PROYECTO FIN DE CARRERA

Universidad Carlos III de Madrid

ESCUELA POLITÉCNICA SUPERIOR
INGENIERÍA EN INFORMÁTICA

Creación de un simulador aeroportuario. Sistema experto de asignación en el desembarque de pasajeros y generación de reglas a partir del conocimiento adquirido

---

Realizado por:

Javier Fernández Pérez

Dirigido por:

Julio Villena Román

Pedro Daniel Borches Juzgado

Junio, 2012

A mis padres, hermano, compañeros de la empresa Asseco y a todos los que han contribuido en el desarrollo de mi carrera universitaria y de este proyecto.

Resumen

Los aeropuertos son las infraestructuras del transporte aéreo donde las aeronaves aterrizan, despegan y se estacionan, para proceder al embarque y desembarque de los pasajeros, sus equipajes y la carga. Tanto en el movimiento de pasajeros, equipajes y carga en el aeropuerto como la estancia de una aeronave en el mismo, requieren la ayuda y colaboración de diferentes personas y equipos, entre los que destacan, de forma importante, los pertenecientes a las distintas actividades del "handling".

Cualquiera que haya tenido que pasar por un aeropuerto para coger un avión ha recibido muchos de estos servicios ya sea de forma directa o indirecta: facturación del equipaje, transporte en autobús hasta el avión, carga y descarga del equipaje, limpieza del avión, llenado de combustible de los depósitos del avión, etc. Esto hace que los servicios de "handling" (o también llamados servicios de asistencia en tierra) sean un elemento esencial y estratégico en la parte del transporte aéreo que se lleva a cabo en los aeropuertos, ya que está estrechamente relacionado con el nivel de la calidad del servicio global prestado a los usuarios. Un fallo en la prestación de estos servicios puede tener consecuencias negativas importantes tanto para la compañía aérea, como para los pasajeros, la carga aérea, el aeropuerto y el prestador de los servicios.

Para que no se produzcan estos fallos es fundamental que se asignen tanto los recursos materiales como los recursos de personal aeroportuario lo más óptimamente posible a cada uno de los servicios prestados, además de dar una solución rápida si sucede algún imprevisto y se debieran de cancelar asignaciones de recursos porque fuera prioritario que estuvieran realizando otra función. En la mayoría de los aeropuertos actuales el trabajo de asignar los recursos a los servicios aeroportuarios lo realiza o el propio prestador de servicios o un sistema software de forma semiautomática.

Este proyecto consiste en la creación de un simulador aeroportuario, y a partir del mismo, recoger información sobre las variables de decisión en las que se apoya el prestador de servicios para realizar la asignación (o cancelación de la asignación) de las escalerillas por las que bajan los pasajeros del avión cuando éste llega a su zona de estacionamiento dentro del aeropuerto. Posteriormente, se usa un modelo de minería de datos para crear el conjunto de reglas que describen el razonamiento llevado a cabo por el prestador de servicios. Aparte, en dicho simulador se puede cargar el modelo que representa el conjunto de reglas para que se realice la asignación o cancelación automática de las escalerillas a los aviones que van aterrizando.

Abstract

Airports are infraestructures of air traffic where the aircrafts land, take off and park, to proceed to the boarding and disembarkation of passengers, their luggage and freight. The movement of passengers, luggage and freight at the airport and the stay of an aircraft at it require the help and collaboration of different people and equipment, being of special importance those that belong to different activities of airport handling.

Someone who has had to go through an airport to take an aircraft has received a lot of services either of a direct or indirect way: luggage check-in, transport in bus to aircraft, loading and unloading of luggage, cleaning of aircraft, filling of fuel of tanks of aircraft, etc. This makes that handling services (also called ground handling services) are an essential and strategic element in the part of air transport carried out at the airports, because this is closely related to the level of quality of global service that it is provided to users. A mistake in the provision of these services can have important negative consequences for airlines, passengers, air freight, the airport and the people responsible of rendering these services.

To prevent that these mistakes happen it is fundamental that both material resources and human resources of the airport are allocated in the most optimal possible way to each one of rendered services, apart from giving a fast solution if some unforeseen event happens and it was necessary to cancel allocations of resources because the priority was that they perform another function. In most of current airports the task of allocating resources to the airport services is carried out by people responsible of rendering these services or a software system of a semiautomatic way.

In this project we aim to create an airport simulator, and with its help, obtain information about the decision variables used by people responsible of rendering airport services to make the allocation (or cancellation of allocation) of the steps that passengers use to go down the aircraft when it arrives at its parking space at the airport. Subsequently, we track a data mining model to create the rules set that describe the concluded reasoning of people responsible of rendering of airport services. Moreover, the model that represents the rules set can be loaded in this simulator to simulate the allocation or automatic cancellation of steps when aircrafts land.

Índice general

Índice de tablas

Índice de ilustraciones

1. Motivación y objetivos

1.1. Motivación y descripción del ámbito de estudio

En España existen 49 aeropuertos, de los cuales 24 superaron el millón de pasajeros en el año 2011 según las estadísticas suministradas por AENA [1]. Esta estadística da una magnitud de la importancia que tiene actualmente el tráfico aéreo y de la gran cantidad de vuelos que son realizados en el día a día.

La principal funcionalidad de los aeropuertos no es otra que facilitar el intercambio del modo de transporte aéreo al terrestre y viceversa, así como del aéreo al aéreo (en el caso de los tránsitos y escalas). Para poder llevar a cabo esta funcionalidad es fundamental que se realice un conjunto de actividades de soporte en el aeropuerto, es el denominado "handling" aeroportuario. Resulta difícil imaginar que todos los flujos que se producen en un aeropuerto, tanto de pasajeros como de equipajes o mercancías, se puedan llevar a cabo sin la intervención y ayuda de los prestadores de estos servicios.

Tal y como se expresa en el anexo del Real Decreto 1161/1999 [2], que es la adecuación al marco legislativo español de la Directiva Europea 96/97/CE, los servicios de "handling" (o servicios de asistencia en tierra) se clasifican en once grupos o categorías, según se muestra en la tabla siguiente.

NÚMERO	NOMBRE DEL GRUPO DE SERVICIO DE ASISTENCIA EN TIERRA
1	Asistencia administrativa en tierra y supervisión
2	Asistencia de pasajeros
3	Asistencia de equipajes
4	Asistencia de carga y correo
5	Asistencia de operaciones en pista
6	Asistencia de limpieza y servicio de la aeronave
7	Asistencia de combustible y lubricante
8	Asistencia de mantenimiento en línea
9	Asistencia de operaciones de vuelo y administración de la tripulación
10	Asistencia de transporte de superficie
11	Asistencia de mayordomía (catering)

Tabla 1: Clasificación de los servicios de asistencia en tierra

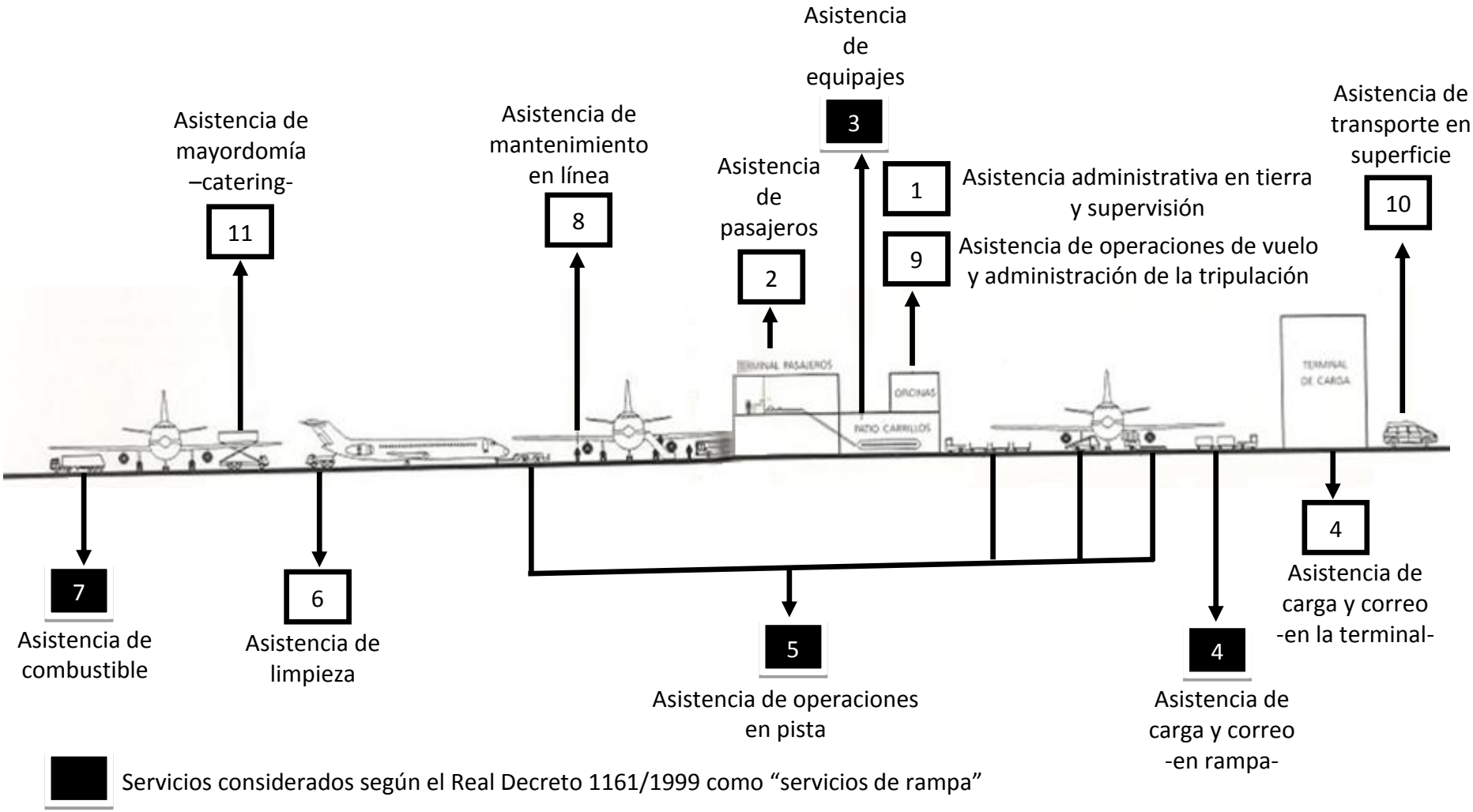


Ilustración 1: Esquema de prestación de servicios aeroportuarios de asistencia en tierra en un aeropuerto [3]

El grupo de asistencia de operaciones en pista (grupo 5), constituye uno de los grupos más importantes y es en el que se concentran gran parte de las prestaciones básicas que necesita cualquier aeronave durante su escala en un aeropuerto. Las actividades incluidas en este grupo comprenden, fundamentalmente, las comunicaciones entre la aeronave y tierra, así como todas las actividades de carga y descarga y el transporte de pasajeros, tripulación y equipajes entre el edificio terminal y el avión. Por último se incluye también el desplazamiento de la aeronave cuando ésta no puede efectuarlo por sus propios medios, así como la asistencia para el arranque del avión y el suministro de los medios adecuados.

El embarque y desembarque de pasajeros depende de la posición de estacionamiento del avión, ya sea a través de una pasarela de embarque/desembarque en contacto directo con el edificio terminal o en una posición remota sin ningún contacto con dicho edificio. Cuando el estacionamiento del avión está en una posición remota, el proceso de embarque y desembarque se realiza mediante escalerillas conectadas al avión y a través de autobuses (llamados jardineras) que llevan a los pasajeros hasta el avión o hasta el edificio terminal, siempre y cuando ese trayecto no se realice a pie. En el caso del desembarque de pasajeros, una vez el avión está estacionado y han sido colocados los calzos, se procede a continuación a situar las escalerillas en la puerta del avión por donde va a desembarcar. En el caso de los embarques que no se realizan a pie, los pasajeros siguen el proceso inverso al del desembarque, subiendo a los autobuses y siendo conducidos a pie del avión, al que acceden a través de las escalerillas.

Para tener una idea de la dimensión real de la relevancia de estos servicios dentro del transporte aéreo hay que tener en cuenta la importancia y las variadas consecuencias que tienen para las distintas partes implicadas cualquier fallo en la prestación de los mismos:

- Para la compañía aérea: la mayor rentabilidad de un avión se obtiene cuando está en el aire transportando el mayor número de pasajeros o mercancías posibles. Por este motivo cada vez son más ajustadas las programaciones de las compañías aéreas, de tal forma que la rotación diaria de un avión es tan alta que una pérdida de tiempo puede tener fatales repercusiones: compensación a los pasajeros por los daños causados, reprogramación de las rotaciones previstas con el avión retrasado y pérdida de imagen de cara al pasajero.

- Para los pasajeros: los retrasos que se produzcan en las escalas de los aviones generan demoras que en muchas ocasiones no pueden ser compensadas ya que salir y llegar en el tiempo previsto es lo que un pasajero busca siempre.

- Para la carga aérea: cualquier demora puede originar, por ejemplo entre otros, que la mercancía perecedera se deteriore, con el siguiente impacto económico que supone y que da lugar a las correspondientes indemnizaciones.

- Para el aeropuerto: una calidad insuficiente puede originar un incremento en el número de reclamaciones por parte de los usuarios, así como una pérdida de imagen. También, al ser la capacidad de los aeropuertos limitada, la asignación de medios aeroportuarios disponibles para todos los vuelos programados suele estar ajustada en los grandes aeropuertos, de forma que las perturbaciones inciden negativamente en la operatividad del mismo.

