formaciones. Se busca que las tareas que realice sean sencillas y fácilmente comprensibles. Con este preconcepto entre manos, este objeto tiene definidos solo dos métodos, approach() y control(). Ambos métodos reciben información que le fue enviada al agente agent2Controller. Recordar que hay dos estructuras de datos que contienen toda la información sobre los agentes que se encuentran bajo la supervisión de un controlador, sortedRecords y environments.

El método approach()

Este método controla que las formaciones no lleguen a acercarse a una distancia posiblemente peligrosa. Para ello, busca el orden en que se encuentra la formación que es objeto de control de aproximación en la estructura sortedRecords. Si detecta que, en la posición anterior a la suya, se encuentra una formación (desconociendo si está cerca o no), obtiene el objeto correspondiente a la formación en condición de alcance. A este objeto, se le consulta por su posición dentro del circuito de vía. Si la posición es menor o igual a la que posee la cloudBFrontal del tren alcanzante, se pasa a informar al tren alcanzando (el tren que está siendo objeto de evaluación) que disminuya su velocidad a la del tren que está dando alcance, si este último se encuentra en movimiento. En caso de que se encuentre detenido, se le indica que disminuya su velocidad a 0 km/h.

Todo esto, definirá qué mensaje se le enviará al agent2Train en cuestión, porque además de devolver si se encuentra el tren en una situación de alcance o no, mediante los valores true o false, modificará el correspondiente mensaje de respuesta al tren para que tome la acción indicada.

El método que controla la detención, stop()

Este método tiene la responsabilidad de controlar que la formación se detenga en los sitios dispuestos para tal fin. Para comenzar, lo primero que debe detectar este método es si la formación se encuentra en movimiento o no. En este punto, podemos dividir al algoritmo en dos grandes partes.

Para el caso en que la formación se esté moviendo, el algoritmo evalúa si la formación está circulando dentro de una zona de detención y si la distancia necesaria para realizar una detención segura a la velocidad que circula es suficiente. De no alcanzar la distancia, se ordena al tren que comience a detener de inmediato. El término “de no alcanzar” no significa literalmente que la distancia que necesita para detenerse no alcanzara. Porque esta distancia o evaluación se hace sobre el valor ponderado en un 2%. Este 2% parecería ser un valor pequeño, pero para comprenderlo, a continuación, se realiza el cálculo que realiza el método getSpeedAbove() de la clase Train. El método, por defecto y por una cuestión de seguridad de diseño devuelve TRUE, bajo la premisa de que es preferible detener la formación a que ocurra un efecto no deseado.

$$
\begin{aligned}
&\boldsymbol{retorno} = \boldsymbol{TRUE};\\
&\boldsymbol{AVERAGE\_STOP} = \mathbf{1}\,\boldsymbol{m/s}. \qquad //\boldsymbol{aceleración\ promedio\ de\ detención}.\\
&\boldsymbol{DistanciaInicial} = \mathbf{300}\,\boldsymbol{metros}. \qquad //\boldsymbol{distancia\ inicial\ supuesta}\\
&\boldsymbol{Velocidad} = \mathbf{27.7778}\,\boldsymbol{m/s}. \qquad //\boldsymbol{velocidad\ inicial\ supuesta}\\
&\boldsymbol{VelocidadMáximaAceptada} =\\
&\qquad \sqrt[2]{(\boldsymbol{DistanciaInicial} \times \boldsymbol{AVERAGE\_STOP} \times \mathbf{2})};\ //\mathbf{24.4948}\,\boldsymbol{m/s}.\\
&\boldsymbol{si}\ (\boldsymbol{VelocidadMaximaAceptada} > \boldsymbol{Velocidad} \times \mathbf{1.02})\ \ //\mathbf{24.4948}\\
&\qquad > \mathbf{27.7778} \times \mathbf{1.02}\ ?\ \boldsymbol{Falso}\\
&\boldsymbol{retorno} = \boldsymbol{FALSE};\\
&\boldsymbol{sino}\\
&\qquad \boldsymbol{si}\ (\boldsymbol{DistanciaInicial} == \mathbf{0})\ \ //\boldsymbol{Falso}\\
&\qquad\qquad \boldsymbol{retorno} = \boldsymbol{FALSE};\\
&\quad \boldsymbol{return\ retorno};\qquad\qquad //\boldsymbol{Para\ este\ ejemplo, retorna\ TRUE}.
\end{aligned}
$$

Ecuación 5-9 - Ecuación que realiza el Control de controlador para determinar el exceso de velocidad para la detención en un sitio concreto.

Para este ejemplo, el método no estará devolviendo TRUE indicando que la formación debe detenerse inmediatamente. Véase que, la velocidad junto con el promedio de frenado, AVERAGE_STOP, la formación no cuenta con el espacio necesario y no podrá detenerse. Por ello, para la detención es muy

importante la distancia a la que se colocan las zonas de detención previamente a las estaciones.

Con esta valoración en conjunción con la condición de si el tren se encuentra dentro de una zona de detención, el controlador envía al Agent2Train la orden de comenzar a detenerse y además detalla en la colección environments que la formación se halla en el proceso de detención motivado porque delante hay una estación. Para las sucesivas evaluaciones de la condición de detención, se le seguirá ordenando que se detenga, salvo que el flag que contiene la colección environments sea modificada, indicando que no se encuentra en una zona de detención. Cambio muy poco probable, ya que solo podrá cambiar de estado una vez que la formación se halla detenido.

La otra parte del control de stop(), ocurre si la formación ya se encuentra detenida. Lo primero que analiza el algoritmo, es si la mencionada detención se encuentra sobre una estación. De no ser esta la condición, el controlador liberara al tren de estar detenido para que pueda proseguir. Ahora bien, si el tren se encuentra detenido en el lugar de una estación, se le obligará a que realice la notificación un número finito de veces. Para este análisis, el número de veces que se le solicita es STOP_CONFIRMATION que es un atributo definido en la interface Agent2TrainVocabulary y tiene un valor de 10 repeticiones.

Completadas las fases de repetición de la condición de detención sobre la estación, el algoritmo se centra en confirmar a qué estado debe conducir a la formación. Tiene varias alternativas, liberarlo para que continúe, liberarlo para que cambie de controlador o liberarlo para que comience el cierre de servicio.

El tren será liberado para que continúe el servicio dentro del mismo controlador, si el algoritmo detecta que no se ha llegado al final del track. Para ello chequea en que estación se detuvo y si no es la última, simplemente resetea el contador de confirmación de detención, STOPS_CONFIRMATION y envía el mensaje de continuar, TYPE_CONTINUE definido en Agent2TrainVocabulary, a la formación. Para el estado que el tren debe realizar el cambio de controlador, se verifica que la estación es la última de las definidas para este trayecto y que el controlador no es el último controlador no corresponde al último controlador del circuito de vía definido. Siendo estas configuraciones verificadas, se resetea el contador de confirmación de detención y se envía el mensaje de TYPE_CHANGE_CONTROLLER_SERVICE definido en la interface de Agent2TrainVocabulary.

Si las dos condiciones descritas en el párrafo anterior no sucedieron, la última alternativa que nos deja es que el tren se encuentre en la última estación del último controlador. El controlador remitirá el agent2Train el mensaje con el

tipo de respuesta TYPE_OUT_OF_SERVICE indicando que el servicio que cumple ha finalizado.

5.4 El proyecto Agent2Train

Este proyecto es el tren propiamente dicho. Este agente será el que recorre el circuito de vía, pasando de un controlador a otro, además de encontrarse en distintos puntos del controlador. Cuenta con distintos comportamientos como conectarse al controlador, solicitar el inicio del servicio, etc. que veremos a continuación.

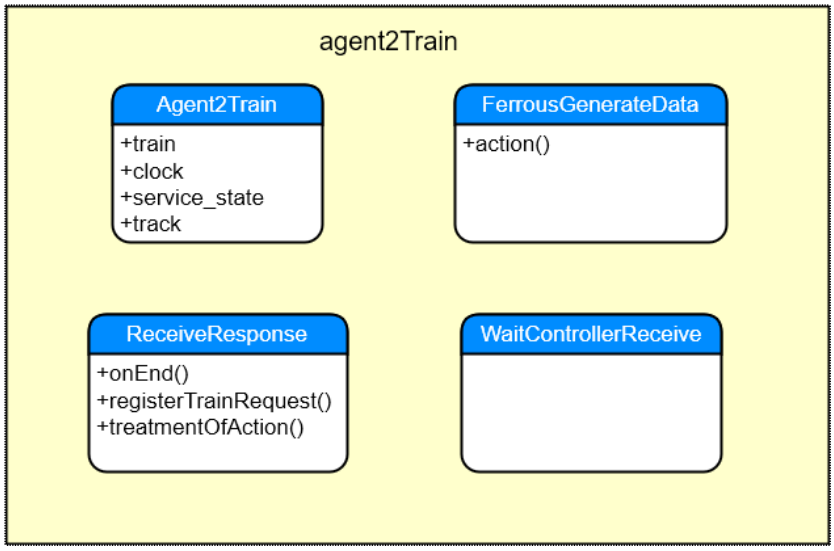


Ilustración 5-22 - Clases del proyecto agent2Train.

Para instanciar un agente del tipo agent2Train se necesita configurar los parámetros de funcionamiento. Estos parámetros son la longitud de la formación que simulará, la velocidad de la simulación con la que se desee trabajar, el formato de funcionamiento del agente asistido o aleatorio, el tiempo de demora para poder enviar mensajes, el modo de operación y, por último, el tiempo que deberá detenerse en las estaciones.

Descripción de los parámetros para instanciar una formación

Aquí, el detalle de los parámetros que son aceptados por el Agent2Train al momento de instanciarse:

train4:agent2train.Agent2Train(

“velocidadMaxima”, //velocidad máxima a la que puede circular la formación.

“velocidadSimulacion”, //velocidad de simulación.

“Automático”, //espera o no en las estaciones por la tecla ENTER.

“timeLatch”, //tiempo en milisegundos que realiza las comunicaciones.

“MODOAproximación”, //nube dinámica o estática.

“lapsoDetenciónPromedio”) //tiempo de detención promedio en las estaciones.

De estos parámetros, unos funcionarán directamente sobre el agente y otros pasarán al objeto Train para que configure los parámetros de ejecución de la formación a simular. El comportamiento principal de este agente es el denominado FerrousGenerateData, que basándose en los estados del agente decidirá qué métodos deben ejecutarse.

5.4.1 El comportamiento FerrousGenerateData

Una vez creadas las estructuras internas de control, el agente buscará conectarse con el primer controlador que figura en el DF de la arquitectura JADE o directorio de servicios ofrecidos por agentes. El método utilizado en esta ocasión es registerTrain(). Una vez que se establece la conexión entre el controlador y el agente, el agente recibirá la información del Track que tiene que recorrer y almacena quien es su controlador actual para este circuito de vía.

Este behaviour FerrousGenerateData, funciona en paralelo con la recepción de los mensajes, alternadamente con el servicio de envío de información, con la salvedad de que el mensaje se generará o no dependiendo del tiempo del latch para volver a dar paso al comportamiento de FerrousGenerateData.

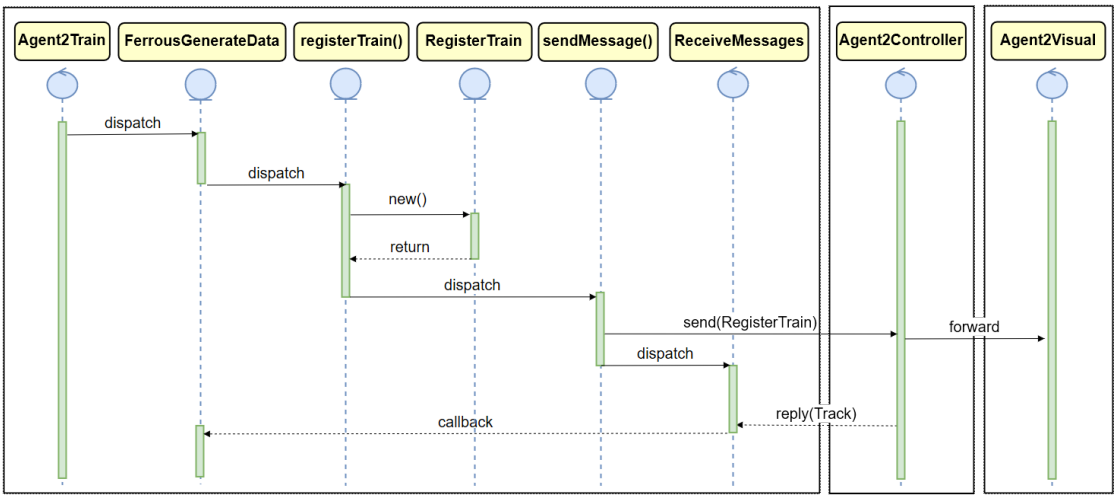


Ilustración 5-23 - Registro del tren en el controlador.

Luego de registrarse el agente tren en el controlador, solicita permiso para iniciar el recorrido del trayecto. Si la petición es aceptada, se retorna el valor definido en la interface Agent2TrainVocabulary TYPE_CONTINUE para que el tren comience su recorrido por el circuito. En caso de que no se permita el inicio del servicio, se devuelve el valor de TYPE_REGISTER_INIT_SERVICE que tornara a la formación al estado correspondiente para que vuelva a solicitar el inicio del servicio. Esto provocará que se demore a la formación el tiempo indicado por el Latch antes de que vuelva a realizar la solicitud de inicio de servicio. Esta petición la realiza mediante el método serviceInitiationRequest().