- Para el prestador de servicios: las consecuencias más evidentes para el propio prestador de un mal servicio son la posibilidad de pérdida de clientes y las sanciones económicas que pueda recibir.

Para poder realizar los servicios de asistencia en tierra de una forma óptima y más rápida es fundamental la coordinación entre los diversos recursos (personal, aviones, servicios aeroportuarios, etc.). La correcta ejecución de los servicios requiere la participación de muchos expertos que evalúan constantemente la información recibida y toman acciones en consecuencia, por ejemplo, realizar asignaciones y cancelaciones de asignaciones de personal y recursos. Para ello, estos expertos hacen uso de una serie de sistemas aeroportuarios con los que se busca una mejor calidad y rapidez en las respuestas, dando así lugar a una mejora de la productividad del aeropuerto, como se muestra en la ilustración 2.

Ilustración 2: Arquitectura sistemas aeroportuarios

Los coordinadores de los servicios aeroportuarios han de manejar cantidades muy elevadas de información, y, basándose en su experiencia, tomar la mejor de las decisiones en un momento dado. Por tanto, es muy interesante encontrar una solución para liberar de las tareas rutinarias a estos coordinadores, y que éstos puedan enfocarse en la atención y resolución de incidencias.

1.2. Objetivo del proyecto

El objetivo principal del proyecto es construir un sistema experto capaz de asignar y cancelar asignaciones del material disponible en un aeropuerto para poder llevar a cabo los servicios que se deben prestar a los aviones del aeropuerto. Por tanto, este sistema experto debe servir para extraer el conocimiento del coordinador de los servicios aeroportuarios y plasmar dicho conocimiento en un conjunto de reglas gracias a la minería de datos.

La primera idea que se tuvo sobre el proyecto trataba de abarcar varios de los servicios de asistencia en tierra que se prestan a los aviones en cualquier aeropuerto, creando un conjunto de reglas de asignación de recursos por cada uno de estos servicios. Pero se vio que era inviable tenerlo hecho en un tiempo razonable, así que se decidió centrarse en uno de estos servicios y dejar de lado el resto para posteriores trabajos. En un primer lugar se pensó en la tarea de asignación de una zona de estacionamiento a cada avión que fuera aterrizando en el aeropuerto, pero se supo que la asignación de la zona de estacionamiento a cada avión no la realizaba el prestador de servicios, sino que ya era dado por la propia dirección del aeropuerto. Desechada esta opción, de todos los servicios de asistencia en tierra se eligió el de desembarque de pasajeros cuando el avión llega a su zona de estacionamiento porque este servicio es uno de los primeros que se realiza al avión cuando llega al aeropuerto y, por tanto, el coordinador no necesita suponer la realización de servicios no contemplados en el simulador. Por tanto, son las escalerillas existentes en el aeropuerto el material que el sistema experto debe ser capaz de asignar (o cancelar la asignación cuando sea oportuno) a los aviones que aterrizan en el aeropuerto. Además, por supuesto, los resultados obtenidos se pueden extrapolar a otros servicios aeroportuarios.

Tanto los sistemas expertos como la minería de datos se encuadran dentro de la disciplina de la inteligencia artificial. La inteligencia artificial trata la resolución automática de problemas, encargándose de construir modelos que, al ser ejecutados sobre una arquitectura, producen acciones o resultados que maximizan una medida de rendimiento determinada, basándose en la secuencia de entradas percibidas y en el conocimiento almacenado en tal arquitectura. El por qué se ha elegido esta disciplina para abordar este proyecto es debido a que la finalidad del mismo es construir un

modelo (conjunto de reglas) y que, luego, este conjunto de reglas sirva para asignar o cancelar las asignaciones de las escalerillas de forma automática. El motivo de construir un sistema experto es que es necesario emular el razonamiento de un coordinador de servicios aeroportuarios (experto) y el motivo de usar minería de datos para generar el conjunto de reglas es que, gracias a la información recogida por el sistema experto, se tiene una gran cantidad de datos de los que se pueden obtener reglas que expliquen el comportamiento de los mismos aplicando una serie de técnicas de exploración.

1.3. Novedades y retos

Las novedades que se pretenden aportar con este proyecto son las siguientes:

- Adaptación al cambio gracias a la adquisición del conocimiento: al ser un sistema experto que se retroalimenta está en continuo proceso de aprendizaje, siendo capaz de variar el conjunto de reglas generadas si el coordinador por cualquier motivo empieza a basar sus asignaciones en otros parámetros diferentes.

- Da prioridad en la asignación del material: como el sistema experto extrae el modo de razonamiento del coordinador puede dar mayor prioridad a algunos materiales para ser asignados respecto a otros.

- Da a conocer el motivo por el que las cosas suceden: como el sistema experto genera un conjunto de reglas se pueden dar explicaciones y conocer de primera mano por qué se hacen las asignaciones de material.

- Asignación automática del material: en la actualidad la mayoría de los sistemas existentes de este tipo realizan una asignación manual o semiautomática de los recursos disponibles.

- Menor gasto en sanciones económicas debido a retrasos por parte del prestador de servicios: como el sistema experto está orientado a ayudar en sus funciones al coordinador de los servicios (el prestador de servicios) se producirán menos fallos en la asignación del material y, por tanto, menos aviones serán retrasados.

Para que este proyecto pueda aportar las novedades relacionadas anteriormente es necesario primero enfrentarse a una serie de retos relacionados con el dominio en el que el proyecto se centra:

- El dominio aeroportuario es muy complejo, existiendo muchas restricciones y limitaciones, así como una gran variedad de aviones y de materiales (escalerillas, autobuses o jardineras para transportar a los pasajeros desde el avión hasta la terminal o viceversa, maquinaria de carga, etc.).

- El coordinador de servicios se fija en muchos factores (tipo de avión, distancias, horas de salida de los aviones, etc.) para decidir si asigna o no recursos a un avión en particular y, por tanto, su decisión no es en absoluto trivial.

- El tiempo es un factor crítico: si un avión no puede salir a la hora prevista porque ha habido algunos servicios que no se le han podido realizar debido a la ausencia de recursos, esto tiene graves consecuencias para el prestador de servicios, el aeropuerto, la compañía aérea y los pasajeros.

- Sin duda, el mayor reto de este proyecto es la extracción del conocimiento del coordinador de los servicios del aeropuerto. Dependiendo de la situación y de las características técnicas de los aviones, el coordinador se va a fijar en unas cosas u otras para tomar una decisión y, por tanto, es fundamental separar qué aspectos son importantes para el coordinador y cuáles no, así como cuándo influyen éstos en sus decisiones.

1.4. Fases de desarrollo

En este apartado se va a realizar una breve descripción de los hitos de los que se compone el proyecto.

En primer lugar es necesario adquirir una serie de conocimientos previos para la realización del proyecto, acerca del entorno aeroportuario.

A continuación, se debe realizar un estudio de los sistemas con una funcionalidad similar que ya han sido implantados en diferentes aeropuertos, así como analizar las diferentes técnicas que existen, tanto para la extracción del conocimiento o información como para la generación de reglas.

Tras realizar una ponderación de las técnicas estudiadas y elegir las que se van a usar en el desarrollo del proyecto, se debe realizar un diseño del sistema experto de asignación de escalerillas en el que se tenga en cuenta los requisitos indispensables que el sistema debe poseer.

Posteriormente, se debe llevar a cabo la construcción del sistema experto, implementando tanto las técnicas de extracción del conocimiento y generación de reglas elegidas para tal fin como toda la funcionalidad adicional del mismo.

Finalmente, se deben realizar una serie de pruebas para verificar que el sistema experto construido es capaz de generar un conjunto de reglas acorde al método de razonamiento usado para la asignación o cancelación de las escalerillas, así como el proceso inverso (asignación o cancelación automática de las escalerillas basándose en el conjunto de reglas generado).

1.5. Medios empleados

A continuación se van a detallar los medios empleados para la realización del proyecto:

- Medios hardware: ha sido necesario la utilización de un ordenador portátil o PC tanto para llevar a cabo el diseño como la implementación y pruebas del sistema experto.

- Medios software: ha sido necesario utilizar un sistema operativo Windows, en este caso se ha usado Windows XP. También se ha utilizado Microsoft Access 2010 para poder visualizar una muestra suministrada del aeropuerto de Madrid-Barajas con información del material y aviones existentes. También, se ha usado el entorno de ejecución Java versión 1.7.0 [4] y el entorno de desarrollo integrado libre NetBeans versión 7.0.1 [5] para poder tanto programar como ejecutar el sistema experto construido. Otro medio software indispensable en el proyecto es Weka [6], una plataforma de software libre para el aprendizaje automático y minería de datos, escrito en Java.

- Medios humanos: durante todo el desarrollo del proyecto ha sido indispensable la ayuda de diferentes personas expertas en el dominio aeroportuario, sobre todo en las fases de adquisición de conocimientos técnicos previos a la construcción del sistema como en la fase de pruebas del sistema experto construido.

- Otros medios empleados: internet y libros cuyo objetivo es acercar el mundo de los aeropuertos a la gente sin conocimientos específicos de la materia ([3] y [7]).

1.6. Estructura de la memoria

El formato del presente proyecto, está estructurado en 11 capítulos que se describen brevemente a continuación.

El primer capítulo implica una introducción donde se detallan los motivos, objetivos, novedades y retos, fases de desarrollo, medios empleados y estructura del proyecto.

El segundo se centra en el estudio de métodos de extracción del conocimiento, así como la obtención de reglas a partir de una cierta cantidad de información. Así, se permite establecer la base sobre la que construir el desarrollo del presente proyecto.

El tercero aborda el análisis y diseño del sistema desarrollado a través de la identificación de requisitos, casos de uso, arquitectura y diagramas de secuencias.

En el cuarto se relata el modelo de conocimiento y la implementación realizada, mientras que en el quinto se hace hincapié en las pruebas realizadas y en los resultados obtenidos.

En el sexto se encuentra el plan de trabajo del proyecto así como el presupuesto del mismo, para que, a continuación, en el séptimo se detallen las conclusiones obtenidas durante todo el desarrollo del proyecto y se abran nuevas líneas de investigación para trabajos futuros.

En los últimos capítulos se encuentran, por este orden: anexos (con el manual de usuario de la solución dada), bibliografía, acrónimos y el glosario de términos.

2. Estado del arte

2.1. Sistemas de asignación en los aeropuertos

Aunque el sistema experto que se va a desarrollar en este proyecto es totalmente innovador, en la actualidad existen algunos sistemas implantados en algunos aeropuertos con una funcionalidad que se asemeja algo a la idea que se pretende llevar a cabo. Es importante señalar que, como estos sistemas han sido desarrollados por empresas y son de código cerrado y de pago (excepto uno que fue desarrollado por la empresa Terminal Systems, integrada actualmente en Asseco Spain, empresa donde este proyecto ha sido ideado), es imposible obtener el código de dichos sistemas. A continuación se detallan algunos aspectos de estos sistemas.

2.1.1. SADAMA

En 1992 se instaló en el aeropuerto de Madrid-Barajas los sistemas de gestión aeroportuaria, denominados Conoper y SADAMA. El primero de ellos, es la base de datos central de operaciones del aeropuerto, lo que en el lenguaje aeroportuario se conoce como una AODB ("Airport and Obstacle DataBase", en español base de datos operacional del aeropuerto), que contiene toda la información necesaria para manejar las operaciones del aeropuerto. Por su parte, SADAMA (desarrollado por HP) es el sistema inteligente de asignación de medios aeroportuarios. En 1997 AENA decidió poner en marcha un sistema de uso compartido denominado UCA. Este sistema no lo utiliza el aeropuerto sino que lo emplean las compañías y los agentes de handling, permitiendo operar en cualquier mostrador de facturación y embarque, con acceso a su host como si fuera un terminal de su propia oficina. En el año 2003 se produjo una actualización de estas aplicaciones para su instalación por primera vez en la terminal T4 de Madrid-Barajas. En este nuevo proyecto se unificaron las aplicaciones Conoper y SADAMA en una única base de datos, tanto para aeropuertos como para el resto de sistemas de control, vídeo y gestión de incidencias de mantenimiento y así se consiguió una mayor integración [8].

La asignación de los medios aeropuertos mediante la herramienta SADAMA es proporcionada a todos los vuelos y aeronaves que están en el aeropuerto. Los medios que se facilitan son: estacionamientos, puertas de embarque, salas, cintas de llegadas, mostradores de facturación y las medidas de seguridad pertinentes a cada tipo de tráfico. Dicha asignación de medios es realizada de forma automática. Para realizar la asignación de medios a un vuelo determinado se observan los siguientes aspectos de carácter general: la compañía que realiza el vuelo (esto es necesario para saber qué agente de handling le proporciona los medios a la compañía y qué medios están asignados a este agente, esto es, qué mostradores, qué puertas de embarque, etc.) y el

tipo de tráfico del vuelo (es necesario saber si el vuelo es nacional o internacional para proporcionar zonas donde se desarrolle el control de pasaportes o el de aduanas) [9].

2.1.2. Quintiq

Quintiq [10] proporciona funciones avanzadas de planificación y programación de software que da soporte a los aeropuertos, aerolíneas y organizaciones relacionadas con la utilización óptima de los recursos, mejorando la productividad del personal y reduciendo los costos operativos. Es usado, por ejemplo, en el aeropuerto de Bruselas.

Quintiq ofrece soluciones de aviación para una variedad de problemas de planificación, incluyendo:

- Aeropuertos: planificación de las puertas de embarque y de los stands, planificación de la facturación de equipajes, los servicios de asistencia en tierra, programación del personal de tierra y de los turnos.

- Compañías aéreas: la vinculación, turnos, programación de la tripulación, la gestión de la flota y mantenimiento de la flota.

- Control del tráfico aéreo: la capacidad de planificación estratégica, turnos, horarios de los empleados y retrasos en la planificación.

La plataforma de planificación Quintiq ofrece una solución integrada para la planificación estratégica y táctica, programación operativa y en tiempo real para que los aeropuertos y las compañías aéreas puedan optimizar sus recursos clave en el entorno dinámico de la industria de la aviación. Múltiples problemas de planificación como la asignación de recursos del aeropuerto, del personal de tierra, del personal de la aerolínea, de la flota de la aeronave y de los controladores del tráfico aéreo pueden ser así integrados en una única solución.

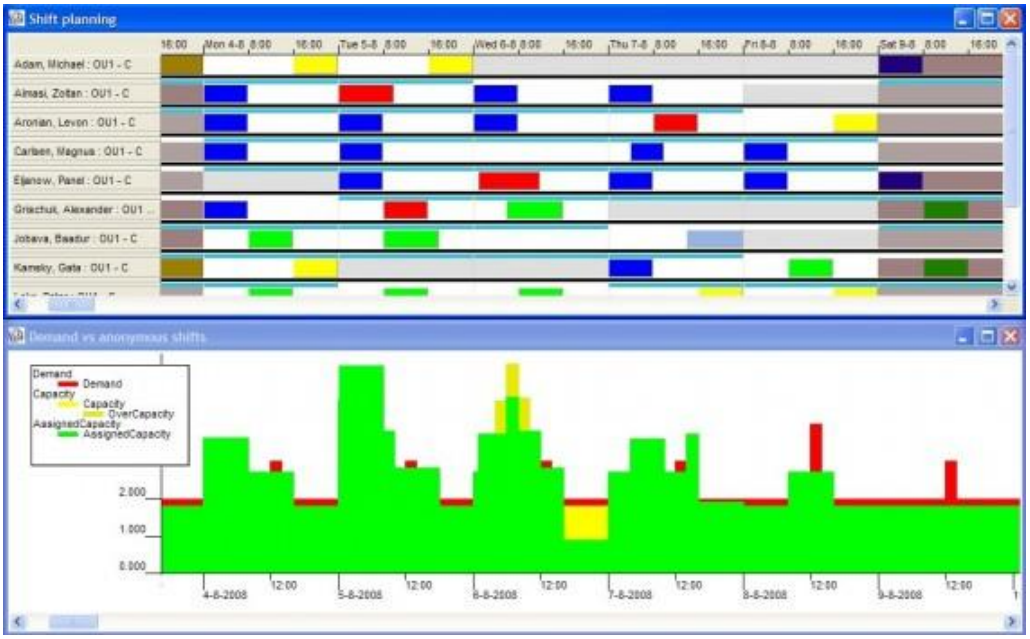


Ilustración 3: Planificación del personal de tierra en Quintiq

2.1.3. SITA Workbridge

SITA Workbridge [11] es un sistema de recursos de software de gestión de la industria de la aviación.

SITA Workbridge optimiza todos los aeropuertos y las operaciones de asistencia en tierra mediante la planificación automatizada y optimizada, la asignación de recursos, gestión de personal, el conocimiento en tiempo real de la situación y soporte para dispositivos de mano.

Las capacidades clave de la solución incluyen:

- Operaciones de control en tiempo real (planificación automática, conocimiento de la situación, la gestión de los equipos de campo, etc.).

- Planificación y programación de turnos (planificación de la capacidad, la optimización y programación de turnos, las simulaciones de escenarios, análisis de costos, etc.).

- Gestión del personal (bases de datos de recursos humanos, gestión de vacaciones y enfermedad, la información de la nómina, etc.).

- Captura de datos móvil (servicios de entrega de documentos y asignar el trabajo con dispositivos de mano y soluciones web).

- Servicio al cliente (servicio web basado en autoservicio, el registro de solicitudes, etc.).

- Control de calidad (informes operativos, indicadores clave de rendimiento, datos de referencia, facturación automatizada y validación, etc.)

SITA Workbridge soporta todas las operaciones del aeropuerto, incluyendo los servicios de pasajeros y de rampa. La plataforma SITA Workbridge de gestión de recursos se compone de varios módulos que se dedica a apoyar a todos los procesos críticos de manejo en tierra.

2.1.4. Sistema de Terminal Systems

Sistema de asignación de recursos hecho en Flash construido por la empresa Terminal Systems [12] (actualmente integrada en Asseco Spain [13]). Este sistema consiste en un simulador del aeropuerto de Madrid-Barajas donde se puede visualizar todos los elementos que componen el aeropuerto: aviones, zonas de estacionamiento, pasarelas, puertas de embarque, cintas de transporte de equipajes, etc. Según va transcurriendo el tiempo algunos aviones van aterrizando en el aeropuerto y otros van despegando, así como se coloca la pasarela que conecta al avión con la terminal cuando éste llega a una zona de estacionamiento próxima a la terminal. Además, el sistema es capaz de realizar una serie de recomendaciones de asignaciones de recursos para poder prestar los servicios en el aeropuerto y el coordinador de aeropuertos es el que finalmente decide si la asignación recomendada se lleva a cabo o no. La gran ventaja de este sistema es que es muy intuitivo para el coordinador ya que es como si estuviera viendo un aeropuerto real visto desde arriba, pero en cambio la gran desventaja es que la asignación de los recursos es manual, ya que es el coordinador en última instancia quien decide si una asignación se realiza o no.

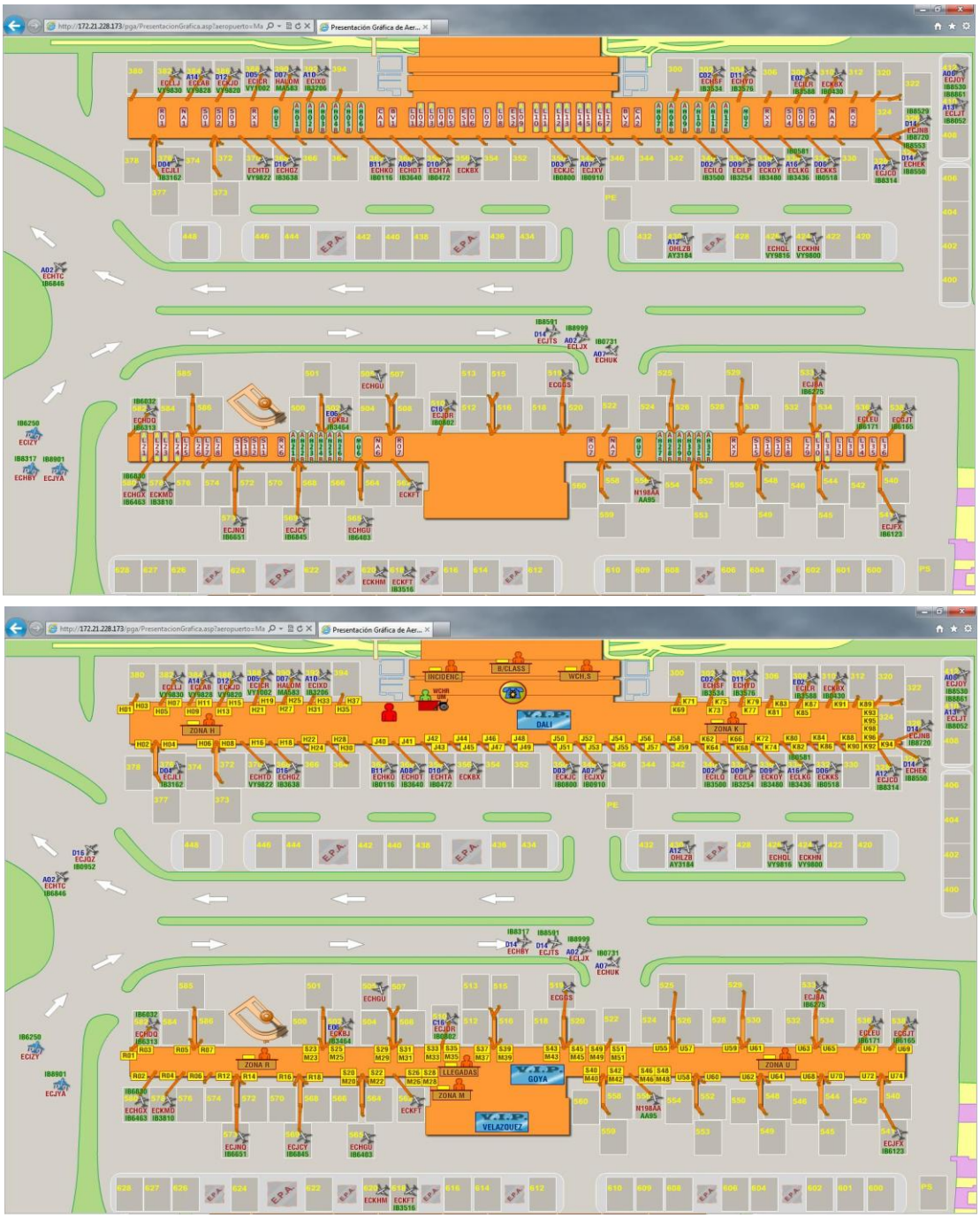


Ilustración 4 (a y b): Interfaz del sistema de Terminal Systems

2.2. Técnicas de adquisición del conocimiento

Para poder plasmar el conocimiento del coordinador de servicios aeroportuarios en un conjunto de reglas es necesario primero usar alguna técnica que permita extraer o adquirir el conocimiento de este experto. Para ello están las técnicas de adquisición del conocimiento, que son métodos que ayudan al traspaso del conocimiento de los expertos (o de fuentes de conocimiento) en un dominio

determinado hacia un formalismo de representación computable del conocimiento. Las técnicas de adquisición del conocimiento más importantes son según [14], [15], [16] y [17] las que se detallan a continuación.

2.2.1. Documentación

Estudio de libros y otro tipo de documentación con el que se identifican los conceptos básicos. Adaptado a lo que se quiere hacer sería construir un sistema capaz de leer documentos y sacar de ahí la información necesaria, por lo que es un sistema bastante difícil de construir.

2.2.2. Entrevistas

Se realiza una serie de preguntas al experto y de ahí se obtiene el conocimiento. Esta técnica es buena para obtener un conocimiento explícito pero mala para obtener conocimiento tácito (implícito, pero sin que la persona sea consciente de poseer ese conocimiento). En el caso del sistema experto aeroportuario sería un sistema que realiza una serie de preguntas (cuestionario) y el coordinador las va respondiendo. El problema principal de esta técnica sería que las preguntas deben ir encaminadas dependiendo de las respuestas del coordinador.

2.2.3. Observación de tareas

Se observa al coordinador trabajando en un problema real habitual. No se debe interferir en la actuación del experto, dejando hacer al experto en todo momento. Orientado al proyecto a realizar sería construir una especie de simulación o juego donde se presenta el dominio típico en el que se envuelve el coordinador, es decir, un aeropuerto donde hay aeronaves, pasajeros, equipajes, material y operarios, y el coordinador debe ir indicando a los operarios que presten los servicios, intentando que ninguno de los aviones sufra retrasos ni en la salida ni en la llegada al aeropuerto.

2.2.4. Incidentes críticos

En esta técnica el coordinador describe cómo resolvió casos límite por su rareza, escasez de información de entrada, síntomas que confunden, etc. En los casos complejos se obtienen detalles que normalmente se pasan por alto. El juego o la simulación descrita en la anterior técnica también puede servir para observar la manera de actuar en estos incidentes críticos (muy pocos operarios disponibles, muchas aeronaves a punto de aterrizar en el aeropuerto, un avión que debe despegar en poco tiempo y todavía necesita que se le realicen varios servicios, etc.).

2.2.5. “Teach back”

En esta técnica el ingeniero del conocimiento (persona o sistema que desea adquirir el conocimiento) explica parte del conocimiento al experto, y entonces el experto hace comentarios. Es útil para detectar aspectos mal comprendidos y clarificar la terminología. Esta técnica, por tanto, se debe combinar con otras técnicas para, antes de hacer “teach back”, obtener gran parte del conocimiento y así clarificar dudas. Adaptado a lo que se quiere hacer sería un sistema que genera una serie de preguntas o afirmaciones hechas al coordinador, que éste te las respondiera o confirmara que son correctas, y que se compare las respuestas dadas con el conocimiento adquirido previamente mediante otra técnica.

2.2.6. 20 cuestiones

En esta técnica el ingeniero del conocimiento piensa en un concepto o escenario que el experto debe averiguar. El experto hace preguntas de tipo sí/no al ingeniero del conocimiento para descubrir el escenario o concepto y debe acertarlo antes de realizar 20 preguntas. Un sistema que usara esta técnica sería difícil de implementar porque necesitaría comprender las preguntas que le envía el coordinador por teclado y dar una respuesta lógica a dichas a preguntas.