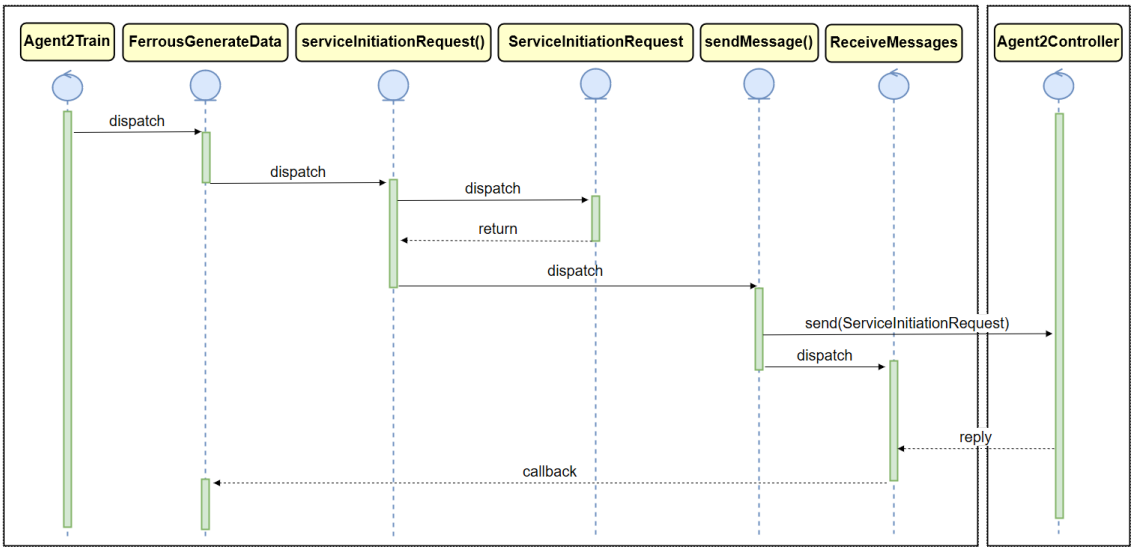


Ilustración 5-24- Solicitud de inicio de servicio desde el tren.

Una vez aceptado la solicitud de iniciar el servicio por parte de agente controlador, este le remite al agente tren el estado TYPE_CONTINUE. Al recibir

este estado, el tren comenzará su servicio ejecutando el método generateAdvance(), que a través del objeto Train que tiene definido, genera la información correspondiente al estado del tren como velocidad, distancia recorrida, aceleración, etc. e intentara, si el timeLatch se lo permite, remitir esa información generada al controlador.

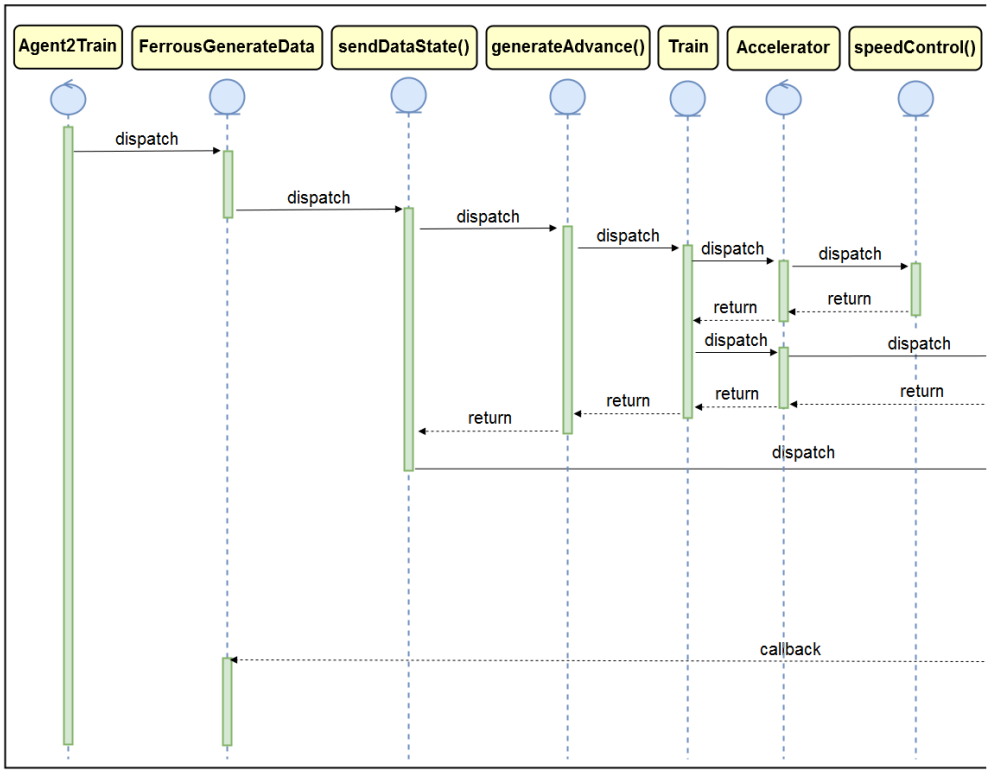


Ilustración 5-25 - Envío de la posición desde el agente tren al controlador (parte A).

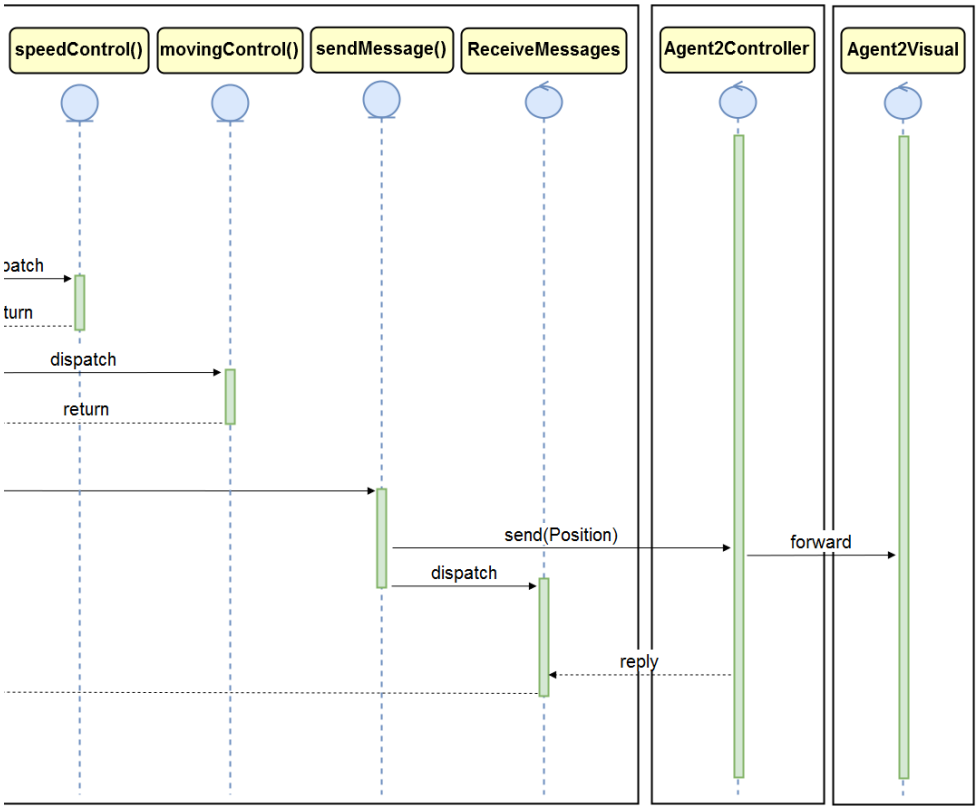


Ilustración 5-26 - Envío de la posición desde el agente tren al controlador (parte B).

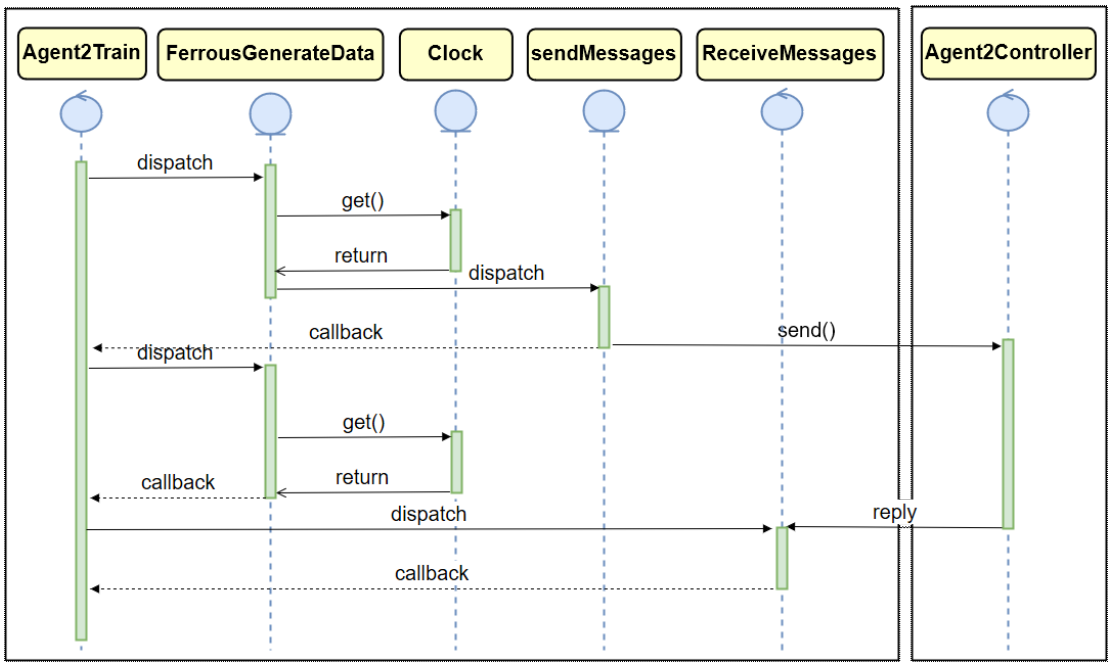


Ilustración 5-27 - Diagrama general de envío de información.

5.4.2 El sistema de envío de mensajes

El sistema de mensajería que utiliza el agente está controlado por un objeto Clock que limita o restringe el tiempo que transcurre entre el envío de un mensaje y el posterior. El modelo de funcionamiento es similar encuadra en un paradigma de productor consumidor. En el instante que el productor genera un clock, habilita la interfaz de envío de mensajes. En cambio, cuando el consumidor lo consume, permite la recepción de mensajes.

De esta forma se gestiona el tiempo del que disponen los agent2Train para hacer uso de la red y se controla que cuando se va enviar un mensaje paralelamente no se está recibiendo otro que modifique el estado de la formación. También, se logra emular el funcionamiento de equipos que se utilizan para este tipo de tareas.

En su ejecución, el agente solicita por medio del método get() del objeto Clock el permiso para realizar el envío del mensaje. Si tiene éxito, el clock le devolverá el valor solicitado a modo de token para completar el envío. A su vez, el método get() notifica al objeto clock que se consumió el valor producido y que necesita que produzca uno nuevo. En caso contrario, el agente continúa con la ejecución y en la próxima oportunidad que tenga para realizar el envío repetirá la acción. El ThreadClock funciona a modo de productor ejecutando el método set() del clock. Para ello se demora el tiempo que se le ha configurado a la formación en el parámetro Latch. Este tiempo puede ser de entre 10 segundos hasta 500 segundos, lo que con un tiempo tan alto es probable que no se busque una performance del circuito de vía.

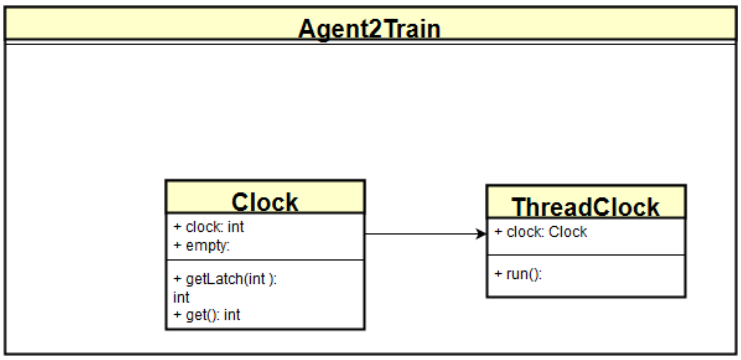


Ilustración 5-28 - Modelo de clases Clock y ThreadClock

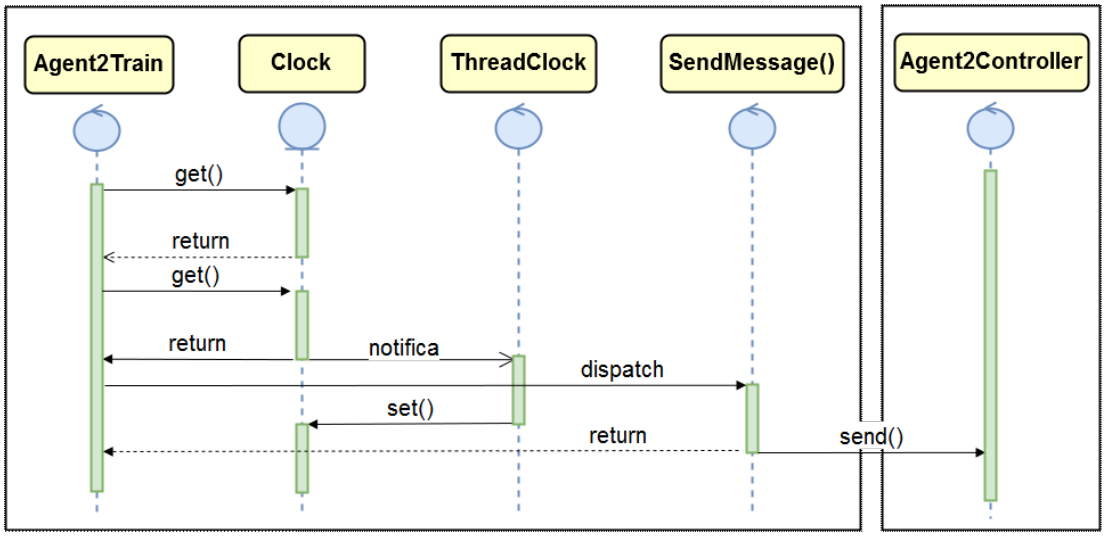


Ilustración 5-29 - Diagrama de secuencia de Agent2Train, Clock y ThreadClock.