2.2.7. Tarjeta de clasificación (“card sorting”)

Es un buen método para capturar la manera en que los expertos comparan y ordenan conceptos, y puede guiar a la relevación de conocimiento sobre clases, propiedades y prioridades. Esta técnica consiste en que al experto se le da un número de tarjetas en la que en cada una de ellas hay un concepto y él clasifica las tarjetas en pilas de tal forma que las tarjetas de la misma pila tienen algo en común. Orientado al proyecto a realizar sería construir un sistema donde en la interfaz se mostraría una serie de elementos del dominio aeroportuario y el coordinador los pudiera clasificar en diferentes pilas e indicarle al sistema vía teclado qué ha originado esa clasificación.

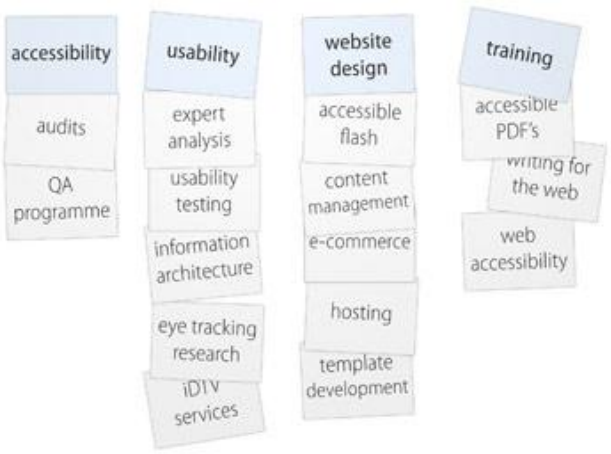


Ilustración 5: Ejemplo de "card sorting"

2.2.8. Método triádico

Consiste en la presentación de tres conceptos aleatorios (de una lista) y preguntar al experto en qué son dos de ellos parecidos y en qué se diferencian del otro. Está pensado para descubrir cómo "ve" el experto el dominio y es muy útil para obtener conocimiento tácito. En el caso del sistema experto aeroportuario sería un sistema que presentara al coordinador tres conceptos del entorno aeroportuario y el coordinador pudiera seleccionar dos e indicarle al sistema vía teclado la semejanza entre esos dos conceptos y la diferencia con respecto al otro concepto.

2.2.9. Clasificación de conceptos ("laddering")

Sirve para organizar conceptos y detectar relaciones. Los pasos a seguir son: haber identificado previamente una serie de conceptos, representar cada concepto en una ficha, el experto los clasifica en grupos en función de varias características y obtención de una jerarquía y de relaciones entre conceptos. Existen diferentes tipos de clasificación de conceptos (ver figura 6):

- Clasificación de conceptos: muestra clases de conceptos y sus subtipos.
- Clasificación de composición: muestra la manera en que un objeto está compuesto de sus partes constituyentes.
- Clasificación de decisiones: muestra los caminos alternativos de acción para una decisión particular.
- Clasificación de atributos: muestra los atributos y valores.
- Clasificación de procesos: muestra procesos (tareas, actividades) y los subprocesos (subtareas, subactividades) de que están compuestos.

Un sistema que usara esta técnica consistiría en una aplicación donde el coordinador pudiera construir representaciones gráficas del dominio en términos de relaciones y clases y atributos, es decir diagramas jerárquicos, y de ahí el sistema extrajera el conocimiento.

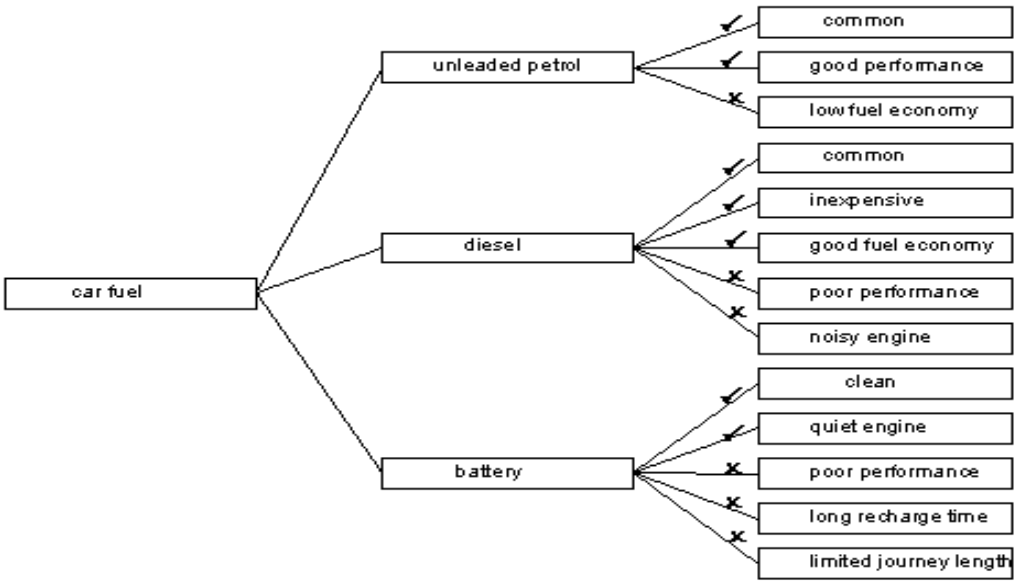


Ilustración 6: Ejemplo de "laddering"

2.2.10. Emparrillado

Permite establecer prioridades entre características diferenciadoras de elementos. Consiste en una tabla de características (filas) y elementos (columnas) y el experto establece una graduación de características aplicables a los elementos (de 1 a n), según se muestra en la figura 7. A menor valor, menor importancia de la característica en el elemento. Se concluye una clasificación de las características y de los elementos. Adaptado a lo que se quiere hacer sería construir un sistema que presentara al usuario una tabla donde en la fila superior estuvieran una serie de elementos del entorno aeroportuario y en la columna de la izquierda una serie de características (atributos) de dichos elementos. El coordinador debería ir rellenando la tabla con números, donde cada número indica el grado de importancia de la característica en el elemento. Después la aplicación extraería el correspondiente conocimiento de dicha tabla.

	Neptuno	Urano	Jupiter	Saturno	Plutón	Marte	Mercurio	Tierra	Venus
Tamaño	7	7	9	7	3	3	1	4	4
Ausencia de Gravedad	8	8	6	7	3	3	2	2	2
Distancia al sol	7	6	4	5	9	3	1	2	2
Frío	8	8	7	8	9	5	1	2	2
Lentitud velocidad orbital	8	9	7	7	9	5	1	4	1
Num. anillos	4	5	6	9	1	1	1	2	1
Num. lunas	5	1	9	7	2	3	1	2	1

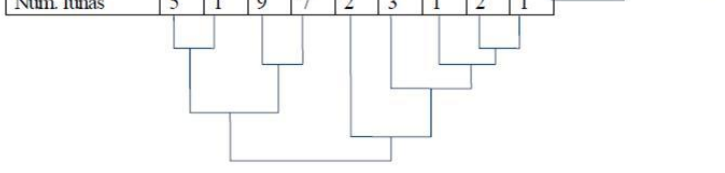


Ilustración 7: Ejemplo de emparrillado

2.3. Elección de la técnica de adquisición del conocimiento

Para elegir la técnica de adquisición del conocimiento a usar en este proyecto se ha ido puntuando cada una de las diferentes técnicas en cuatro aspectos que son de especial relevancia para elegir cuál es la mejor técnica en este proyecto: cómo es de aplicable al dominio, lo fácil que es implementar esa técnica, lo ameno y amigable que es para el coordinador la forma de usar la técnica y la cantidad de conocimiento que se extrae con esa técnica.

Los cuatro aspectos puntuados no tienen la misma importancia ya que, como es lógico, el que se pueda aplicar la técnica al dominio y que se extraiga el mayor conocimiento posible es más importante que sea fácil de implementar o que sea ameno y amigable para el coordinador. Por ello, se darán los siguientes porcentajes para obtener la puntuación final de la técnica: 30% tanto para la aplicabilidad al dominio como al conocimiento extraído y un 20% tanto a la facilidad de implementación como a lo ameno y amigable que sea para el coordinador el sistema.

Estos cuatro aspectos han sido puntuados de 1 a 10, siendo el baremo para asignar un valor a cada uno de ellos el siguiente:

- Aplicable al dominio
  - Puntuación 1-3 → la técnica no se puede acomodar al dominio o su adaptación sería tan costosa que no merece la pena hacerlo.
  - Puntuación 4-6 → la técnica puede ser adaptada para ser aplicada en el dominio, aunque los cambios a realizar no son sencillos de hacer.
  - Puntuación 7-8 → la técnica, con pocos ajustes, puede ser aplicada al dominio.
  - Puntuación 9-10 → los ajustes a realizar para que la técnica se adapte al dominio son nulos o muy escasos y los resultados son muy satisfactorios. La técnica es comúnmente usada en el dominio que se maneja.

- Facilidad de implementación
  - Puntuación 1-4 → el implementar esa técnica adaptada a ese dominio es muy complicada o imposible.
  - Puntuación 5-7 → la implementación de esa técnica tiene su grado de dificultad y su correspondiente coste en el tiempo de desarrollo de la misma.
  - Puntuación 8-9 → la técnica no es muy difícil de implementar, excepto en algún aspecto que sí requiere de unos mayores conocimientos.

- Puntuación 10 → la implementación de la técnica no causa ninguna dificultad.

➢ Ameno y amigable para el coordinador
  - Puntuación 1-3 → la forma en la que se produce la interacción sistema-coordinador hace que el coordinador pierda el interés rápidamente (o no haya interés) por dar información al sistema sobre su forma de razonamiento. Por tanto, no se extrae apenas conocimiento.
  - Puntuación 4-5 → el coordinador puede estar interactuando un rato con el sistema, pero, tras un período relativamente corto de tiempo, pierde el interés.
  - Puntuación 6-8 → el aspecto y las características del sistema hacen que el coordinador tenga predisposición a interactuar con el sistema, pero, según va pasando el tiempo y el coordinador va interactuando con el sistema, la interacción se va haciendo más aburrida para el coordinador.
  - Puntuación 9-10 → el coordinador ve el sistema como un hobby, haciendo que la interacción sistema-coordinador sea muy fluida y que el sistema pueda conseguir obtener mucha información del coordinador.

➢ Conocimiento extraído
  - Puntuación 1-4 → el sistema no consigue mucha información del coordinador o la que consigue no es relevante.
  - Puntuación 5-8 → el sistema es capaz de extraer información útil del coordinador, pero no toda la deseada.
  - Puntuación 9-10 → el sistema obtiene una gran cantidad de información procedente del coordinador y con un alto grado de relevancia.

	Aplicable a dominio (30 %)	Facilidad de implementación (20 %)	Ameno y amigable para coordinador (20 %)	Conocimiento extraído (30 %)	Total
Documentación	2	1	6	8	4,4
Entrevista	7	8	7	9	7,8
Observación de tareas	8	7	9	9	8,3
Incidentes críticos	8	7	9	9	8,3
Teach back	8	8	8	3	6,5
20 cuestiones	6	4	8	6	6
Tarjetas de clasificación	7	7	6	6	6,5
Método triádico	8	6	6	6	6,6
Clasificación de conceptos	5	6	3	7	5,4
Emparrillado	7	9	7	6	7,1

Tabla 2: Puntuación de las técnicas de adquisición del conocimiento

Como se puede ver en la tabla que resume las puntuaciones de las diferentes técnicas, las dos técnicas que mayor puntuación han obtenido y, por tanto, se ha estimado que serán las mejores para usar en este proyecto, son la observación de tareas y los incidentes críticos. Tanto una como otra se puede implementar de la manera dicha anteriormente, mediante una especie de simulación o juego ambientado en un aeropuerto donde hay aeronaves, zonas de estacionamiento, terminales y escalerillas, y el coordinador (la persona que juega) debe ir asignando las escalerillas a los aviones que van aterrizando en el aeropuerto y así poder desembarcar a los pasajeros del avión, intentando que ninguno de los aviones sufra retrasos en su posterior salida.

2.4. Estudios basados en la extracción de reglas

En la actualidad existen varios estudios e investigaciones acerca de los diferentes modelos de inteligencia artificial que son capaces de generar reglas a partir de cierta información. A continuación se exponen las ideas más importantes de los analizados para este proyecto.

2.4.1. Un algoritmo genético de onda de propagación para el problema de la asignación de parkings en un aeropuerto

El GAP ("Gate Assignment Problem", en español problema de asignación de parkings) en las terminales aeroportuarias es un tema importante en las operaciones diarias del control del tráfico aéreo. Los algoritmos genéticos tienen mucho potencial para solventar problemas de optimización combinatoria NP-complejo, tales como el GAP.

Esta investigación [18] usa una representación básica binaria para diseñar un algoritmo genético híbrido para el GAP aeroportuario. Con este fin, un modelo determinista inspirado en el fenómeno de la propagación natural de ondas sobre una superficie líquida es usado para representar la espera de los aviones hasta que puedan ocupar el parking como puntos en un espacio parametrizado, y luego un algoritmo genético tradicional es diseñado para evolucionar los parámetros espaciales para encontrar una solución óptima o cercana a la óptima en el GAP.

En esta investigación se prueba que usar algoritmos genéticos binarios básicos para solventar problemas de optimización combinatoria, que normalmente requieren representaciones de permutación, da, por norma general, muy buenos resultados.