6 RESULTADOS

En los últimos años en nuestro país, hemos conocidos una serie de accidentes que han marcado la historia ferroviaria de Argentina. Este tipo de acontecimientos, motivó que el servicio que ofrecen las prestadoras sea objeto de análisis e informes que demostraron un conjunto de falencias y disconformidad de los usuarios. Uno de los accidentes más importantes que padeció la sociedad fue el hecho ocurrido en febrero de 2012, donde más allá el problema que originó el siniestro, se detectó que la formación contenía un número de pasajeros superior a la capacidad permitida, siendo esto, un indicativo de un servicio ineficiente.

En esto se centra el presente trabajo, la seguridad de las formaciones que circulan en un circuito de vía y el uso eficiente de la infraestructura y el material rodante. Otorgando como resultado directo una mayor frecuencia del servicio con un número mayor de formaciones en la vía. El aumento del número de formaciones se traduce a nivel de los usuarios en un servicio de calidad, donde cada uno de estos cuente con su correspondiente plaza, como según rigen las normas de seguridad exigidas por el sistema de transporte o CNRT (Comisión Nacional de Regulación del Transporte) [Link33].

En la sección 6.1 del presente capitulo, se explica detalladamente el análisis llevado a cabo para validar el simulador con el entorno real. En la sección 6.2, se muestra cual es el verdadero objetivo del prototipo presentado, exponiendo su capacidad de conservar la seguridad de las formaciones dentro de un circuito de vía propuesto.

6.1 ANÁLISIS DE LA INFORMACIÓN CON UN MODELO REAL

Para el análisis de este trabajo, se consultó la fuente de información provista por la sociedad Operadora Ferroviaria Sociedad del Estado (SOFSE) [Link34]. En el portal web de la mencionada sociedad, en el apartado “Trenes En Directo | Mapas en Directo”, se encuentra un mapa interactivo en el cual, los usuarios pueden consultar la posición en la que se encuentran las formaciones ferroviarias. A continuación, se muestran las imágenes que corresponden con el acceso al mapa y el mapa mencionado.

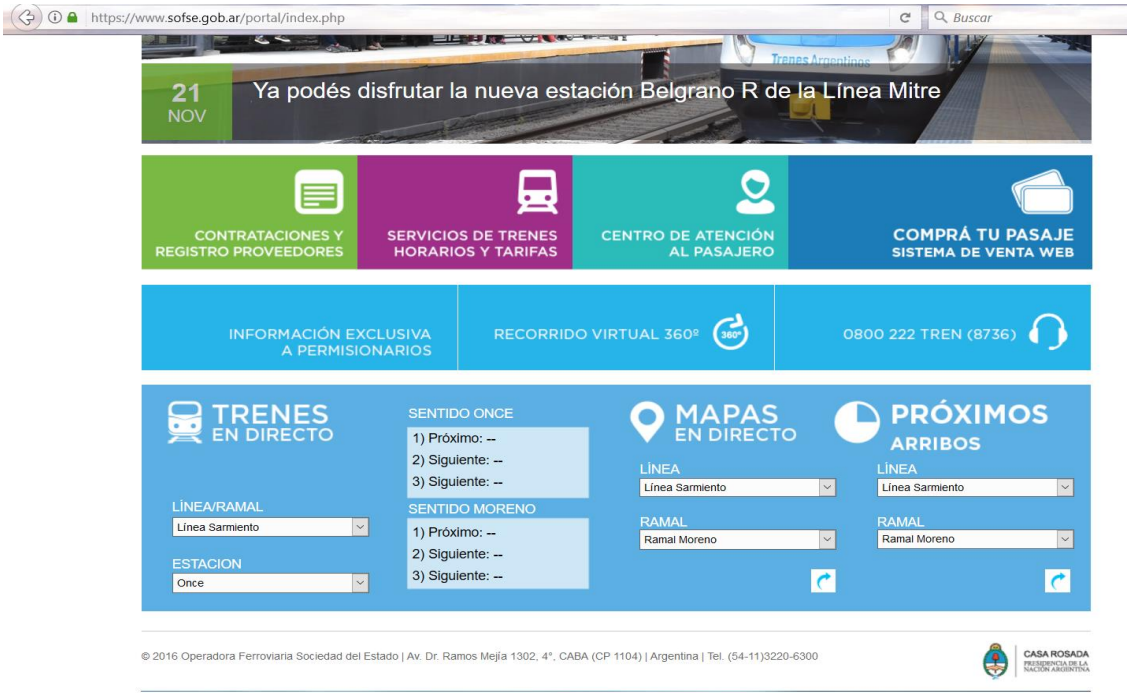


Ilustración 6-1 - Portal de acceso al mapa interactivo del servicio ferroviario.

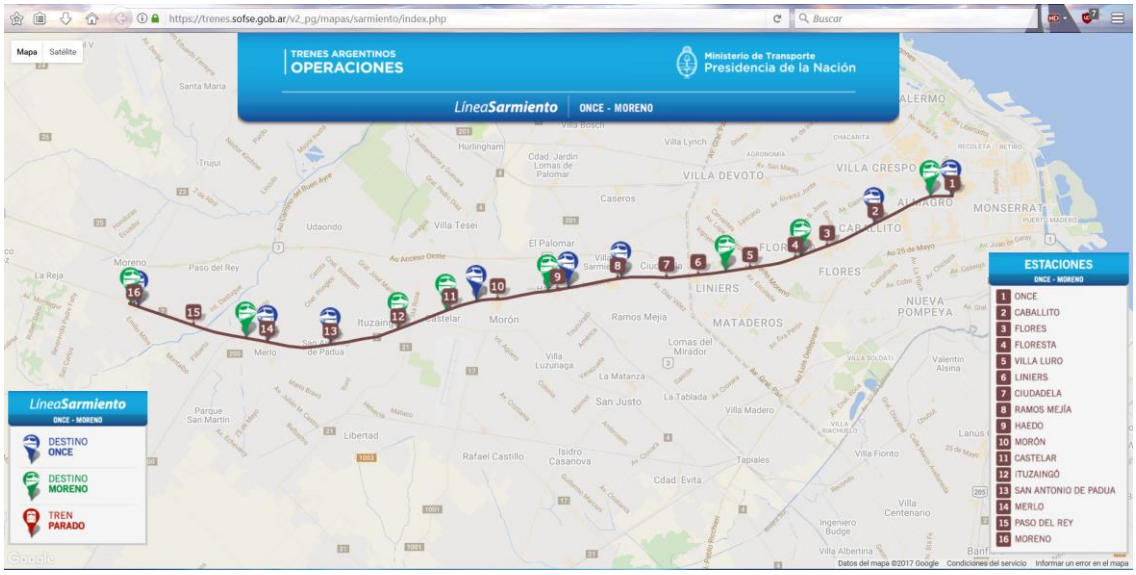


Ilustración 6-2 - Mapa interactivo, propiedad SOFSE.

En la Ilustración 6-2 pueden apreciarse las formaciones están en funcionamiento en la vía. En la leyenda que está en el borde inferior izquierdo, se detalla el sentido en que están circulando las formaciones. Para el momento en que se capturó esta imagen, 25/01/2017 a las 09:53 horas, se tiene un total de 16 trenes circulantes. Solo se tienen 8 trenes que circulan en sentido a

Moreno y 8 trenes con destino Once para un total de 16 estaciones entre Moreno y Once.

Para obtener resultados y poder compararlos con el entorno real, fue necesario tomar medidas para dar valores a los distintos parámetros como tiempo de detención en la estación, velocidad promedio de desplazamiento de las formaciones, tiempo total en realizar el recorrido por completo, etc. (entre la estación de Plaza Miserere (ONCE) y la estación de MORENO). La medición se realizó por medio de la herramienta Google Maps. El resultado obtenido se muestra en la Ilustración 6-3 en la tabla que contiene las distancias obtenidas para las estaciones y en la Tabla 6-1 se reflejaron los datos puntuales.

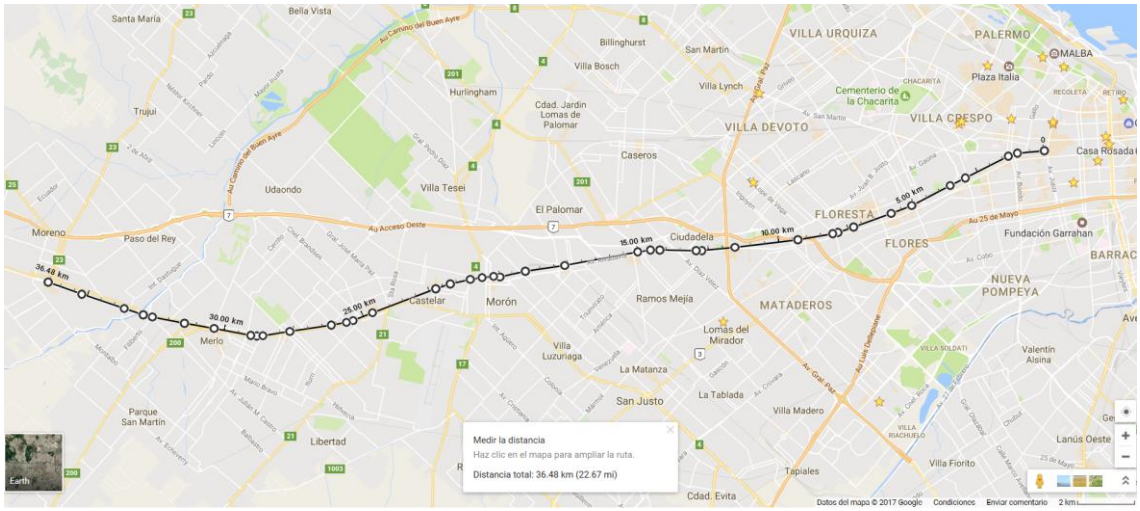


Ilustración 6-3 - Mapa utilizado para cuantificar la distancia de cada estación.

Recorrido desde estación Once a estación Moreno.		
Número de estación	Nombre de la Estación	Distancia desde la estación Once
1	**Once**	**0.000 metros**
2	Caballito	3.590 metros
3	**Flores**	**5.900 metros**
4	Floresta	7.320 metros
5	**Villa Luro**	**9.330 metros**
6	Liniers	11.570 metros
7	**Ciudadela**	**12.940 metros**
8	Ramos Mejía	15.020 metros
9	**Haedo**	**17.640 metros**
10	Morón	20.180 metros
11	**Castelar**	**22.270 metros**
12	Ituzaingó	24.670 metros
13	**San Antonio de Padua**	**27.670 metros**
14	Merlo	30.730 metros
15	**Paso del Rey**	**33.630 metros**
16	Moreno	36.480 metros

Tabla 6-1 - Tabla de distancia de las estaciones.

Cuando comparamos los datos aportados por el simulador con la traza que disponemos de control, podemos inferir que el sistema ferroviario real que actualmente se encuentra funcionando para el ramal estación Once-Moreno, no hace un uso eficiente de sus formaciones y de sus circuitos de vías. Si bien, en la versión simulada no se cuenta con los distintos accidentes geográficos por lo que transitan las formaciones reales, el simulador arroja resultados no muy distantes con los obtenidos del modelo real.

La información con la que se realizó el análisis, se obtuvo en base a un video generado [Video6-1] a partir del mapa interactivo mostrado en la Ilustración 6-3. Este análisis arrojó velocidad promedio de circulación, tiempo total empleado para el recorrido, tiempo que las formaciones suelen detenerse, etc. En la tabla 6-2, 6-3 y 6-4 se deja al lector los valores obtenidos del modelo real.

Haciendo un resumen de dicha tabla, encontramos que las formaciones reales tienen un desempeño de velocidad promedio de 10,74 metros/segundo. Esto se traduce a una velocidad de 38,66 km/h. El simulador, con diversos valores de cada una de las formaciones de velocidad final y tiempos de detención en las estaciones, marcó una velocidad promedio de 44,92 km/h para hacer el mismo recorrido. Esto supuso una mejora para el simulador de un 16% en la velocidad promedio, además de acercar las formaciones a los usuarios con una frecuencia eficiente y útil.

Para los tiempos de detención, entonces el servicio del tren Sarmiento deja una demora promedio de 84,6 segundos. Siendo que el simulador, obtuvo un tiempo de detención por estación bastante inferior, 48,7 segundos. Nuevamente, el simulador vuelve a hacer un uso eficaz de los recursos, empleando para la detención de la formación un tiempo inferior al tiempo real medido. Si se observa la tabla con los valores reales, se aprecia que en varias ocasiones el tren se demora menos del promedio que entrega el simulador, llegando incluso a estar detenido escasos 24 segundos.

Esto puede ser relacionado con la predisposición que tienen los usuarios a la irregularidad de la frecuencia que tiene el servicio real, evento que el simulador no contempla, pero que si refleja y corrige. Otro aspecto que se observa, en ciertos instantes, en el servicio real, es que las formaciones se encuentran detenidas en sectores de vías entre estaciones, muy frecuentemente en la zona de la estación de Once y la que la sucede.