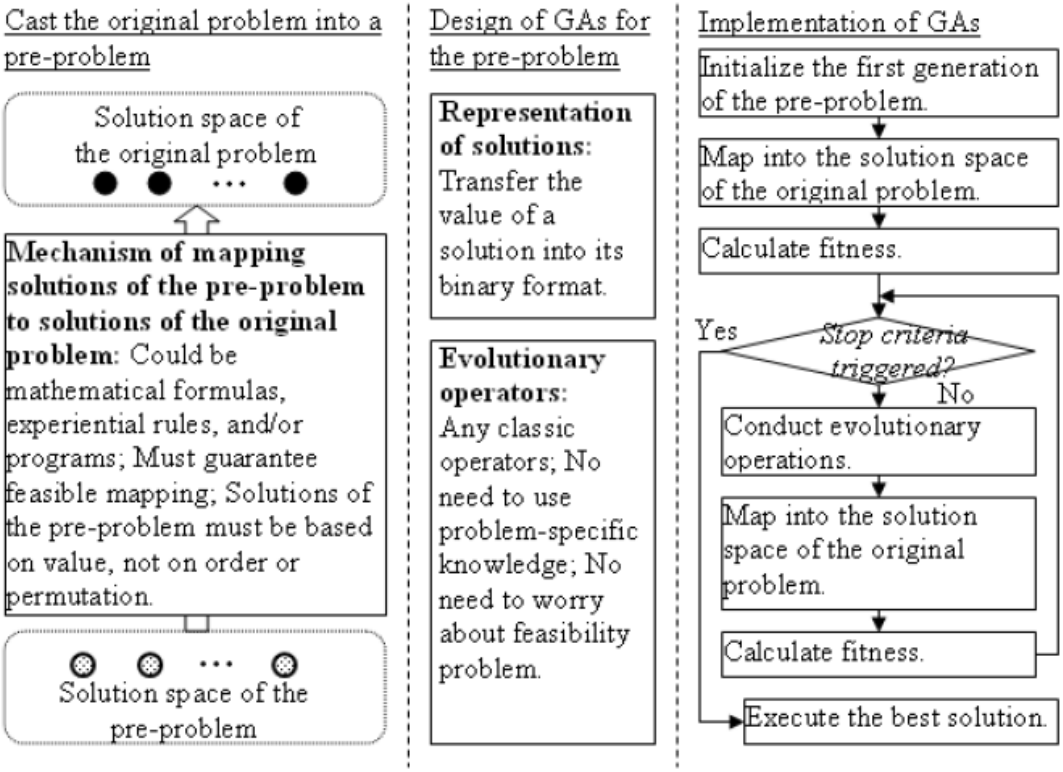


Ilustración 8: Diseño del algoritmo híbrido para GAP aeroportuario

2.4.2. Extracción de reglas simbólicas de redes de neuronas artificiales

Extraer reglas simbólicas de redes de neuronas artificiales entrenadas es una de las áreas que es comúnmente usada para explicar la funcionalidad de las redes de neuronas artificiales (ANNs). El objetivo del artículo [19] es introducir un nuevo algoritmo para extraer reglas simbólicas de ANNs. El nuevo algoritmo es conocido como REANN.

Una red de neuronas artificial de tres capas que se retroalimenta es la base del algoritmo REANN propuesto. Los pasos de REANN son resumidos en la siguiente figura.

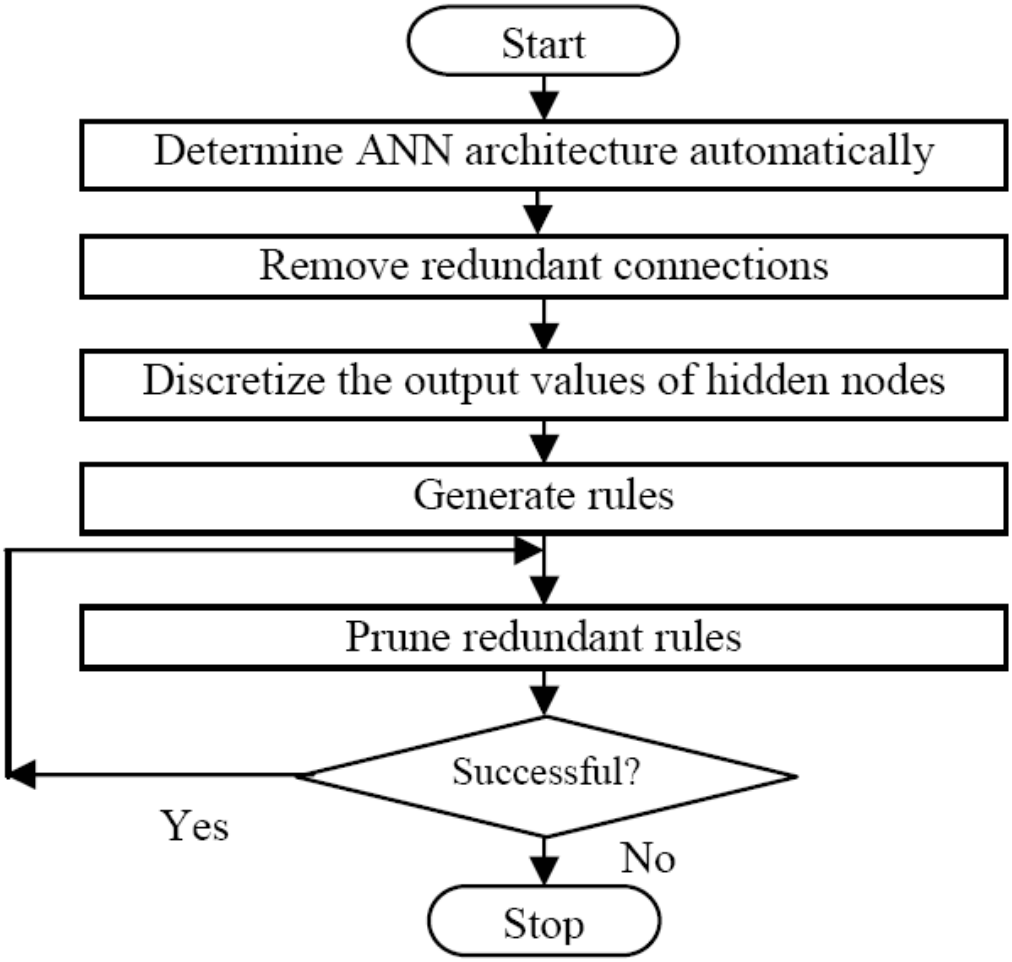


Ilustración 9: Esquema del algoritmo REANN

- Paso 1: crear una arquitectura inicial de la red de neuronas artificial. La arquitectura inicial tiene tres capas: una de entrada, una de salida, y una oculta. Inicialmente, la capa oculta contiene sólo un nodo. El número de nodos en la capa oculta es automáticamente determinado usando un algoritmo básico constructivo. Aleatoriamente se inicializan todos los pesos de las conexiones dentro de un cierto rango pequeño.

- Paso 2: eliminar nodos redundantes de entrada, y conexiones entre nodos de entrada y nodos ocultos y entre nodos ocultos y nodos de salida, usando un algoritmo básico de poda. Cuando la poda se termina, la arquitectura del ANN contiene sólo nodos importantes y conexiones. Esta arquitectura se guarda para el próximo paso.

- Paso 3: discretizar las salidas de los nodos ocultos usando un algoritmo eficiente heurístico de clústering. La razón de la discretización es que las salidas de los nodos ocultos son continuas, por lo tanto las reglas no son fácilmente extraíbles del ANN.

- Paso 4: generar reglas que mapean las relaciones de entradas y salidas.

- Paso 5: podar reglas redundantes generadas en el paso 4. Reemplazar las reglas específicas con reglas más generales.

- Paso 6: comprobar la precisión de la clasificación de la red. Si la precisión está por debajo de un nivel aceptable, es decir la poda de la regla no es exitosa, entonces parar. En caso contrario ir al paso 5.

2.4.3. Adquisición de reglas con un algoritmo genético

En esta investigación [20] se describe la implementación y el funcionamiento de RAGA, un algoritmo genético basado en un sistema de minería de datos adecuado para una extracción del conocimiento supervisada y ciertos tipos de extracción del conocimiento no supervisada desde bases de datos grandes y posiblemente con ruido. RAGA difiere de un algoritmo genético estándar en varios aspectos cruciales, incluyendo los siguientes: sus cromosomas son estructuras simbólicas de longitud variable; además de cruzamientos tipados y operadores de mutación, se usan macromutaciones como operadores de generalización y especialización para explorar eficientemente el espacio de reglas, y evoluciona por defecto una jerarquía de reglas.

El procesamiento de una generación en RAGA involucra 3 pasos:

- Controlado por dos parámetros, el elitismo ordinario copia uno o más de los mejores individuos actuales en la próxima generación para garantizar que los niveles de fitness altos no se eliminarán entre generaciones. El elitismo de clasificación copia cada regla que únicamente cubre al menos un elemento de datos y además contribuye al establecimiento de las reglas finales.

- Tras esto, se aplica una selección proporcional del fitness, cruzamiento, macromutaciones y micromutaciones. Las reglas emparejadas son seleccionadas repetidamente y posiblemente cruzadas. A causa de esto las reglas pueden crecer o encoger durante el proceso. Antes de que las dos reglas hijo entren en la próxima fase, son sometidas a micromutaciones y macromutaciones con ratios limitados. Ya que todas las reglas con confidencia positiva son macromutadas, el tamaño de la población crece durante generaciones.

- Finalmente, el proceso de generación interna se realiza para asegurar la validez y que no exista redundancia. Se pueden borrar varias comparaciones de las reglas para estar en concordancia con lo que está

permitido. Si después de este punto la regla se ha convertido en inválida o es idéntica a una que ya existe en la nueva población, es descartada.

2.4.4. Extraer reglas comprensibles de redes de neuronas vía algoritmos genéticos

En el estudio [21] se presenta un método para extraer reglas precisas y comprensibles de redes de neuronas. El método propuesto usa un algoritmo genético para encontrar una buena topología de redes de neuronas. Esta topología es entonces pasada a un algoritmo de extracción de reglas, y la calidad de las reglas extraídas es entonces retroalimentada al algoritmo genético.

La idea básica es usar un método de búsqueda potente donde una solución candidata es una topología de la red de neuronas. La calidad de esa topología es evaluada respecto a la precisión y a la comprensión de las reglas extraídas. La calidad es retroalimentada al algoritmo de búsqueda, que tiene esta calidad en cuenta para generar nuevas topologías de red candidatas.

Este documento combina la idea de usar un algoritmo genético para evolucionar una topología de red con la idea de extraer reglas de una red de neuronas.

2.4.5. Un algoritmo evolutivo memético para la extracción de reglas

Esta investigación [22] propone un algoritmo memético evolutivo (EMA) para la extracción de reglas. El algoritmo de búsqueda global del algoritmo evolutivo evoluciona la arquitectura del conjunto de reglas, mientras que la búsqueda local es aplicada en cada generación para afinar los parámetros de la regla. Este esquema es logrado gracias al uso de una representación de cromosomas de longitud variable que da flexibilidad en la representación del conjunto de reglas y permite manipulaciones fáciles gracias a los avanzados operadores de variación.

Se propone el uso de una representación de cromosomas de longitud variable para representar la topología del conjunto de reglas. Cada cromosoma representa un conjunto de reglas que consta de varias reglas. Cada cromosoma consta de diferente número de reglas que refleja la complejidad del conjunto de reglas; sin embargo el número de parámetros dentro de cada regla está fijado por el número de atributos de entrada del conjunto de datos. Esta representación es muy eficiente.

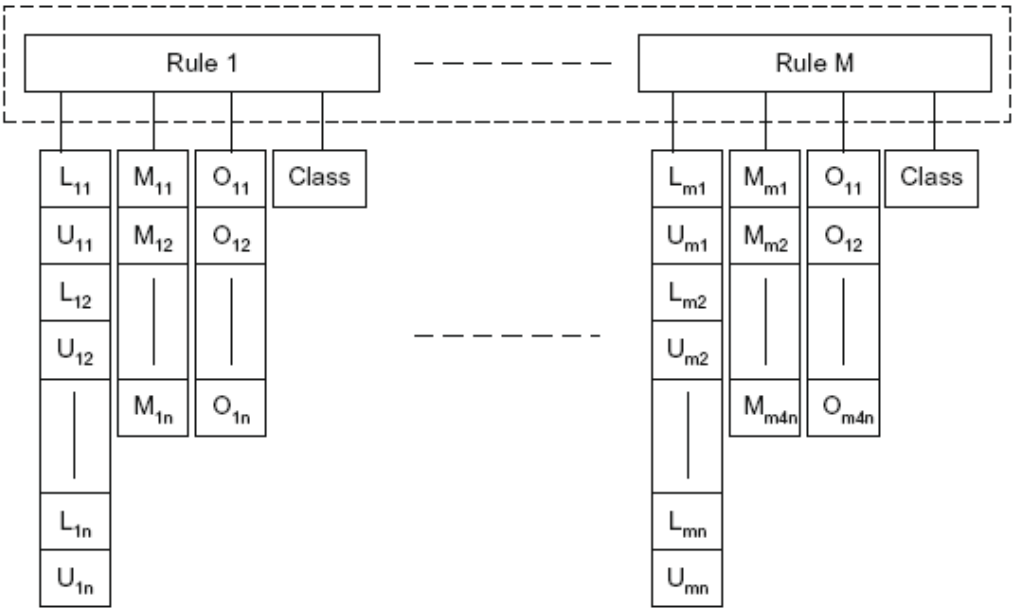


Ilustración 10: Representación genotípica de un cromosoma del conjunto de reglas

La siguiente figura muestra el flujo del algoritmo.

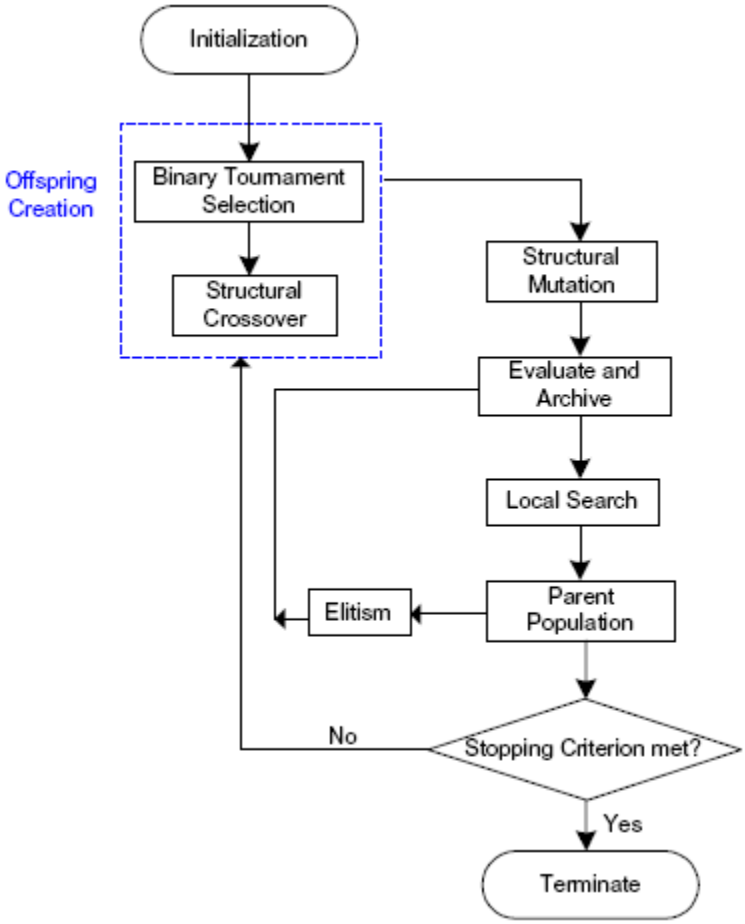


Ilustración 11: Esquema del algoritmo memético evolutivo EMA

2.4.6. Extraer reglas de perceptrones multicapa en problemas de clasificación: una aproximación basada en clústering

Las redes de neuronas han sido aplicadas exitosamente para solventar problemas de minería de datos en varios dominios. En este sentido, los perceptrones multicapa (MPs) logran alta precisión en la clasificación, pero el conocimiento adquirido por estas redes de neuronas es normalmente incomprensible para los humanos.

En este trabajo [23], un algoritmo genético de clústering (CGA) se emplea para extraer reglas de MPs. El método propuesto está basado en los valores de activación de las unidades ocultas y consiste en dos pasos principales. Primero, el CGA se emplea para encontrar clústers de valores de activación de las unidades ocultas. Luego, estos clústers son traducidos a reglas lógicas.

2.4.7. La lógica difusa y el algoritmo evolutivo – dos técnicas en la extracción de reglas de redes de neuronas

En esta investigación [24] se presenta el método REX de extracción de reglas difusas de las redes de neuronas (NN). Está basado en los algoritmos evolutivos. En el proceso de búsqueda del algoritmo evolutivo encontramos un conjunto de reglas que describen el rendimiento de la NN. Un algoritmo evolutivo es también responsable de obtener apropiados conjuntos difusos. Se comparan dos aproximaciones, denominadas REX Pitt y REX Michigan. En REX Pitt, un individuo representa un conjunto de reglas, mientras en REX Michigan representa una regla.

Se proponen dos algoritmos evolutivos especialmente diseñados que buscan reglas que describen el comportamiento de la NN entrenada solventando un problema dado. REX Pitt codifica todas las reglas extraídas de una NN en un cromosoma, mientras REX Michigan está compuesto de dos algoritmos evolutivos. Uno de ellos es responsable de buscar reglas difusas apropiadas, mientras el otro genera conjuntos difusos.

La siguiente figura presenta la idea fundamental de los algoritmos REX. Las reglas (con los conjuntos difusos) encontrados por REX son evaluados sobre la base de respuestas NN en el conjunto de patrones.

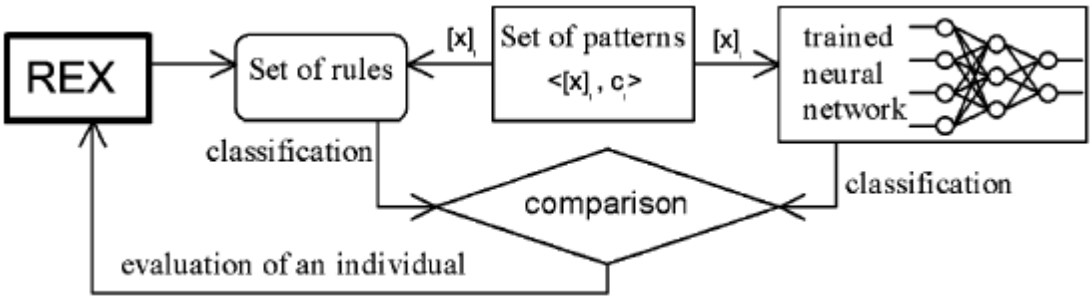


Ilustración 12: Esquema del algoritmo REX

En REX Pitt, un individuo está formado por un conjunto de reglas difusas y una colección de grupos de conjuntos difusos describiendo las variables de entrada. El código de una regla consiste de: un bit de activación, una lista de genes de premisas y un gen de conclusión.

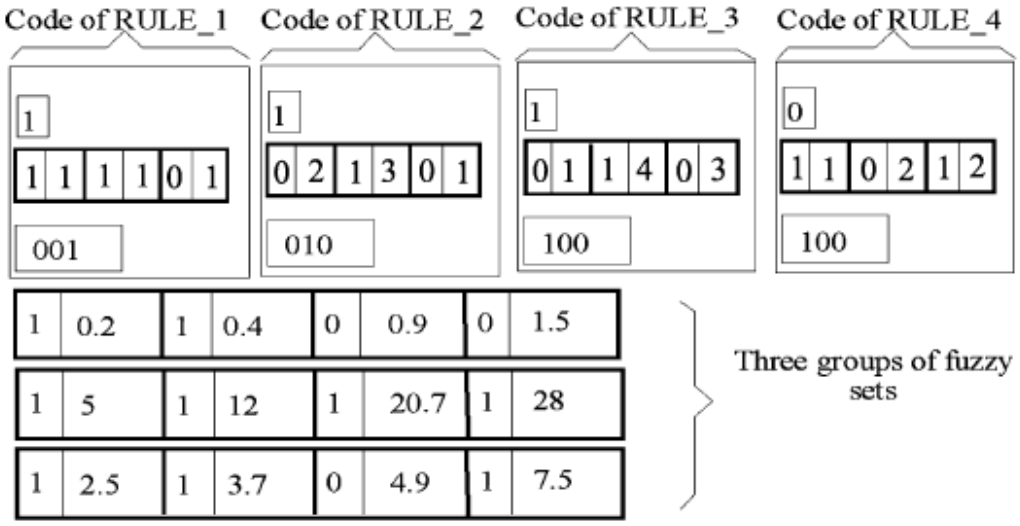


Fig. 7. An example of the chromosome in REX Pitt.

Ilustración 13: Cromosoma en REX Pitt

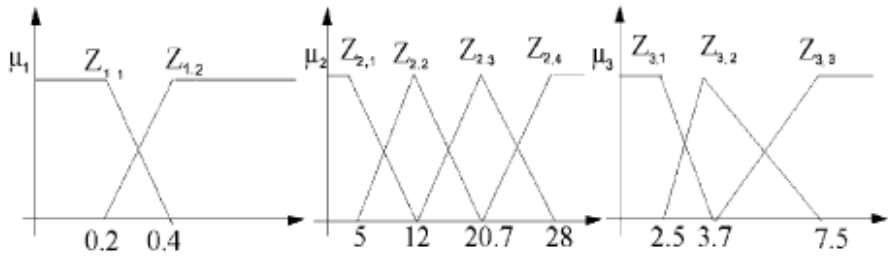


Ilustración 14: Conjuntos difusos en REX Pitt

REX Michigan consiste de dos algoritmos evolutivos especializados alternando uno con otro. El primero EARules busca reglas, mientras el segundo algoritmo evolutivo, que se llama EAFuzzySets, optimiza las funciones de pertenencia de conjuntos difusos aplicados a las reglas.

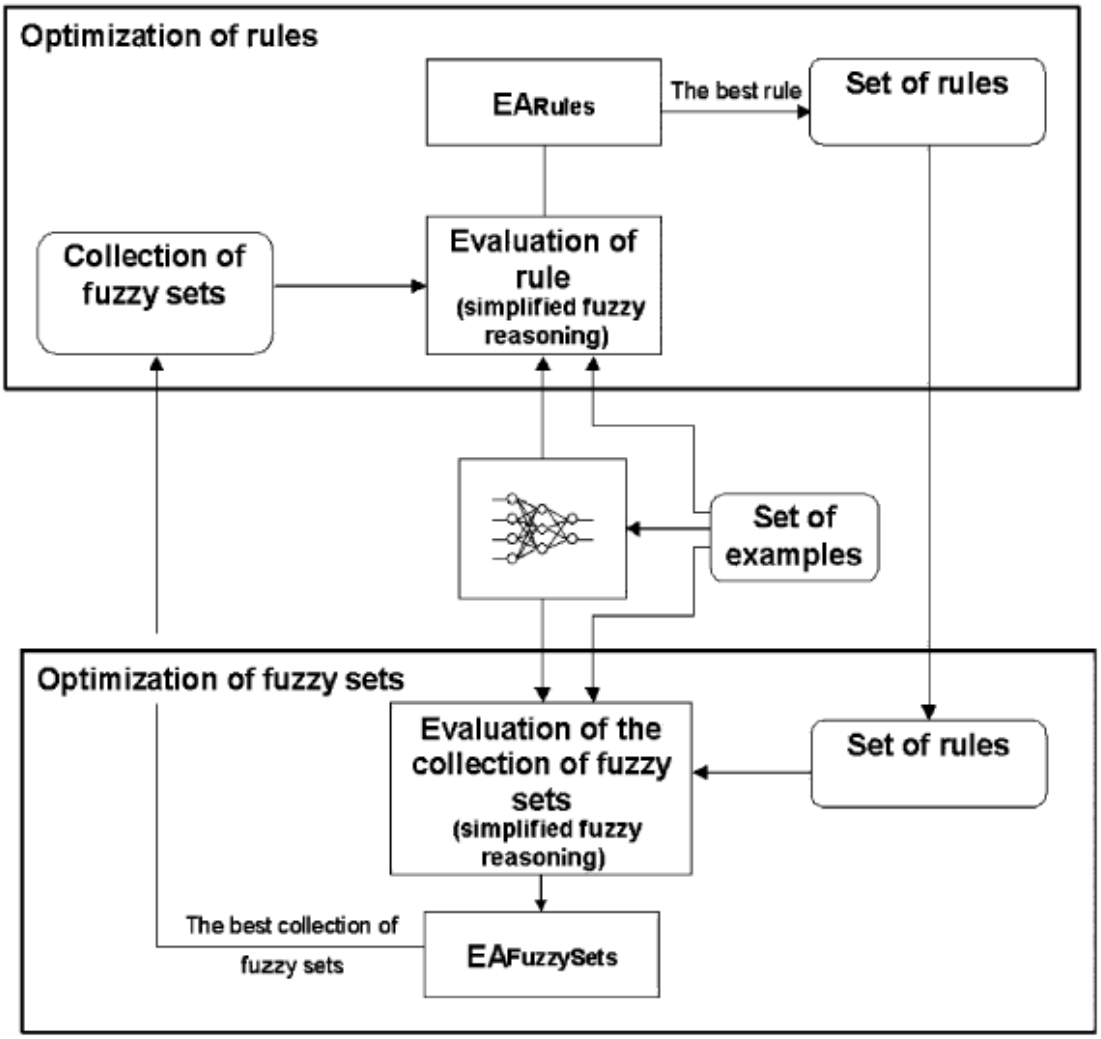


Ilustración 15: Esquema de REX Michigan

2.4.8. Generación de reglas greedy de datos discretos y su uso en la extracción de reglas de redes de neuronas

Este estudio [25] propone un algoritmo GRG ("Greedy Rule Generation", en español generación de reglas greedy), un método nuevo para generar reglas de clasificación de un conjunto de datos con atributos discretos. El algoritmo es avaricioso en el sentido que cada iteración busca la mejor regla a generar. Este método es empleado para extraer reglas de redes de neuronas que han sido entrenadas y podadas para solventar problemas de clasificación.

Se presenta un algoritmo de generación de reglas para datos discretos que puede ser incorporado en un método de extracción de reglas de una red de neuronas. La contribución principal de este documento es la presentación de un algoritmo de generación de reglas greedy que trabaja en conjuntos de datos con atributos discretos. El algoritmo es greedy porque en cada iteración se generan las mejores reglas de clasificación en términos del número de ejemplos de entrenamiento clasificados, el número de atributos involucrados en las condiciones de la regla, y el tamaño del subespacio de entrada cubierto.