Resultados obtenidos a partir del análisis del [video 6.1].

estaciones		Estación 1		Estación 2		Estación 3		Estación 4		Estación 5		Estac	
trenes desde/hasta		arranca	se detiene	arranca	se detiene	arranca	se detiene	arranca	se detiene	arranca	se detiene	arranca	se detiene
		3590		2310		1420		2010		2240			
2	16	0	30	216	60	138	45	162	54	162	69		
				10,69444444		10,28985507		12,40740741		13,82716049			
5	16	0	0	0	0	0	0	0	57	159	57		
										14,08805031			
8	16	0	0	0	0	0	0	0	0	0	0		
10	16	0	0	0	0	0	0	0	0	0	0		
12	16	0	0	0	0	0	0	0	0	0	0		
14	16	0	0	0	0	0	0	0	0	0	0		
2	16	498	54	171	63	138	24	174	51	123	75		
		7,208835341		13,50877193		10,28985507		11,55172414		18,21138211			
1	13	621	43	203	45	135	41	165	33	168	63		
		5,78099839		11,37931034		10,51851852		12,18181818		13,33333333			
1	11	870	38	159	45	126	36	153	36	168	60		
		4,126436782		14,52830189		11,26984127		13,1372549		13,33333333			
1	11	708	45	204	51	138	39	189	54	165	54		
		5,070621469		11,32352941		10,28985507		10,63492063		13,57575758			
1	9	624	60	156	54	117	45	147	45	2544	54		
		5,753205128		14,80769231		12,13675214		13,67346939		0,880503145			
1	7	486	102	2577	60	138	42	159	60	153	45		
		7,386831276		0,896391153		10,28985507		12,64150943		14,64052288			
1	5	2886	48	153	51	114	54	135	48		0		
		1,243936244		15,09803922		12,45614035		14,88888889					
Promedios por estación		956	55,7	518	53,6	131	40,8	161	47,6	455	59,6		
		5,224409233		11,52956009		10,94258407		12,63962412		12,7362554			

2	16	0	30	0	30	4	6	5	6	7	24	8	9	10	51	11	45	14	27
5	16	0	0	0	0	0	0	0	0	0	0	0	0	57	0	159	57	3	36
8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	16	498	54	171	63	138	24	174	51	123	75								
1	13	621	43	203	45	135	41	165	33	168	63								
1	11	870	38	159	45	126	36	153	36	168	60								
1	11	708	45	204	51	138	39	189	54	165	54								
1	9	624	60	156	54	117	45	147	45	2544	54								
1	7	486	102	2577	60	138	42	159	60	153	45								
1	5	2886	48	153	51	114	54	135	48		0								

Tabla 6-2- Resultados obtenidos a partir del análisis del video[6.1] – Parte A.

Resultados obtenidos a partir del análisis del [video 6.1].

Estación 6				Estación 7				Estación 8				Estación 9				Estación 10				Estación 11				Esta	
se detiene		arranca		se detiene		arranca		se detiene		arranca		se detiene		arranca		se detiene		arranca		se detiene		arranca		se detiene	
240		1370				2080				2620				2540				2090				2400			
14	27	15	36	19	9	20	48	23	33	25	6	29	30	30	39	34	18	35	15	38	12	39	51	43	48
69		213		99		165		93		264		69		219		57		177		99		237		48	
716049		6,431924883				12,60606061				9,924242424				11,59817352				11,8079096				10,12658228			
3	36	4	33	8	15	9	42	12	33	13	36	17	30	18	36	22	36	23	30	26	51	28	0	33	3
57		222		87		171		63		234		66		240		54		201		69		303		54	
805031		6,171171171				12,16374269				11,1965812				10,58333333				10,39800995				7,920792079			
								1	0	1	57	5	57	6	54	10	30	11	45	15	6	16	54	21	54
0		0		0		60		57		240		57		216		75		201		108		300		42	
						34,66666667				10,91666667				11,75925926				10,39800995				8			
																		0	48	3	49	5	52	11	31
0		0		0		0		0		0		0		0		48		181		123		339		66	
																		11,54696133				7,079646018			
																								2	27
0		0		0		0		0		0		0		0		0		0		0		147		48	
																						16,32653061			
0		0		0		0		0		0		0		0		0		0		0		0		0	
21	36	22	51	25	57	26	45	30	0	30	45	35	6	35	51	39	42	40	45	44	18	47	24	52	36
75		186		48		195		45		261		45		231		63		213		186		312		48	
138211		7,365591398				10,66666667				10,03831418				10,995671				9,812206573				7,692307692			
32	26	33	29	36	38	37	48	40	44	42	14	46	17	47	11	51	20	52	29	55	38	58	58	103	19
63		189		70		176		90		243		54		249		69		189		200		2661		36	
333333		7,248677249				11,81818182				10,781893				10,20080321				11,05820106				0,901916573			
45	35	46	35	49	47	51	59	54	44	55	2	100	47	101	53	107	2	108	29	111	59	114	20		
60		192		132		165		18		2745		66		309		87		210		141				0	
333333		7,135416667				12,60606061				0,954462659				8,220064725				9,952380952							
55	0	55	54	58	57	100	51	104	6	104	51	108	6	108	57	112	51	113	45	116	54				
54		183		2514		195		45		195		51		234		54		189				0		0	
575758		7,486338798				10,66666667				13,43589744				10,85470085				11,05820106							
101	38	102	32	105	11	106	35	109	47	110	44	114	32	115	35										
54		159		84		192		57		228		63				0		0		0		0		0	
503145		8,616352201				10,83333333				11,49122807															
111	17	112	2	115	53	117	5																		
45		231		72				0		0		0		0		0		0		0		0		0	
052288		5,930735931																							
0		0		0		0		0		0		0		0		0		0		0		0		0	
59,6		197		388		165		58,5		551		58,9		243		63,4		195		132		614		48,9	
62554		7,048276037				14,50342238				9,842410704				10,60171513				10,75398506				8,292539322			

Tabla 6-3- Resultados obtenidos a partir del análisis del video[6.1] – Parte B.

Resultados obtenidos a partir del análisis del [video 6.1].																								velocidad promedio	detencion promedio
ión 11		Estación 12				Estación 13				Estación 14				Estación 15				Estación 16							
arranca		se detiene		arranca		se detiene		arranca		se detiene		arranca		se detiene		arranca		se detiene						metros/ segundos	segundos
2400				3000				3060				2900				2850									
39	51	43	48	44	36	47	51	48	54	52	48	53	42	58	3	58	45	102	21						
237		48		195		63		234		54		261		42		2616								10,74113285	63
10,12658228				15,38461538				13,07692308				11,11111111				1,089449541									
28	0	33	3	33	57	37	12	38	0	42	27	43	33	47	51	48	56	56	29						
303		54		195		48		267		66		258		65		453								10,62715192	49
7,920792079				15,38461538				11,46067416				11,24031008				6,291390728									
16	54	21	54	22	36	26	45	27	18	31	27	32	21	36	36	37	27	42	9						
300		42		249		33		249		54		255		51		282								13,50632044	59,625
8				12,04819277				12,28915663				11,37254902				10,10638298									
5	52	11	31	12	37	16	28	17	34	22	25	23	13	28	13	29	7	34	49						
339		66		231		66		291		48		300		54		342								10,02151404	67,5
7,079646018				12,98701299				10,51546392				9,666666667				8,333333333									
		2	27	3	15	6	42	7	54	11	24	13	18	17	9	18	6	23	18						
147		48		207		72		210		114		231		57		312								13,41588815	72,75
16,32653061				14,49275362				14,57142857				12,55411255				9,134615385									
										2	57	5	30	9	36	10	27	15	18						
0		0		0		0		177		153		246		51		291								12,95685597	102
								17,28813559				11,78861789				9,793814433									
47	24	52	36	53	24	57	24	58	6	101	39	102	24	107	0	107	48	112	48						
312		48		240		42		2613		45		276		48		300								10,06797601	59,78571429
7,692307692				12,5				1,171067738				10,50724638				9,5									
58	58	103	19	103	55	107	43	108	28	113	1	113	49												
2661		36		228		45		273		48				0		0								9,863462202	64,38461538
0,901916573				13,15789474																					
114	20																								
		0		0		0		0		0		0		0		0								9,526355378	65,9
0		0		0		0		0		0		0		0		0								10,4396489	323
0		0		0		0		0		0		0		0		0								9,774066964	57,75
0		0		0		0		0		0		0		0		0								8,63097429	63,5
0		0		0		0		0		0		0		0		0								10,92175117	50,25
614		48,9		221		52,7		539		72,8		261		52,6		657								10,74003193	84,59122914
8,292539322				13,70786927				11,48183567				11,17723053				7,7498552									

Tabla 6-4- Resultados obtenidos a partir del análisis del video[6.1] – Parte C.

La siguiente tabla, contiene los datos obtenidos a partir de una simulación en el simulador. Para cotejar estos valores, el video [Video6-2] y el archivo XML [Archivo6.1] generado a partir de la simulación quedan como respaldo de la información.

Resultados obtenidos mediante el uso del simulador.

estaciones		Estación 1		Estación 2		Estación 3		Estación 4		Estación 5		Estaci
trenes desde/hasta		arranca	se detiene	arranca	se detiene	arranca	se detiene	arranca	se detiene	arranca	se detiene	
		3590		2310		1420		2010		2240		
1	16	0 0	5 14	5 53	9 18	9 59	12 10	12 49	15 50	16 29	19 49	
		314	39	205	41	131	39	181	39	200	46	
		11,43312102		11,26829268		10,83969466		11,10497238		11,2		
1	16	9 59	14 33	15 24	18 25	19 15	21 12	22 6	24 45	25 37	28 33	
		274	51	181	50	117	54	159	52	176	56	
		13,10218978		12,76243094		12,13675214		12,64150943		12,72727273		
1	14	20 5	24 53	25 53	29 4	30 4	32 6	33 6	35 54	36 54	39 59	
		288	60	191	60	122	60	168	60	185	66	
		12,46527778		12,09424084		11,63934426		11,96428571		12,10810811		
1	13	30 2	34 35	35 25	38 25	39 15	41 12	42 2	44 41	45 30	48 26	
		273	50	180	50	117	50	159	49	176	57	
		13,15018315		12,83333333		12,13675214		12,64150943		12,72727273		
1	12	40 8	44 11	44 52	47 35	48 16	50 2	50 43	53 7	53 46	56 25	
		243	41	163	41	106	41	144	39	159	49	
		14,77366255		14,17177914		13,39622642		13,95833333		14,08805031		
1	8	50 1	54 34	55 38	58 38	59 35	61 32	62 32	65 11	66 10	69 6	
		273	64	180	57	117	60	159	59	176	63	
		13,15018315		12,83333333		12,13675214		12,64150943		12,72727273		
1	4	60 0	65 35	66 7	69 46	70 17	72 36	73 6	76 19	76 45		
		335	32	219	31	139	30	193	26			
		10,71641791		10,54794521		10,21582734		10,41450777				
1	3	70 31	74 42	75 23	78 1	78 51						
		251	41	158	50							
		14,30278884		14,62025316								
Promedios por estación		281	47,3	185	47,5	121	47,7	166	46,3	179	56,2	
		13,19923756		12,64145108		11,78590701		12,1952325		12,59632943		

Tabla 6-5- Resultados obtenidos a partir del análisis de los datos aportados por el simulador. Parte A.

Resultados obtenidos mediante el uso del simulador.

Estación 6				Estación 7				Estación 8				Estación 9				Estación 10				Estación 11			
se detiene		arranca		se detiene		arranca		se detiene		arranca		se detiene		arranca		se detiene		arranca		se detiene		arranca	
40		1370				2080				2620				2540				2090				24	
19	49	20	35	22	42	23	22	26	28	27	7	30	59	31	39	35	24	36	2	39	10	40	8
46		127		40		186		39		232		40		225		38		188		58		206	
,2		10,78740157				11,1827957				11,29310345				11,28888889				11,11702128				11,650	
28	33	29	29	31	22	32	12	34	56	35	44	39	7	39	57	43	15	44	5	46	50	47	45
56		113		50		164		48		203		50		198		50		165		55		187	
27273		12,12389381				12,68292683				12,90640394				12,82828283				12,66666667				12,83	
39	59	41	5	43	4	44	3	46	56	47	57	51	32	52	31	55	59	57	2	59	55	61	1
66		119		59		173		61		215		59		208		63		173		66		198	
10811		11,51260504				12,02312139				12,18604651				12,21153846				12,08092486				12,121	
48	26	49	23	51	16	52	6	54	50	55	39	59	2	59	54	63	11	64	2	66	46	67	42
57		113		50		164		49		203		52		197		51		164		56		187	
27273		12,12389381				12,68292683				12,90640394				12,89340102				12,74390244				12,83	
56	25	57	14	58	57	59	36	62	4	62	43	65	45	66	23	69	20	70	0	72	29	73	16
49		103		39		148		39		182		38		177		40		149		47		168	
05031		13,30097087				14,05405405				14,3956044				14,35028249				14,02684564				14,285	
69	6	70	9	72	3	73	2	75	47	76	49												
63		114		59		165		62															
27273		12,01754386				12,60606061																	
56,2		115		49,5		167		49,7		207		47,8		201		48,4		163		56,4		189	
32943		11,97771816				12,53864757				12,73751245				12,71447874				12,52707218				12,745	

Tabla 6-6- Resultados obtenidos a partir del análisis de los datos aportados por el simulador. Parte B.

Resultados obtenidos mediante el uso del simulador.

ión 11		Estación 12		Estación 13		Estación 14		Estación 15		Estación 16		velocidad promedio	detencion promedio
arranca		se detiene		arranca	se detiene	arranca	se detiene	arranca	se detiene	arranca	se detiene	metros/ segundos	segundos
2400				3000		3060		2900		2850			
40	8	43	34	44	12	48	36	49	18	53	46	54	28	58	44	59	23	63	35		
206		38		264		42		268		42		256		39		252				11,22513226	41,42857143
11,65048544				11,36363636				11,41791045				11,328125				11,30952381					
47	45	50	52	51	42	55	32	56	20	60	15	61	5	64	48	65	38	69	18		
187		50		230		48		235		50		223		50		220				12,79940509	51
12,8342246				13,04347826				13,0212766				13,0044843				12,95454545					
61	1	64	19	65	20	69	24	70	27	74	32	75	31								
198		61		244		63		245		59										12,20110303	61,30769231
12,12121212				12,29508197				12,48979592													
67	42	70	49	71	41	75	31	76	19												
187		52		230		48														12,72644014	51,16666667
12,8342246				13,04347826																	
73	16	76	4	76	44																
168		40																		14,07286577	41,27272727
14,28571429																					
																				12,58752218	60,57142857
																				10,47367456	29,75
																				14,461521	45,5
189		48,2		242		50,3		249		50,3		240		44,5		236				12,48916704	48,72562513
12,74517221				12,43641871				12,30966099				12,16630465				12,13203463					

Tabla 6-7- Resultados obtenidos a partir del análisis de los datos aportados por el simulador. Parte C.