El algoritmo GRG propuesto es básicamente un algoritmo secuencial. El algoritmo secuencial empieza con un conjunto de reglas vacío, luego examina el conjunto de datos para generar una regla mejor para cubrir un subconjunto de datos, elimina el subconjunto de datos cubierto, y repite este proceso hasta que no puede ser generada ninguna regla que satisfaga un umbral de rendimiento.

2.5. Modelos de generación de reglas en Weka

Weka ([5] y [26]) es una plataforma de software libre para aprendizaje automático y minería de datos escrito en Java y desarrollado por la Universidad de Waikato. Se utiliza en muchas y muy diferentes áreas, en particular con finalidades docentes y de investigación. Las ventajas de Weka es que es muy portable porque está completamente implementado en Java y puede ejecutarse en casi cualquier plataforma, además de contener una extensa colección de técnicas para preprocesamiento de datos y modelados.

Weka soporta varias tareas estándares de minería de datos, especialmente, preprocesamiento de datos, clústering, clasificación, regresión, visualización, y selección. Todas las técnicas de Weka se fundamentan en la asunción de que los datos están disponibles en un fichero plano o una relación, en la que cada registro de datos está descrito por un número fijo de atributos.

Una de las grandes ventajas que tiene la herramienta Weka es que proporciona un API en Java. Dicho API ofrece un conjunto de métodos que pueden ser invocados por otro software y así no es necesario implementar el algoritmo que construye el modelo.

Los modelos de generación de reglas existentes en Weka basados en árboles de decisión y en inducción de reglas son los siguientes.

2.5.1. Árbol de decisión

Una de las técnicas de predicción que más relevancia ha tenido en el proyecto han sido los árboles de decisión ([27] y [28]). Es una técnica de aprendizaje automático por inducción que permiten identificar conceptos (clases de objetos) a partir de las características de un conjunto de ejemplos que los representan. La información extraída de los mismos queda organizada jerárquicamente en forma de árbol.

El árbol de decisión se construye a base de ir haciendo preguntas sobre características determinadas a los ejemplos y clasificándolos según la respuesta. Por tanto, un árbol de decisión trabaja como un "clasificador". Las diferentes opciones de clasificación (respuesta a las preguntas) son excluyentes entre sí, lo que hace que, a partir de casos desconocidos y siguiendo el árbol adecuadamente, se llegue a una única conclusión o decisión a tomar. Una vez creado el árbol de decisión, cada vez que se presente un nuevo caso únicamente sería necesario recorrer dicho árbol para establecer la clasificación a la que pertenece.

La construcción de un árbol de decisión requiere: un conjunto de ejemplos representativos de lo que se desea aprender (conjunto de entrenamiento), una representación simbólica del conocimiento a través de atributos y sus valores, un algoritmo de aprendizaje de clasificación y un esquema de valoración.

Y se realiza de la siguiente forma:

- Se parte del conjunto de entrenamiento

- Se usa un criterio para seleccionar un atributo "separador" capaz de dividir el conjunto de entrenamiento. El árbol de decisión usa la estrategia de "divide y vencerás".

- El conjunto de entrenamiento es subdivido progresivamente usando los separadores seleccionados como nodos. Los ejemplos caen en las hojas del árbol.

Un árbol de decisión tiene un nodo raíz, nodos intermedios y hojas. Cualquier nodo intermedio puede ser un nodo raíz de un subárbol. Esto conduce a una definición recursiva de árbol de decisión. Cada nodo intermedio y raíz tienen asociados separadores que formulan una pregunta o realizan un test acerca de la existencia o no de una característica en cada caso ejemplo. Esto permite clasificar los ejemplos y determinar cuáles serían los nodos sucesores.

Una hoja en el árbol corresponde a un conjunto de ejemplos que representan una sola clase. La clase de la hoja se asigna por el criterio de la clase a la que pertenezcan la mayoría de los ejemplos en ella. Las hojas del árbol de decisión

representan los conceptos extraídos de manera automática. Una vez construido un árbol de decisión, un nuevo ejemplo desconocido será representante de la clase en donde caiga recorriendo el árbol desarrollado desde la raíz a las hojas.

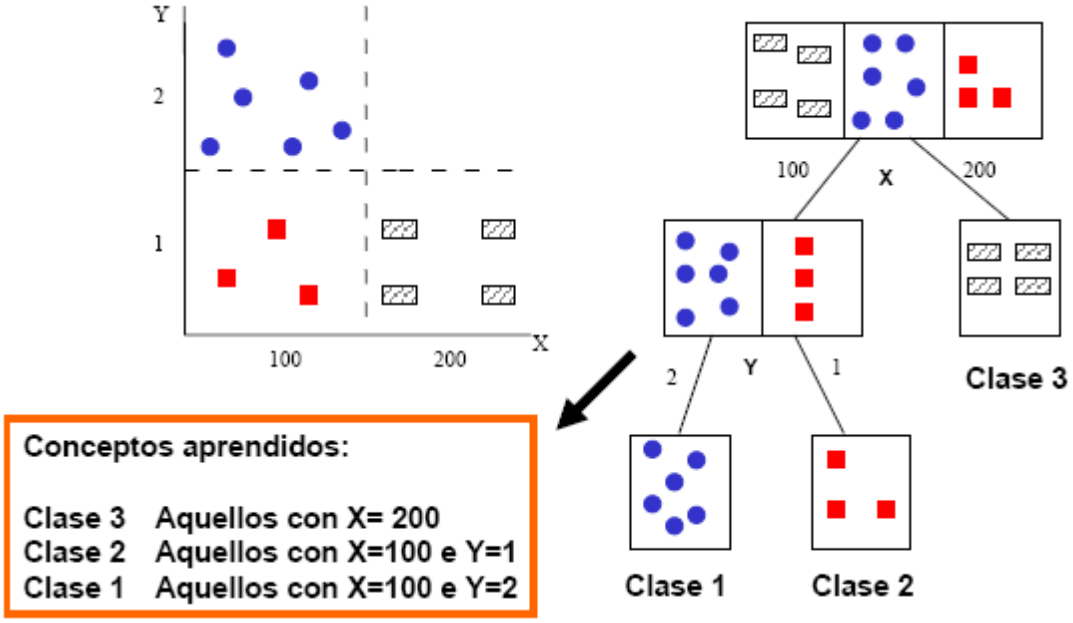


Ilustración 16: Ejemplo de construcción de un árbol de decisión

El rasgo o descriptor a seleccionar debe de cumplir el objetivo de que su posición en algún punto del árbol genere un subárbol tan simple como sea posible y dé una concreta clasificación.

Cuando se construye un árbol de decisión, es necesario tener un medio para determinar tanto los atributos importantes requeridos para la clasificación como el orden de uso de esos atributos importantes.

Es necesario un criterio de selección de separadores. Cada criterio de selección será un test o prueba restringido a una función de solamente uno de los atributos.

En Weka hay implementados varios algoritmos para desarrollar árboles de decisión, citados a continuación.

2.5.1.1. Árbol de decisión J48 (C4.5)

El árbol de decisión J48 está basado en el algoritmo C4.5 ([27] y [29]). El algoritmo C4.5 fue desarrollado por JR Quinlan en 1993, como una extensión (mejora) del algoritmo ID3 que desarrolló en 1986.

El algoritmo C4.5 genera un árbol de decisión a partir de los datos mediante particiones realizadas recursivamente. El árbol se construye mediante la estrategia de profundidad ("depth-first").

El algoritmo considera todas las pruebas posibles que pueden dividir el conjunto de datos y selecciona la prueba que resulta en la mayor ganancia de información. Para cada atributo discreto, se considera una prueba con n resultados, siendo n el número de valores posibles que puede tomar el atributo. Para cada atributo continuo, se realiza una prueba binaria sobre cada uno de los valores que toma el atributo en los datos. En cada nodo, el sistema debe decidir qué prueba escoge para dividir los datos.

Los tres tipos de pruebas posibles propuestas por el C4.5 son:

- La prueba "estándar" para las variables discretas, con un resultado y una rama para cada valor posible de la variable.

- Una prueba más compleja, basada en una variable discreta, en donde los valores posibles son asignados a un número variable de grupos con un resultado posible para cada grupo, en lugar de para cada valor.

- Si una variable A tiene valores numéricos continuos, se realiza una prueba binaria con resultados A <= Z y A > Z, para lo cual debe determinarse el valor límite Z.

Todas estas pruebas se evalúan de la misma manera, mirando el resultado de la proporción de ganancia, o alternativamente, el de la ganancia resultante de la división que producen. Además, hay una restricción adicional: para cualquier división, al menos dos de los subconjuntos Ci_i deben contener un número razonable de casos. Esta restricción, que evita las subdivisiones casi triviales, se tiene en cuenta solamente cuando el conjunto C es pequeño.

El algoritmo C4.5 tiene las siguientes características:

- Permite trabajar con valores continuos para los atributos, separando los posibles resultados en 2 ramas Ai<=N y Ai>N.

- Los árboles son menos frondosos (tienen menos ramas), ya que cada hoja cubre una distribución de clases, no una clase en particular.

- Utiliza el método "divide y vencerás" para generar el árbol de decisión inicial a partir de un conjunto de datos de entrenamiento.

- Se basa en la utilización del criterio de proporción de ganancia (“gain ratio”), definido como I(Xi,C)/H(Xi). De esta manera se consigue evitar que las variables con mayor número de posibles valores salgan beneficiadas en la selección.

- Es recursivo.

En cuanto a los atributos en el algoritmo C4.5:

- Atributos de valores continuos: inicialmente el algoritmo ID3 se planteó para atributos que presentaban un número discreto de valores. Podemos fácilmente incorporar atributos con valores continuos, simplemente dividiendo estos valores en intervalos discretos, de forma que el atributo tendrá siempre valores comprendidos en uno de estos intervalos.

- Medidas alternativas en la selección de atributos: al utilizar la ganancia de información estamos introduciendo involuntariamente un sesgo que favorece a los atributos con muchos valores distintos. Debido a que dividen el conjunto de ejemplos en muchos subconjuntos, la ganancia de información es forzosamente alta. Sin embargo, estos atributos no son buenos predictores de la función objetivo para nuevos ejemplos. Una medida alternativa que se ha usado con éxito es el "gain ratio".

- Atributos con valores perdidos: en ciertos casos existen atributos de los cuales conocemos su valor para algunos ejemplos, y para otros no. En estos casos lo más común es estimar el valor basándose en otros ejemplos de los que sí conocemos el valor. Normalmente se fija la atención en los demás ejemplos de ese mismo nodo. Así, al ejemplo de valor desconocido se le da el valor que más aparezca en los demás ejemplos.

- Atributos con pesos diferentes: en algunas tareas de aprendizaje los atributos pueden tener costes asociados.

A medida que se añaden niveles al árbol, las hipótesis se refinan tanto que describen muy bien los ejemplos utilizados en el aprendizaje, pero el error de clasificación puede aumentar al evaluar los ejemplos. Es decir, clasifica muy bien los datos de entrenamiento pero luego no sabe generalizar al conjunto de prueba. Es debido a que aprende hasta el ruido del conjunto de entrenamiento, adaptándose a las regularidades del conjunto de entrenamiento. Este efecto es, por supuesto, indeseado.

Hay varias causas posibles para que esto ocurra. Las principales son: el exceso de ruido o un conjunto de entrenamiento demasiado pequeño como para ser una muestra representativa de la verdadera función objetivo.

Hay varias estrategias para evitar el sobreajuste en los datos. Pueden ser agrupadas en dos clases: estrategias que frenan el crecimiento del árbol antes de que llegue a clasificar perfectamente los ejemplos del conjunto de entrenamiento y

estrategias que permiten que el árbol crezca completamente, y después realizan una poda.

La post poda ("post prunning") es una variante de la poda y es usada por el C4.5. Consiste en una vez generado el árbol completo, plantearse qué es lo que se debe "podar" para mejorar el rendimiento y de paso obtener un árbol más corto. Pero además el C4.5 convierte el árbol a un conjunto de reglas antes de podarlo. Hay tres razones principales para hacer esto:

- Ayuda a distinguir entre los diferentes contextos en los que se usa un nodo de decisión, debido a que cada camino de la raíz a una hoja se traduce en una regla distinta.

- Deja de existir la distinción entre nodos que están cerca de la raíz y los que están lejos. Así no hay problemas para reorganizar el árbol si se poda un nodo intermedio.

- Mejora la legibilidad: las reglas suelen ser más fáciles de entender.

2.5.1.2. Árbol de decisión BF

El árbol de decisión BF ([30] y [31]) está basado en el algoritmo de primero el mejor ("best-first"). Este algoritmo combina las ventajas de los algoritmos en profundidad y en anchura. En el algoritmo primero el mejor sigue un único camino, pero puede cambiarse a otro camino que parece más prometedor que el que se está siguiendo. Se utiliza una función de evaluación (heurística) para cada nodo y se expande el nodo mejor evaluado no expandido.

2.5.1.3. Árbol de decisión "stump"

Se denomina así a los árboles de decisión con sólo una decisión (árbol de un sólo nivel [32]). Esto es, un árbol de decisión con un nodo interno (la raíz) que está conectado inmediatamente a los nodos terminales. Una decisión "stump" hace una predicción basada en el valor de un único rasgo de entrada.