Los resultados mostrados en la Tabla 6-4 a Tabla 6-7 fueron obtenidos con la siguiente configuración de las formaciones. Los trenes se irán enumerando a partir de la primera fila hacia abajo. El tiempo utilizado para la configuración del Latch fue de 100 milisegundos, este tiempo se respetó en todos los agentes que simularon los trenes, y de un tiempo de inicio entre formación y formación de 10 minutos. Recordar que el tiempo que se le configura a una formación para detenerse, se computa con un (+/-) 15% del tiempo seteado. Esta elección de funcionamiento se implementó así, para que la simulación no sea rígida o demasiado estática. La longitud de las formaciones fue de 100 metros para todas. Las nubes fueron, de 50 metros para la nube A trasera y 75 metros para la nube B trasera; en cuanto a la nube A frontal fue de 60 metros y la nube B frontal se configuró en modo dinámico. Por lo tanto, la longitud de la nube A frontal oscila entre 60 metros y la componente necesaria para el frenado de la formación a 0 km/h en metros.

Configuración de las formaciones para la simulación.		
Ubicación en la tabla	Velocidad máxima:	Tiempo de detención:
Tren de la primera fila:	**43 km/h.**	**45 segundos.**
Tren de la segunda fila:	**50 km/h.**	**56 segundos.**
Tren de la tercera fila:	**47 km/h.**	**62 segundos.**
Tren de la cuarta fila:	**50 km/h.**	**51 segundos.**
Tren de la quinta fila:	**57 km/h.**	**47 segundos.**
Tren de la sexta fila:	**50 km/h.**	**60 segundos.**
Tren de la séptima fila:	**40 km/h.**	**35 segundos.**
Tren de la octava fila:	**55 km/h.**	**45 segundos.**

Tabla 6-8- Configuración de cada uno de los trenes de la simulación.

La tabla principal (Tabla 6-2 a Tabla 6-4) o la que se creó a partir de los datos obtenidos por el portal de SOFSE, se explica como se muestra a continuación. Siendo que la tabla generada por la simulación es construida de la misma forma para una mayor compresión de los datos.

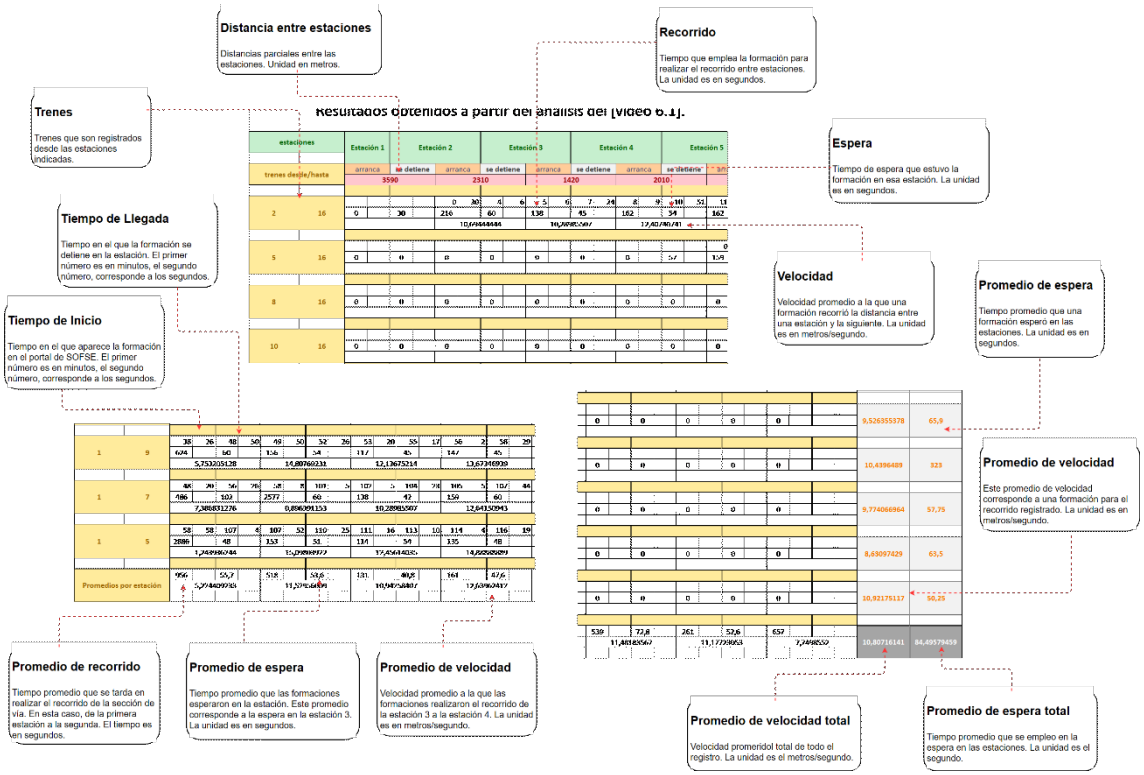


Ilustración 6-4- Gráfica útil para comprender las tablas de los análisis.

Es posible que pueda obtenerse una simulación más próxima al servicio real por medio de este prototipo, pero no ha sido la intención de esta evaluación. El propósito de la comparación fue resaltar los datos mencionados, como la variabilidad que tiene el servicio ante los usuarios y que éstos pueden provocar una demora mayor aún. A su vez, las formaciones circulan a velocidades dispares, por momentos a una velocidad reducida y por momentos a una velocidad más elevada para el mismo sector. Esta inconsistencia, junto con los sistemas que utilizan de posicionamiento de las formaciones, pueden finalizar dando origen a situaciones imprevistas.

Por todo lo expuesto, la información obtenida a través del simulador en manos de personal cualificado para su evaluación es altamente probable que arroje procedimientos o métodos de operación más eficientes en el uso de las formaciones y circuitos de vías.

No está de más recordar que el propósito del simulador es el uso eficiente y seguro de un entorno ferroviario mediante el uso de agentes que se comunican para gestionar el control del circuito, evitando que las formaciones sufran accidentes y que las estaciones de ascenso y descenso de pasajeros sean puntos de detención de las formaciones obligatorios.

6.2 CONTROL EFICAZ DE LAS FORMACIONES

Este proyecto cuenta con un algoritmo de proximidad específico, el cual tiene por interés poder colocar la mayor cantidad de trenes posibles de forma segura. Pero al ser justamente un algoritmo, tiene la posibilidad de cambiarse, adaptarse y combinarse con distintas técnicas para obtener un servicio más eficiente con la necesidad requerida. Entiéndase que eficiente, puede no ser que el simulador funcione a su máxima capacidad de carga. Sino, que por las características que tiene el circuito diseñado, se necesita que una formación circule cada determinado tiempo, dejando libre la vía a otro intervalo de tiempo previamente definido.

Los tiempos/retardos de comunicación son un objetivo importante para un sistema de tiempo real donde la cantidad de información enviada tiene límites. La capacidad que tiene el simulador de poder realizar pruebas de comunicación a distintas frecuencias de intervalos dentro de una red limitada en sus comunicaciones, ofrecerá resultados que cooperan en la toma de decisión de cuanta información comunicar. A su vez, la velocidad de las formaciones no debe pasarse por alto, los valores que resulten de este tipo de pruebas de uso intensivo de la arquitectura de red, no son despreciables en ningún caso.

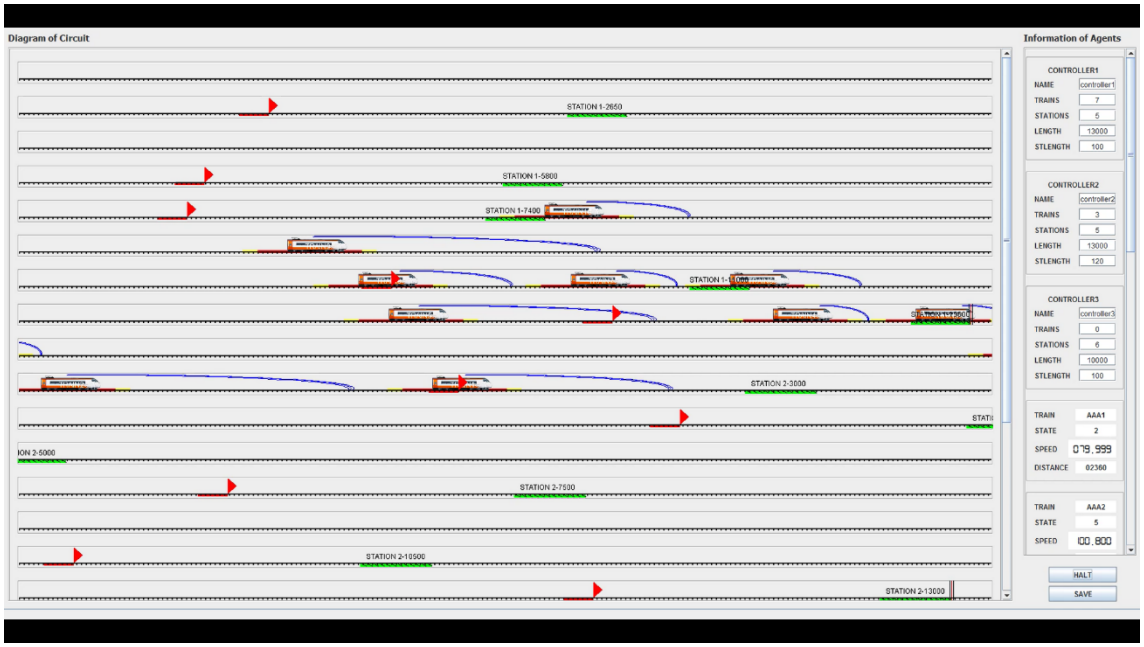


Ilustración 6-5- Situaciones que el simulador controla, obtenida de [Video6-4].

Situaciones como la expresada en la Ilustración 6.5 son eventos para los cuales el simulador fue constituido. Como se mencionó anteriormente, el simulador cuenta con un algoritmo de proximidad que evita que las formaciones se acerquen más de lo permitido. En este trabajo, el algoritmo utilizado presenta la característica de que las formaciones se mantengan alejadas entre sí, a una distancia tal, que la formación que se está acercando por detrás cuente con la capacidad de detenerse sin ingresar a la nube B del tren que se encuentra por delante, independientemente de si este se está moviendo o no.

En la siguiente imagen, Ilustración 6-6, se observa que las formaciones están trabajando en un espacio bastante más reducido que el figurado la Ilustración 6-5. Para esta simulación se dispuso que la formación que va por delante, circule a una velocidad no mayor a 43 km/h. Esta situación, provoca que el resto de las formaciones que tienen una velocidad superior deban adaptarse a la velocidad de su predecesor en el circuito de vía, priorizando la seguridad del sistema.

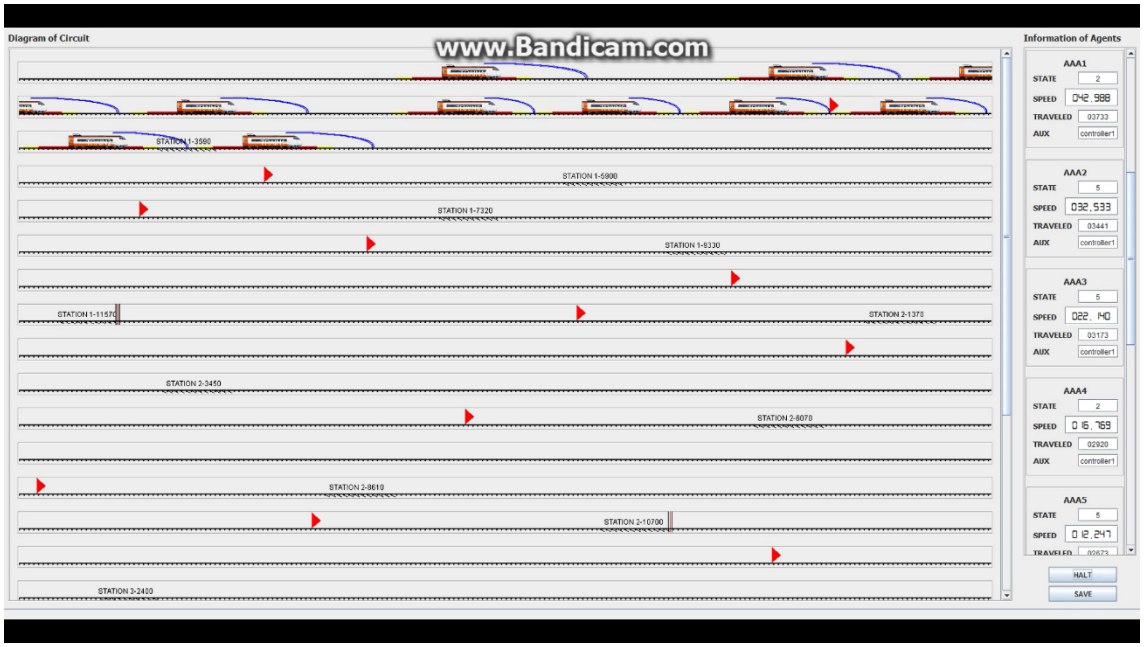


Ilustración 6-6- Trenes con alto grado de cercanía, obtenido de [Video6-3].

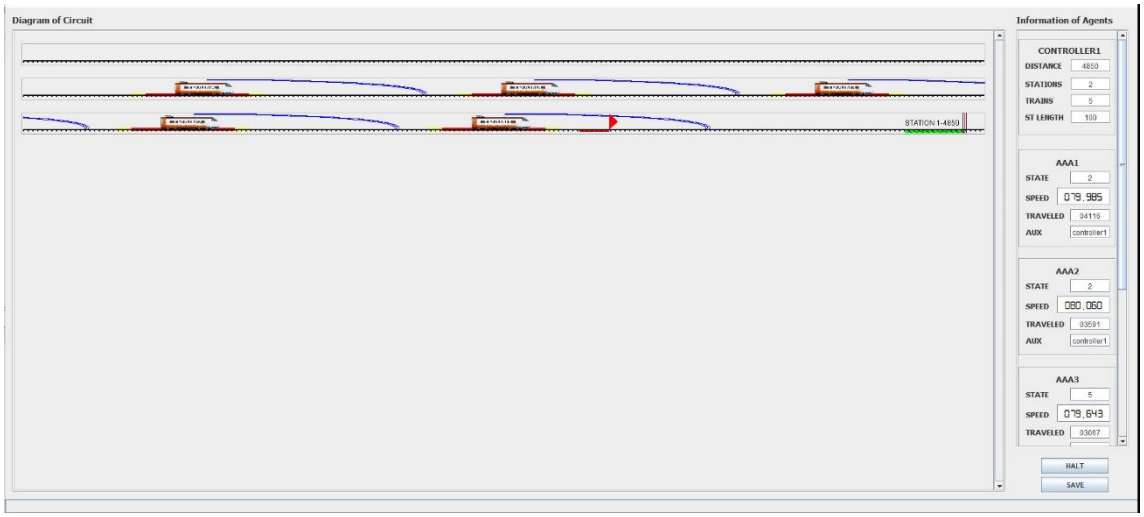


Ilustración 6-7- Trenes circulando a distancia reducida.

En la Ilustración 6-7, es parte visual del archivo XML [archivo 6.2] que se deja como resultado de la simulación. Para que este archivo no sea demasiado extenso, la simulación se realizó sobre solo un agente de control, el cual controla un trayecto de 4.850 metros con cinco (5) estaciones y se simularon cinco (5) formaciones circulando por el circuito de vía. En esta simulación, puede notar visualmente y por supuesto, en la información almacenada en el archivo XML, que las formaciones nunca se cruzan en las nubes de seguridad.

La Ilustración 6-8 es un fragmento del archivo XML generado. En ella puede observarse que, entre una formación cualquiera y su predecesora, se dispone de una distancia de 235 metros. Esta distancia es medida desde la parte

frontal del tren que está por delante a la parte frontal del tren que se encuentra inmediatamente por detrás. Para identificar cual es la distancia real efectiva que separa a las formaciones, se debe quitar la longitud de la formación que se encuentra delante, para obtener la posición de su parte trasera.

Esta distancia entonces se corresponde con los siguientes valores, la nube frontal B de la formación que se encuentra inmediatamente por detrás, para esta simulación, se configuro en 60 metros. La nube B trasera, de la formación desde la que se elija para realizar el análisis, se configuro el valor de 75 metros; y para finalizar, la longitud de esta última formación, se seteo en 100 metros. Resumiendo, se cuenta con los siguientes valores:

$$\boldsymbol{cloudBFront(tren\ que\ alcanza) = 60\ metros.}$$

$$\boldsymbol{cloudBRear(tren\ desde\ el\ que\ se\ realiza\ el\ análisis) = 75\ metros.}$$

$$\boldsymbol{length(longitud\ del\ tren\ analizado) = 100\ metros.}$$

$$\boldsymbol{Distancia\ de\ seguridad\ total = cloudBFront + cloudBRear = 135\ metros.}$$

$$\begin{aligned}\boldsymbol{Distancia\ de\ seguridad\ registrada\ en\ XML}\\ &= \boldsymbol{cloudBFront + cloudBRear + length}\\ &= \boldsymbol{60\ metros + 75\ metros + 10\ metros = 235\ metros.}\end{aligned}$$

Ecuación 6-1 - Distancia de seguridad efectiva entre las formaciones detenidas.

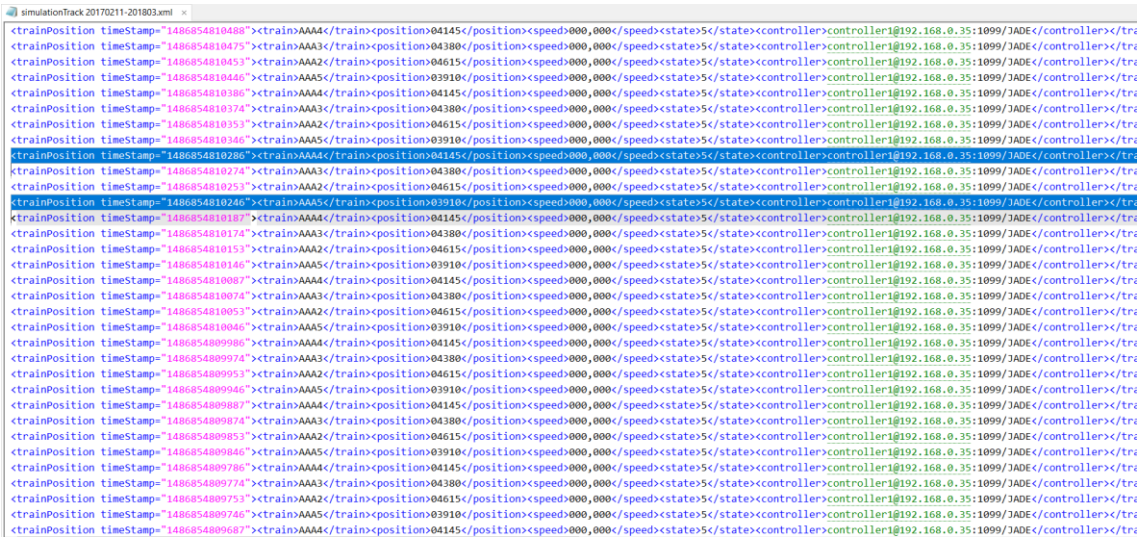


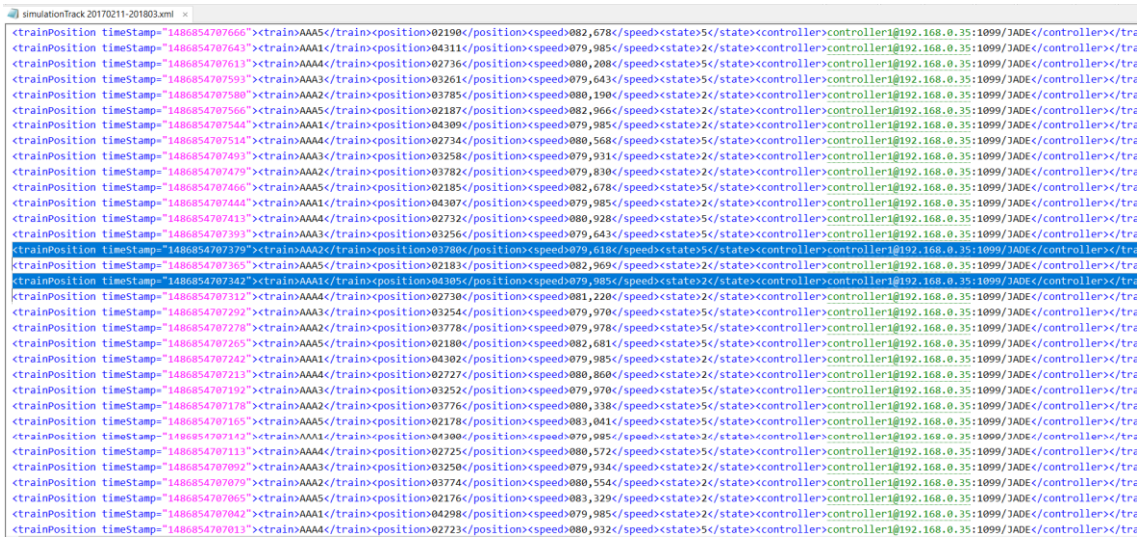
Ilustración 6-8- Traza almacenada en el XML.

La distancia de seguridad efectiva explicada en la Ecuación 6-1, corresponde a la distancia mínima a la que pueden encontrarse las formaciones. En la siguiente Ilustración podemos observar que, tal distancia aumenta conforme aumenta la velocidad a la que se desplazan las formaciones objeto de análisis.

Se tiene para las dos posiciones seleccionadas en la Ilustracion 6-2, una distancia de seguridad efectiva de 525 metros sin descontar la longitud de la formación AAA1, la cual se encuentra por delante. Como se ha explicado para esta simulación, se dispuso una longitud de las formaciones de 100 metros, por lo que la distancia de seguridad efectiva se ve reducida en 100 metros, quedando en 425 metros. Nuevamente, tenemos que la longitud de cloudBFront para la formación AAA2 es de 60 metros. Y la longitud de cloudBRear de la formación AAA1 es de 75 metros.

$$
\begin{aligned}
&\boldsymbol{Distancia\ de\ seguridad\ efectiva} \\
&\quad = \boldsymbol{distancia\ registrada - cloudBRear(AAA1)} \\
&\quad - \boldsymbol{cloudBFront(AAA2) - length(AAA1)} \\
&\quad = \boldsymbol{525\ metros - 75\ metros - 60\ metros - 100\ metros} \\
&\quad = \boldsymbol{290\ metros}.
\end{aligned}
$$

Ecuación 6-2- Cálculo de la distancia efectiva de las formaciones AAA1 y AAA2 rodando.



Ecuación 6-3- Distancia de seguridad efectiva entre las formaciones.

El resultado obtenido en la Ecuación 6-2 de 290 metros ¿Es suficiente para mantener alejadas las formaciones con seguridad? En las Ecuaciones 5-4 y 5-8, se mostró como calcular el tiempo para detenerse, ***deltaTiempo*** y ***espacioParaDetenerse*** respectivamente, aplicando sendas ecuaciones, concluimos los siguientes resultados.

Llevado a cabo el correspondiente cambio de unidades para realizar los cálculos en las mismas unidades. Para ello, se realiza el pasaje de unidades del registro de la velocidad de kilómetros/hora a metros/segundo.

$$\frac{\boldsymbol{km}}{\boldsymbol{h}}\boldsymbol{a}\frac{\boldsymbol{m}}{\boldsymbol{s}} = \frac{\mathbf{79,618}\ \boldsymbol{km}}{\boldsymbol{h}} \times \frac{\mathbf{1000}\ \boldsymbol{m}}{\mathbf{1}\ \boldsymbol{km}} \times \frac{\mathbf{1}\ \boldsymbol{h}}{\mathbf{3600}\ \boldsymbol{s}} = \frac{\mathbf{22,116}\ \boldsymbol{m}}{\boldsymbol{s}}$$

$$\boldsymbol{deltaTiempo} = \left(\mathbf{0}\frac{\boldsymbol{m}}{\boldsymbol{s}} - \mathbf{22,116}\frac{\boldsymbol{m}}{\boldsymbol{s}}\right) \times -\frac{\mathbf{1}\ \boldsymbol{m}}{\boldsymbol{s^2}} = \mathbf{22,116}\ \boldsymbol{s}$$

$$\boldsymbol{espacioParaDetenerse} = \boldsymbol{deltaTiempo^2} \times \frac{\mathbf{1}}{\mathbf{2}} = \mathbf{22,116^2} \times \frac{\mathbf{1}}{\mathbf{2}} = \mathbf{244,56}\ \boldsymbol{m}$$

Ecuación 6-4- Calculo de la distancia de seguridad necesaria para detener la formación en alcance.

Entonces, es válida la distancia a la que se encuentra la formación AAA2 de la AAA1. Aun así, la distancia con la que se cuenta de seguridad es un 18% mayor de la necesaria, sin contemplar que ambas formaciones se están moviendo. Posibilitando ello, a que, cualquier imprevisto que ocurra en la formación de adelante, puede ser resuelto dentro de este margen de seguridad. Si se examina detenidamente, el espacio de proximidad podría disminuir todavía más, incluso reducirse en más del 18% inicial obtenido, puesto que la formación que circula por delante, debe emplear aproximadamente la misma distancia para realizar una detención completa. Empero, éste análisis podría convenir hacia futuros avances, como asi también, en mejoras del presente trabajo.

7 CONCLUSIÓN, TRABAJOS FUTUROS Y PROBLEMAS CURSADOS

Para el desarrollo de la tesina de grado, se construyó un prototipo de un simulador de control de formaciones ferroviarias inicial mediante agentes JADE, el cual está pensado para que pueda ampliarse. Su característica principal fue la de ofrecer control distribuido de las formaciones que circulen en el trayecto simulado con máxima seguridad haciendo un uso eficiente de los recursos ferroviarios, en otras palabras, de los circuitos de vías, de las formaciones y de las estaciones. Al no depender, de sistemas convencionales de señalización ferroviaria (cuyos costos elevados impiden la circulación de una cantidad de formaciones próximas), el simulador permite mantener distancias reducidas entre formaciones, suscitando un desafío para la generación de nuevos sistemas de control y señalización. En ambos casos, se impide que se realicen paradas de todo un servicio por la detención de una formación en un sector del circuito de vía o porque la formación se detuvo en una estación, ocasionando un bloqueo de la vía.

El simulador permite diseñar una gran variedad de circuitos de vías con uno o más controladores que se encarguen de gestionarlo. Todos ellos con un único sentido de circulación. Pudiendo contar con un amplio número de estaciones disponibles, dispuestas a las distancias requeridas. En el diseño de dicho simulador se empleó el sistema métrico decimal, siendo que de esta manera, se facilita el desarrollo de modelos de circuitos de vías y sus distintos modos operativos, que se requieran.

La cuanto a la funcionalidad del simulador, no fue pensada para que interactúe con el usuario final sino, que su finalidad es la de proveer una herramienta de simulación potente y que represente la cinemática del entorno ferroviario, para que por medio del análisis de la información producida se pueda conseguir soluciones eficaces y/o mecanismos de operación que solo pueden obtenerse en base situaciones generadas reiteradas veces, funcionalidad que sí es propia del prototipo de simulador.

El entorno utilizado de agentes de software provisto por JADE es una herramienta potencial y de gran desempeño para ser empleada en la simulación. Permitió que el desarrollo se centre en las funciones que cada agente debe realizar, haciendo una abstracción de los problemas clásicos en este tipo de desarrollos como son, la sincronización de los mensajes, la movilidad de los agentes de un nodo a otro dentro de la arquitectura, fallas inherentes a las comunicaciones, etc. A su vez, al estar desarrollado completamente en JAVA

(lenguaje ampliamente utilizado y con una comunidad de desarrolladores que aportan gran cantidad de soluciones), fue brindando al desarrollo e implementación del prototipo de versatilidad y variedad de implementaciones a los distintos problemas que debieron resolverse.

7.1 CONCLUSIÓN

Concluir que este trabajo es un recorrido por un gran número de asignaturas de las Ciencias de la Información, no es una conclusión que deba pasarse por alto. Se comenzó con la problemática de la distribución de procesos aislados que interactuaban desacoplados y que simulaban los distintos participantes del simulador, como son los trenes y los controladores. Comunicar la información a través de canales de red, generó el diseño de protocolos de comportamiento que cada agente desempeña para cumplir su cometido. Luego, se tuvo la necesidad de someterlos a una dinámica de actuación en la que se fundamentan, proveyéndolos de los valores específicos de los objetos que modelaban, permitiendo que los estímulos y respuestas coincidan con los tiempos de los sistemas reales que emulan.

Entonces, la cinemática, como división de la física que estudia la dinámica del movimiento de los objetos sólidos sin considerar las causas que lo provocan, necesitó adicionarse en el proyecto. Términos como movimiento rectilíneo uniforme, movimiento uniformemente acelerado, movimiento constante, etc., convinieron asimilarse y entenderse para ser incluidos, situando al simulador en el contexto del problema. Para ello fue necesario proponerse distintas técnicas y así conseguir los variados efectos de control deseados sobre las formaciones, con la premisa de brindar seguridad en todo el evento de la simulación.

Para obtener valores aceptables de control e interacción, el desarrollo concurrente y paralelo de las distintas actividades surgió como un único medio de solución posible. Aun así, se percibían limitación de acción a los estímulos generados por los controladores hacia las formaciones. Por consiguiente, se necesitó probar distintas estructuras de datos que faciliten un fluido acceso directo a los volúmenes de datos que se producen. Las limitaciones de la representación gráfica para responder a tiempo, por sobretodo, fueron uno de los principales motivadores de la revisión de las estructuras seleccionadas al inicio del proyecto.

De manera manifiesta concluyó que un simulador con estas características, es un sistema apropiado y transformador para el sector ferroviario. Sin embargo, es un trabajo de nivel inicial y no cuenta con un número

considerable de funcionalidades para los usuarios, no obstante, genera un precedente al que se le pueden incluir, en sucesivos trabajos, distintas capas de funcionalidad. Innovador, porque basado desde el enfoque del software, el prototipo se posiciona frente a equipos de hardware especifico que son diseñados y construidos para proporcionar el control, pudiéndose actualizar, únicamente con un equipo nuevo. En el prototipo el solo rediseño del algoritmo, ofrece una ventaja innovadora del control como así también, la de incluir nuevas funcionalidades en función de la variación de los requerimientos.

El volumen de datos generados, merece un análisis por parte de operadores de sistemas férreos. Estos pueden utilizar la información generada para la toma de decisiones y proponer alternativas de diseño de circuitos de vías a los sistemas convencionales. La información producida es cuantiosa, ya que refleja la posición de las formaciones en promedio 10 veces por segundos, dependiendo del lapso en que se configure la comunicación. Y si se tienen 10 o más formaciones circulantes, dicho volumen puede crecer rápidamente.

El punto más significativo, es la capacidad que expone el simulador de adaptarse a distintos entornos. Al estar desarrollado en su totalidad en Java, puede ejecutarse en múltiples plataformas. Su diseño creado mediante el uso de agentes de software acarrea una cualidad muy importante, la movilidad. La ejecución del simulador no supone que tiene que ejecutarse en una única computadora. Por ese motivo, puede generarse una simulación desde distintas computadoras que comparten una red de datos. Esto supone que una simulación puede integrarse en un modelo real, como podría darse la situación de que se monte el agente en una formación en movimiento para el posterior análisis de los datos producidos.

Esta versión del simulador, centra su funcionamiento en controlar la detención de las formaciones en los sectores definidos para ello. Pero también, en que las formaciones se encuentren durante la simulación a una distancia de seguridad en la que se controle la prevención de accidentes o situaciones de aproximación indeseadas. Esto se cumple para todas las simulaciones que puedan diseñarse mediante el simulador dentro de los límites físicos permitidos. Puesto que, la simulación de una formación con cantón fijo de una longitud inferior a la que puede detenerse en la velocidad máxima permitida, entregara un saldo desfavorable. La simulación de una formación de cantón fijo, no es la mejor opción para llevar a cabo una utilización eficiente, tanto del circuito de vía como de la formación en cuestión.

Y para finalizar, un último dato que propicia el simulador, es relevar las cotas máximas de velocidad de funcionamiento permitido de las formaciones. Es decir que, si una formación supera las cotas entregadas por el simulador para un

sector en que un tren debe detenerse o disminuir su marcha, puede concluir en accidente o en un evento fortuito.

7.2 POTENCIALES TRABAJOS FUTUROS

En esta sección se detalla una breve descripción de posibles trabajos que pueden adicionarse/derivarse del presente prototipo. Trabajos inherentes al desarrollo de una interfaz que permita la conexión de agentes simulando ser barreras, semáforos, señalización de velocidades máximas o mínimas, etc., sobre la plataforma actual. Esto permitirá que se simulen todos estos componentes y aportaría al prototipo un mayor nivel de realismo. Una interface que permita lo mencionado anteriormente, abriría distintos campos de investigación. Podría desarrollarse un agente que no solo simule a una barrera, sino que el mismo agente también puede operarla. Llevando al simulador a una escala más, dentro de las simulaciones, permitiendo una cercana operatividad con elementos del mundo.

7.2.1 Dotar de capacidad de resolución a las formaciones

En esta tesina, se trabajó con la directiva sobre que las formaciones no contasen con “inteligencia” o capacidad de discernir entre las distintas situaciones posibles, delegando en los controladores el control del entorno. Una variación de esta solución con mayor potencia de cómputo, puede hallarse, dotando a las formaciones con la capacidad de actuar o responder a eventos que ellas mismas consideren que deban anticiparse a la orden proveniente del controlador bajo el que se encuentran.

Este agregado, supondría de mayor funcionalidad al simulador, permitiendo que los trenes, cuenten con mayor número de tareas para ejecutar. Ya no solo tareas abocadas al control de las formaciones, sino que, por medio de un relevamiento de la actividad ferroviaria, adicionar situaciones o problemas específicos del entorno, como puede ser el tiempo de apertura de puertas, detectar problemas con la tensión de la catenaria, medir el abatimiento que sufre la formación al pasar por cierto sector que se encuentra inclinado, etc.

7.2.2 No solo controlar, sino generar errores

Como se ha visto a lo largo del trabajo, el eje de la simulación solo se centró en el control de las formaciones que lo circulan. Bien, pero para brindar un nivel de detalle más realista a la simulación, no debería descartarse la producción de errores o eventos fortuitos que causen un cambio de rumbo del entorno que se está simulando. El control está demostrado que fue obtenido. El poder interferir la simulación y generar algunos eventos de errores de comunicación de las formaciones, o que los controladores queden fuera de servicio, etc., sumaría al prototipo un amplio número de estímulos a los que debería, en primer lugar, detectar y para proseguir, responder eficazmente al evento ocurrido.

7.2.3 Uso de plataformas de hardware libre

Existen distintas plataformas de hardware en auge de costo verdaderamente económicas. Una de estas plataformas es Raspberry Pi. Para este equipo existen un amplio número de shields que puede acoplarse y que, por medio de los agentes, simular como se mencionó anteriormente, un control de una barrera ferroviaria, la activación de una señal, etc. De contarse con más dispositivos, como son los acelerómetros y los sensores de movimiento, los agentes que simulan los trenes podrían adaptarse para alimentarse con valores más próximos a la realidad. Esto supondría un gran cambio en el prototipo, para entonces poder acercarse a un sistema real, no muy distante.

7.2.4 Capa de software de seguridad

Otro campo de investigación, podría ser el área de la seguridad para la integración con agentes desarrollados por distintos autores. En el trabajo presentado, no se cuestiona o esquematiza sobre la seguridad informática. Pero no deja de estar presente en todo momento. Dicha capa, obligaría a todos los agentes que integren la simulación a tener que contar con credenciales para su integración y autenticación con la plataforma. Para evitar así, el agregado de agentes de software no reconocido y/o intrusiones no deseadas, para el caso de que la simulación se esté ejecutando en un ambiente geográficamente distribuido.

Todo agente tren o controlador que se integre al simulador, previamente deberá informar sus credenciales. Pero si, además se cuenta con una interface para la interconexión genérica de que cualquier tipo de agente ferroviario que pueda acoplarse a la plataforma, la seguridad se torna un ítem crucial.

7.2.5 Interfaz visual

Al igual que un plano geográfico el cual contiene una representación específica para cada uno de los componentes que lo integran, en un desarrollo técnico, se tiene una representación gráfica para cada uno de sus componentes. Partiendo de esta base, construir una interfaz gráfica que se adecue a las referencias técnicas de los sistemas ferroviarios, proporciona una mejor legibilidad para los técnicos e ingenieros del sector.

A su vez, la interfaz gráfica podría construirse en un lenguaje de programación dedicado al desarrollo de interfaces gráficas. Mediante el uso de un formato de texto ligero para el intercambio de datos, como lo es JSON, puede obtenerse una variedad de interfaces gráficas distintas y con mayor riqueza gráfica.

Además, se notó con el uso del prototipo, que una interfaz partida o desglosada puede ofrecer una distribución de la información visual con mayor feedback. Esta interfaz, puede contener la representación de los controladores individualizados y en una ventana propia, con la capacidad de colocarse en el sector de la pantalla que el usuario prefiera. Misma situación, para el conjunto de tramos que representan las vías y también para los dashboards de las formaciones. En caso de que se cuente con más de una pantalla, la información a visualizar puede distribuirse, la visualización de los circuitos de vías en un monitor, y el conjunto de paneles de control de trenes y controladores, en el monitor restante.

7.3 PROBLEMAS CURSADOS

Un problema con el que se necesitó realizar distintos planteamientos de resolución, fue la visualización de la gráfica de las formaciones en los circuitos de vías. La característica de tener que mostrarse los datos al instante de ser recibidos, para el entorno de un equipo convencional de hardware, significo un problema mayúsculo. Por un lado, se tiene a los controladores y a las formaciones ejecutando constantemente las fases de simulación que

representan. Y por el otro, se encuentra el agente de la interfaz visual, otorgándole sentido visual a la información recibida. En el medio, el hardware de un equipo ejecutando la máquina virtual de Java que requería a cada instante más recursos, no solo memoria, sino, más bien recursos de tipo de multiprogramación, para dar respuesta a la representación gráfica.

Otro inconveniente, que ocurrió en las primeras etapas del proyecto fue comprender el hecho de anticipar a las formaciones para proveerles el correspondiente control. La solución se presentó de la mano de la cinemática. Más allá del conocimiento que se pueda tener movimiento de objetos sólidos, la solución radicó en como detenerlos a tiempo y con seguridad, evitando que se sobrepasen las estaciones o atravesasen las formaciones que se encontraran por delante.

8 BIBLIOGRAFÍA

[Book2] IEEE Standard for Communications-Based Train Control (CBTC) Performance and Functional Requirements - The Institute of Electrical and Electronics Engineers, Inc. – New York, USA – 1999 - PDF: ISBN 0-7381-1826-5 SS94795.

[Book4] Control Through Communication: The Rise of System in American Management - JoAnne Yates., Johns Hopkins Press - 1993. ISBN 9780801846137.

[Book5] Artificial Intelligence: A Modern Approach - Stuart Rusell & Peter Norvig - Prentice Hall, 1st Edition, - 1995 - ISBN 978-0131038059 - pp 30-45.

[Book6] Intelligent Agents: Theory and Practice – Michael Wooldridge & Nicholas R. Jennings – London, UK - Queen Mary & Westfield College University of London, capítulo 1.1 – 1995.

[Book7] Agentes Software y Sistemas Multi-Agente: Conceptos, arquitecturas y aplicaciones - Ana Mas – Pearson Educación – 2004 - ISBN 84-205-4367-5 - pp 10-160.

[Book8] Concurrent and Real-Time Programming in ADA - Alan Burns & Andy Wellings – Cambridge University Press, Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo - 2007 - ISBN-13 978-0-511-29663-5.

[Book9] Introduction to Parallel Computing, Second Edition - Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar - Addison Wesley - January 16, 2003 - ISBN: 0-201-64865-2.

[Book16] Developing Multi-agent Systems with JADE - Fabio Bellifemine, Agostino Poggi, Giovanni Rimassa – 2001 - Wiley Editorial, ISBN 978-0-470-05747-6 - pp 30-32.

[Book20] Intelligent Agents V: Agents Theories, Architectures, and Languages 5th International Workshop, ATAL'98 Paris, France, July 4–7, 1998 Proceedings - Jörg P. Müller, Anand S. Rao, Munindar P. Singh - ISBN: 978-3-540-65713-2 (Print) 978-3-540-49057-9 (Online) - 1999 - 464 páginas.

[Sim1] Agent-based Planning and Simulation of Combined Rail/Road Transport - Luca Maria Gambardella, Andrea E. Rizzoli IDSIA, Galleria 2, CH-6928 Manno, Switzerland Petra Funk DFKI, Stuhlsatzenhausweg 3, D-66123, Saabrücken, Germany.

[Sim2] Simulation and evaluation of urban rail transit network based on multi-agent approach - Xiang-Ming Yao, Peng Zhao, Ke Qiao School of Traffic and Transportation, Beijing Jiaotong University (China).

[Sim3] Multi agent based train simulation, #3407918 - Jiske van der Lof, Universiteit Utrecht, Artificial Intelligence Information and computing science Faculty of Science October 2014 – August 2015.

[Book27] Design Patterns Elements of Reusable Object-Oriented Software - *Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides* - Pearson Education - 1994 - ISBN 9780321700698.

[Book29] Ingeniería mecánica: Dinámica - William F. Riley &, Leroy D. Sturges - España, Editorial Reverté, S.A. 1996 - ISBN: 84-291-4256-8.

[Book30] Física, Elementos fundamentales - Mecánica y termodinámica clásicas - Relatividad especial - Tomo I - Mario Guerra, Juan Correa, Ismael, Núñez, Juan Miguel Scaron - España, Editorial Reverté, S.A. 1994 - ISBN: 84-291-4308-4.

[Book31] Developing Multi-Agent Systems with JADE Volume 7 of Wiley Series in Agent Technology - Fabio Luigi Bellifemine, Giovanni Caire, Dominic Greenwood - John Wiley & Sons, 2007 - ISBN 0470058404, 9780470058404.

9 ENLACES:

[Link1] Google Self-Driving Car Project: https://www.google.com/selfdrivingcar/ - (último acceso 16/08/2016).

[Link3] INNOVIA APM 100 – San Francisco International Airport, USA: http://www.bombardier.com/en/transportation/projects/project.innovia-san-francisco-usa.html?f-region=americas/en/transportation/products-services/transportation-systems/driverless-systems/automated-monorails/s& - (ultimo acceso 16/08/2016).

[Link9] FIPA ACL Message structure presentation - IEEE Fundation for intelligent physical agents - 2002 - http://www.fipa.org/ - (último acceso 09/12/2016).

[Link10] Mobile-C: A Multi-Agent Platform for Mobile C/C++ Agents - http://www.mobilec.org/ - (último acceso 09/12/2016).

[Link11] Open source project KATO for PHP and Java developers to write software agents - http://kato.sourceforge.net/kato.html - (último acceso 09/12/2016).

[Link12] JADE, Java Agent Developing Framework, an Open Source framework developed by Telecom Italia Labs - http://jade.tilab.com/ - (último acceso 09/12/2016).

[Link13] Simulación basada en agentes de software para la evaluación de indicadores técnicos - Tesis de Alejandro Escobar, Julián Moreno, Sebastián Múnera, Universidad Nacional de Colombia Sede Medellín - http://www.redalyc.org/articulo.oa?id=43021467013 - (último acceso 09/12/2016).

[Link14] Agentes de Software como Herramienta para medir la Calidad de Servicio Prestado en un Sistema de Transporte Público Colectivo Urbano - Tesis de Mauro Callejas-Cuervo, Helver. A. Valero-Bustos y Andrea. C. Alarcón-Aldana - Universidad Pedagógica y Tecnológica de Colombia, Sede Central Tunja-Boyacá-Colombia - http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-07642014000500020 - (último acceso 09/12/2016).

[Link15] Simulación basada en Agentes Reactivos - Tesis de Andrés Díaz Pace - Federico Trilnik - Marcelo Campo - Alejandro Clausse Universidad Nacional del Centro de la Provincia de Buenos Aires Facultad de Ciencias Exactas - http://sedici.unlp.edu.ar/bitstream/handle/10915/22269/008Simulaci%F3n+basada+en+Agentes+

Reactivos.pdf;jsessionid=A56976BE56894162CDE721954B85159F?sequence=1 - (último acceso 09/12/2016).

[Link24] NETWORKED SIMULATION WITH HLA AND MODSIM III - Glen D. Johnson https://pdfs.semanticscholar.org/1bf4/43efd614dc216382636a7b983b4844d42bab.pdf - (último acceso 16/12/2016).

[Link25] Especificaciones Técnicas Básicas - Equipamiento Electromecánico Y Material Rodante - Tramo: Villa El Salvador – Av. Grau Tomo 2 Setiembre 2007 - http://www.proyectosapp.pe/RepositorioAPS/0/0/JER/TREN_ELECTRICO_DOCS_CONTRATO/Tomo%202_Material%20Rodante%20(28.09.07).pdf - (último acceso 23/12/2016).

[Link26] Sistemas de señalización y control ferroviario en alta velocidad - Rodríguez Cea, Ángel Iván - Julio de 2015 - https://uvadoc.uva.es/bitstream/10324/13367/1/TFG-P-267.pdf - (último acceso 23/12/2016).

[Link28] Detectores de vía para aplicaciones ferroviarias - http://www.sensorstecnics.net/en/productos/category/151/sensores-y-transmisores/detectores-de-via-para-aplicaciones-ferroviarias - (último acceso 04/01/2017).

[Link32] Multi-Agent-Based Simulation (MABS) - http://www.pcs.usp.br/~mabs/ - (último acceso 06/02/2017).

[Link33] CNRT – Comisión Nacional de Regulación del Transporte - https://cnrt.gob.ar/ - (último acceso 06/02/2017).

[Link34] SOFSE – Operadora Ferroviaria Sociedad del Estado - https://www.sofse.gob.ar/portal/index.php - (último acceso 06/02/2017).

[Video6-1] Video a partir del cual se realizaron las mediciones de operatoria del ferrocarril Sarmiento (Once-Moreno) - https://drive.google.com/open?id=0B7z0snirUSWYX1ZKLUZzUUlsNzA - (último acceso 06/02/2017).

[Video6-2] Video realizado a partir de la simulación configurada para evaluar los datos obtenidos a través del prototipo en comparación con [Video6-1] - https://drive.google.com/open?id=0B7z0snirUSWYQU9TMFVQZ3ZXUXc - (último acceso 06/02/2017).

[Video6-3] Video extra que muestra una simulación donde puede apreciarse el control de las formaciones - https://drive.google.com/open?id=0B7z0snirUSWYTGtzeFZqVldBVlk - (último acceso 06/02/2017).

[Video6-4] Video que muestra al simulador controlando formaciones a velocidades elevadas - https://drive.google.com/open?id=0B7z0snirUSWYa3NOSmJmckJrZmM - (último acceso 06/02/2017).

[Video6-5a] En conjunto con el [Video6-5b] forman parte de la traza almacenada en el [Archivo6.2] - https://drive.google.com/open?id=0B7z0snirUSWYRzRqMmdoaG01azg - (último acceso 06/02/2017).

[Video6-5b] En conjunto con el [Video6-5a] forman parte de la traza almacenada en el [Archivo6.2] - https://drive.google.com/open?id=0B7z0snirUSWYcmN0XzlTankyZDg - (último acceso 06/02/2017).

[Archivo6.1] Archivo XML resultante de la traza simulada para contraponer con los datos del tren Sarmiento(Once-Moreno) obtenido en [Video6-1] - https://drive.google.com/open?id=0B7z0snirUSWYcWV5Z3oxZmdHM2c - (último acceso 06/02/2017).

[Archivo6.2] Archivo XML como resultado de la simulación mostrada en el [Video6-5a] y [Video6-5b] que demuestra la capacidad de control del prototipo - https://drive.google.com/open?id=0B7z0snirUSWYMjdmaG1BVmdqR1k - (último acceso 06/02/2017).

10GLOSARIO

Agentes de software: sistemas computacionales que habitan en algún ambiente dinámico y complejo, monitorean y actúan de forma autónoma en este ambiente, y realizando esto, logran un conjunto de objetivos o tareas para los cuales están diseñados.

ATP: Automatic Train Protection (Sistema de protección automática de trenes).

ATO: Automatic Train Operation (Sistema de operación automática de trenes).

ATS: Automatic Train Stop (Sistema automático de detención de trenes).

Atributo de un objeto: son las características individuales que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades.

ATTLIST: etiqueta que se utiliza para enumerar y declarar los atributos que pueden pertenecer a un elemento dentro de un XML.

Behaviours o comportamiento: especifican tareas o servicios que realiza un agente para lograr sus objetivos.

CallBack: retorno de la ejecución de una función.

CBTC: Communications-Based Trains Control (Control de Trenes Basado en la Comunicación).

Cinemática: rama de la física que estudia el movimiento de los cuerpos sólidos.

Circuito de vía: son los trayectos o tramos definidos para que las formación circulen.

Clase o clases: las clases son usada para construir un objeto. Una clase es como un molde para crear objetos.

Cloud: término utilizado para referirse a las distancias de seguridad utilizada por las formaciones.

CNRT: Comisión Nacional de Regulación del Transporte.

Concurrencia: es una propiedad de los sistemas en la cual los procesos de un cómputo se hacen simultáneamente.

Diagrama de secuencia: es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML.

ELEMENT: etiqueta que describe los datos que contiene. Los elementos también pueden contener otros elementos y atributos.

Estado: el estado de un objeto se refiere al conjunto de atributos y sus valores en un instante de tiempo dado.

EventListener: tipo de objeto que se utiliza para registrar un evento de un objetivo específico.

FIPA: Foundation for Intelligent Physical Agents (Fundacion para los Agentes Físicos Inteligencia). Sociedad de Computación que promueve la

tecnología basada en agentes y la interoperabilidad de sus estándares con otras tecnologías.

Getters: métodos que permiten el acceso a los atributos de un objeto.

GUI: (del inglés graphical user interface), es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

Halt: se utiliza para indicar la detención del simulador.

Handle: software o procedimiento que se encarga de realizar una determinada acción.

HLA: High Level Architecture, Arquitectura de Alto Nivel.

IEEE: Institute of Electrical and Electronics Engineers, Instituto de Ingeniería Eléctrica y Electrónica.

Instanciar: crear o poner en ejecución un objeto de una determinada clase.

Interface: dispositivo de software por el cual 2 o más objetos puede realizar un intercambio de información.

IT: Information Technology, Tecnología de la Información.

ITP: Intermodal Transport Planning, Planificador de Transporte Intermodal (ITP).

ITU: Intermodal Transport Unit, Unidad de transporte intermodal.

JADE: Java Agent DEvelopment (Entorno de Desarrollo de Agentes Java.).

JDK: Java Development Kit (Entorno de Desarrollo en JAVA).

JFieldText: objeto Java diseñado para ingresar valores en una aplicación desde la vista de un usuario final.

JPanel: objeto del lenguaje java diseñado para contener objeto swing.

Latch o **timeLatch**: enganche o tiempo de enganche para permitir ejecutar una acción.

LGPL: Lesser General Public License (Menor Licencia Pública General).

Programación Orientada a Objetos: es un paradigma de programación que viene a innovar la forma de obtener resultados. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial

Open Source Framework: Entorno de desarrollo de código libre.

KATO for PHP and Java developers to write software agents: KATO es una plataforma para desarrollo de agentes en PHP y JAVA.

Ordina: empresa multinacional fundada en 1973 con sede en Países Bajos dedicada a la innovación tecnológica.

MABS: Multi-Agent-Based Simulation (Simulación basada en agentes múltiples).

MAS: Multi-Agent System (Sistema de agentes multiples).

Método: subrutina cuyo código es definido en una clase y puede pertenecer tanto a una clase, como es el caso de los métodos de clase o estáticos, como a un objeto, como es el caso de los métodos de instancia.

Mobile-C: A Multi-Agent Platform for Mobile C/C++ Agents: Es una plataforma para el desarrollo de multi agentes móviles para los lenguajes C y C++, similar a JADE.

Modelo: representación de la realidad por medio de abstracciones. Los modelos enfocan ciertas partes importantes de un sistema (por lo menos, aquella que le interesan a un tipo de modelo específico), restándole importancia a otras.

Model-View-Controller: es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

MODSIM III: es un entorno de desarrollo dedicado a la simulación por computadora.

MRUV: movimiento rectilíneo uniformemente variado.

Raspberry Pi: computadora de placa única o de placa simple (SBC) de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Return: sentencia de retorno de la ejecución de un programa o la devolución del control de la ejecución al programa que realizo la llamada.

Setters: métodos a través de los cuales se pueden modificar los atributos de un objeto.

Shields: placa o circuito electrónico utilizado para extender la funcionalidad de un componente electrónico.

Simulador: herramienta que sirve como punto intermedio entre los conceptos teóricos y la realidad. Cuanto mayor sea la especificación del simulador, mejor serán los resultados obtenidos de la simulación entorno a la realidad.

Sistema de tiempo real: es un sistema informático que interacciona con su entorno físico y responde a los estímulos del entorno dentro de un plazo de tiempo determinado. No basta con que las acciones del sistema sean correctas, sino que, además, tienen que ejecutarse dentro de un intervalo de tiempo determinado.

Sistema distribuido: es un conjunto de computadoras autónomas interconectadas que cooperan compartiendo recursos (físicos y datos).

Sistema paralelo: es la combinación de un algoritmo paralelo y una computadora paralela.

SOFSE: Operadora Ferroviaria Sociedad del Estado.

Stream: es un medio utilizado para leer datos de una fuente y para escribir datos en un destino.

TELETRUCK: El JCB Teletruk es una máquina elevadora diseñada para cumplir una misión; reducir los gastos generales, ahorrar tiempo y espacio, y mejorar la seguridad del sitio.

Thread: es un hilo de control en el flujo de un programa.

UML: el lenguaje unificado de modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el Object Management Group (OMG).

User friendly: describe un dispositivo de hardware o una interfaz de software que es fácil de usar. Es "amigable" para el usuario, lo que significa que no es difícil de aprender o entender. Mientras que "fácil de usar" es un término subjetivo.

Variable: es un tipo de dato que puede ser modificado a lo largo de la ejecución de un programa informática.

Whiteroom: la esencia de este tipo de test, es la de develar factibles errores ocultos a través de pruebas intensivas de la aplicación.

Wrapper: es un mecanismo de “envolver” valores primitivos en un objeto para que estos primitivos puedan ser incluidos en actividades que estén reservadas para objetos, o ser retornadas desde un método con un valor de retorno que sea un objeto.

XML: es una adaptación del SGML (Standard Generalized Markup Language), un lenguaje que permite la organización y el etiquetado de documentos. Esto quiere decir que el XML no es un lenguaje en sí mismo, sino un sistema que permite definir lenguajes de acuerdo a las necesidades.