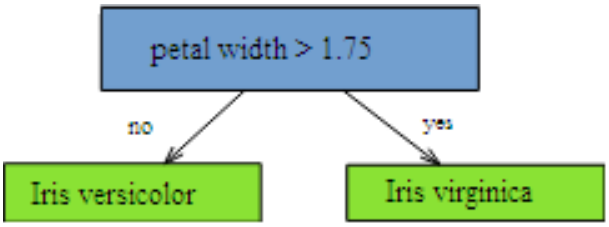


Ilustración 17: Ejemplo de árbol de decisión "stump"

2.5.1.4. Árbol de decisión aleatorio

El árbol de decisión aleatorio ([33]) es construido considerando K atributos elegidos de forma aleatoria en cada nodo y no se realiza poda.

2.5.1.5. Árbol de decisión REP

El árbol de decisión o regresión REP ([33]) es construido usando información de ganancia/varianza y lo poda usando una poda con error reducido (con "backfitting", procedimiento iterativo usado para ajustar un modelo aditivo generalizado). El algoritmo sólo clasifica valores para atributos numéricos de una vez. Los valores desconocidos son tratados gracias a la división de las instancias correspondientes en dos fragmentos.

2.5.1.6. Árbol de decisión "SimpleCart"

El árbol de decisión SimpleCart hace uso del algoritmo CART ([34]). Este algoritmo utiliza el criterio basado en índice Gini (medida para elegir la variable y el punto exacto donde más discrimina el valor de la variable objetivo) para el caso de la clasificación, llevando a cabo una búsqueda exhaustiva de todas las posibles podas para minimizar el porcentaje de clasificación incorrecta.

Como principales ventajas de CART se pueden destacar: no es necesario hipótesis acerca de la distribución de las variables, puede trabajar tanto con datos categóricos como continuos, es invariante a transformaciones monótonas de los datos y permite combinaciones lineales entre las variables.

2.5.2. Inducción de reglas

Este método consiste en derivar un conjunto de reglas para clasificar casos. Aunque los árboles de decisión pueden producir un conjunto de reglas, los métodos de inducción de reglas ([35]) generan un conjunto de reglas independientes que no necesariamente formarán un árbol. Debido a que un inductor de reglas no fuerza divisiones en cada nivel y puede adelantarse, puede ser capaz de encontrar diferentes, y en algunos casos mejores, patrones para la clasificación. A diferencia de los árboles, las reglas generadas pueden no cubrir todas las posibles situaciones, además de que éstas pueden tener conflictos en sus predicciones, en cuyo caso es necesario elegir una regla para seguir.

Un método común para resolver conflictos es asignar confiabilidad a las reglas y usar la que tenga mayor confiabilidad.

2.5.2.1. Inducción de reglas JRip

Basado en un algoritmo de reglas proposicional, RIPPER. Dicho algoritmo consta de 2 fases ([33]): la primera fase construye un conjunto de reglas inicial usando un algoritmo de inducción de reglas llamado IREP* (es la fase de construcción y consta de las subfases de crecimiento y poda); la segunda fase optimiza el conjunto de reglas inicialmente obtenido (fase de optimización).

2.5.2.2. Inducción de reglas Nnge

Basado en el algoritmo del vecino más cercano usando ejemplares generalizados no anidados (que son hiperrectángulos que pueden ser vistos como reglas "if-then"). Es un híbrido entre algoritmos basados en instancias y los de inducción de reglas. Aprende incrementalmente, primero clasifica y luego generaliza cada nuevo ejemplo. La generalización consiste en fusionar la nueva instancia con el ejemplar de la misma clase más próximo ([36]).

2.5.2.3. Inducción de reglas PART

Se basa en el algoritmo de "divide y vencerás". Construye un árbol de decisión parcial C4.5 en cada iteración y construye una regla de la "mejor" hoja ([33]).

2.6. Elección del modelo de generación de reglas

Debido a que es imposible implementar cada uno de los modelos de generación de reglas explicados en el apartado 2.4 del "Estado del arte" para probar cuál puede ser el que más se adecúa a las características del problema, se ha decidido elegir uno de los modelos de generación de reglas existentes en Weka y explicados en el apartado 2.5 del "Estado del arte", ya que así el desarrollo del sistema experto será mucho más sencillo. Más concretamente, el modelo que se va a utilizar en este proyecto es el del árbol de decisión J48, dejando para trabajos futuros probar con el resto de modelos para ver si se mejora en algo o no los resultados obtenidos en este proyecto.

2.7. Elección del lenguaje de programación

Como el árbol de decisión J48 es uno de los modelos que puede ser creado con la herramienta Weka y gracias a que, como se ha dicho anteriormente, esta herramienta proporciona un API en Java ([33]) que permite usar la funcionalidad de esta herramienta en otro software, parece lógico tomar la decisión de usar este

lenguaje de programación para el proyecto debido a las ventajas que aporta en el desarrollo del mismo. Otro punto a favor de este lenguaje es que la persona que va a desarrollar el sistema posee un conocimiento mayor sobre este lenguaje de programación que con respecto a otros como el lenguaje de programación C o C++.

Java ([3] y [37]) fue diseñado por James Gosling, de Sun Microsystems, en 1990, como software para dispositivos electrónicos de consumo, como calculadoras y microondas. En su primera puesta en el mercado este lenguaje fracasó y tuvo que ser Bill Joy, cofundador de Sun y uno de los desarrolladores principales del sistema operativo Unix de Berkeley, quien lo sacara de dicho fracaso, ya que juzgó que Internet podría llegar a ser el campo de juego adecuado para disputar a Microsoft su primacía casi absoluta en el terreno del software, y vio en este lenguaje el instrumento idóneo para llevar a cabo estos planes. Para poderlo presentar tuvo que realizar una serie de modificaciones de diseño y así adaptarlo al propósito mencionado. Y así Java fue presentado en sociedad en agosto de 1995. Actualmente es, sin duda, una de los lenguajes de programación más extendidos entre los desarrolladores.

El éxito de Java reside en varias de sus características. Java es un lenguaje sencillo, orientado a objetos e independiente de plataforma, por lo que un programa hecho en Java se ejecutará igual en un PC con Windows que en una estación de trabajo basada en Unix. También hay que destacar su seguridad, su capacidad multihilo, su robustez o lo integrado que tiene el protocolo TCP/IP, lo que lo hace un lenguaje ideal para Internet. Pero es su sencillez, portabilidad y seguridad lo que le han hecho un lenguaje de tanta importancia.

3. Análisis y diseño

En este apartado se van a definir los requisitos, arquitectura de hardware y software, componentes, módulos y datos del sistema experto de asignación de los recursos a los servicios que se deben prestar a los aviones del aeropuerto. Como ya se ha comentado anteriormente, debido al alto número de servicios que se realizan a los aviones en el aeropuerto (asistencia de pasajeros, asistencia de equipajes, asistencia de limpieza y servicio de la aeronave, etc.), el sistema experto que se va a implementar se centra en el servicio de asistencia de operaciones en pista, que ayuda en el desembarque de los pasajeros de los aviones cuando éstos llegan al aeropuerto. Por tanto, el sistema experto debe ser capaz de asignar las escalerillas que hay en el aeropuerto a los aviones que aterrizan en el mismo, para que, cuando los aviones lleguen a su zona de estacionamiento y llegue también la escalerilla asignada, se produzca el desembarque de los pasajeros, impidiendo que se produzcan retrasos en las salidas de los aviones.

El sistema experto consiste en un simulador en el que se reproduce un aeropuerto. Dicho simulador puede ser arrancado en dos modos: uno en el que el coordinador de los servicios aeroportuarios asigna y cancela manualmente las escalerillas y otro modo en el que la asignación y cancelación es automática.

Cuando se arranca el simulador en el primero de estos modos, comienzan a aterrizar una serie de aviones en el aeropuerto simulado y de forma automática se les asigna, a cada uno de estos aviones, una zona de estacionamiento para que se dirijan a los mismos y se paren sobre esa zona. Además, en el simulador se puede visualizar una serie de escalerillas que se encuentran siempre dentro de la pista de material. El objetivo del coordinador de los servicios aeroportuarios es realizar las asignaciones y cancelaciones de asignaciones de escalerillas correspondientes, para que los pasajeros de los aviones que aterrizan puedan desembarcar. Cuando se realice una asignación, la escalerilla debe moverse hasta la zona de estacionamiento del avión asignado y cuando la escalerilla y el avión se encuentren sobre la zona de estacionamiento comenzará el desembarque de los pasajeros. El coordinador puede, en cualquier momento, obtener información de los aviones, escalerillas y zonas de estacionamiento. Cuando el coordinador lo considere oportuno, con los datos obtenidos de la simulación (asignaciones y cancelaciones) se construyen árboles de decisión J48 para obtener el conjunto de reglas que representa el razonamiento seguido por el coordinador y tanto los árboles de decisión J48, como las reglas y los datos son guardados en ficheros.

En el segundo modo se cargan los árboles de decisión J48 generados en una simulación previa y se produce la asignación y cancelación automática de las escalerillas. Cuando el coordinador de servicios arranca el simulador en este modo

deben comenzar de nuevo a aterrizar los aviones en el aeropuerto simulado y a cada uno de estos aviones se les debe asignar de forma automática una zona de estacionamiento. Aparte de esto, cuando el aeropuerto se encuentra en algún estado que, al pasarlo como entrada de los árboles de decisión J48 cargados tiene como salida una escalerilla y una zona de estacionamiento, se produce la asignación o cancelación de forma automática de esa escalerilla con esa zona de estacionamiento. Este proceso continúa hasta que el coordinador cierra la aplicación.

3.1. Establecimiento de Requisitos Software

3.1.1. Introducción

En esta sección se lleva a cabo la definición y análisis de los requisitos a partir de la información facilitada por la empresa Asseco Spain, obteniendo un catálogo detallado de requisitos que debe cumplir el software, así como la prioridad de los mismos.

La tabla que va a ser usada para describir los requisitos software se muestra a continuación:

Identificador			
Nombre			
Descripción			
Prioridad		**Estabilidad**	
Claridad		**Verificabilidad**	
Necesidad			

Tabla 3: Formato de tablas de requisitos software

Donde cada campo significa lo siguiente:

- Identificador: código que identifica de forma unívoca cada uno de los requisitos.
- Nombre: nombre descriptivo del requisito.
- Descripción: explicación clara y concisa del requisito que se está especificando.
- Prioridad: orden de cumplimiento de un requisito. A mayor prioridad, mayor urgencia para realizarlo.

- Estabilidad: probabilidad de cambio de un requisito. A mayor estabilidad, menor posibilidades de que dicho requisito se vea modificado.

- Claridad: cataloga la no ambigüedad de un requisito. Cuanta mayor claridad tenga, menor ambigüedad contendrá.

- Verificabilidad: facilidad para comprobar que el sistema cumple un requisito.

- Necesidad: esencialidad. Interés del cliente en la realización de un requisito.

Por otro lado, con ánimo de ofrecer una estructura que facilite la realización y comprensión de los requisitos, se van a organizar en ocho categorías:

- Requisitos software funcionales: especifican qué tiene que hacer el software. Definen el propósito del software.

- Requisitos software de rendimiento: especifican valores numéricos para variables de rendimiento.

- Requisitos software de interfaz: especifican hardware y/o software con el que el sistema o componentes del sistema deben interactuar o comunicarse.

- Requisitos software de operación: aquellos que indican cómo va a realizar el sistema las tareas para las que ha sido construido, garantizando los niveles de servicio requeridos.

- Requisitos software de recursos: especifican los recursos y capacidades que debe tener el sistema.

- Requisitos software de comprobación: especifican las limitaciones que afectan a cómo el software debe verificar los datos de entrada y salida.

- Requisitos software de documentación: especifican los requisitos específicos del proyecto para la documentación, además de los contenidos en los estándares.

- Requisitos software de seguridad: especifican los requisitos para asegurar el sistema contra amenazas de confidencialidad, la integridad y la disponibilidad.

- Requisitos software de disponibilidad: especifican la disponibilidad del sistema en distintos ámbitos.

A continuación se detallan los requisitos software del sistema.

3.1.2. Requisitos software funcionales

RSF0010			
Nombre	Mostrar simulador		
Descripción	El sistema mostrará un simulador de aeropuertos con sus terminales, aviones, escalerillas, pistas de aterrizaje de los aviones y pista de material		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 4: Requisito software funcional RSF010

RSF0020			
Nombre	Arranque simulador		
Descripción	El sistema permitirá el comienzo del simulador aeroportuario		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 5: Requisito software funcional RSF020

RSF0030			
Nombre	Parada simulador		
Descripción	El sistema permitirá la terminación de la simulación aeroportuaria		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 6: Requisito software funcional RSF030

RSF0040			
Nombre	Pausa simulador		
Descripción	El sistema permitirá la puesta en pausa del simulador aeroportuario		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 7: Requisito software funcional RSF040

RSF0050			
Nombre	Ayuda simulador		
Descripción	El sistema permitirá consultar información sobre iconos existentes en el simulador aeroportuario		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Deseable		

Tabla 8: Requisito software funcional RSF050

RSF0060			
Nombre	Asignación escalerillas		
Descripción	El sistema permitirá la asignación de escalerillas a los aviones que aterrizan en el aeropuerto simulado		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 9: Requisito software funcional RSF060

RSF0070			
Nombre	Cancelación escalerilla		
Descripción	El sistema permitirá quitar o cancelar asignaciones de escalerillas		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 10: Requisito software funcional RSF070

RSF0070			
Nombre	Encolar asignaciones escalerillas		
Descripción	El sistema permitirá asignar una escalerilla a varios aviones del simulador, realizando su labor por orden de asignación		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 11: Requisito software funcional RSF080

RSF0090			
Nombre	Mostrar información aviones		
Descripción	El sistema mostrará información relevante sobre los aviones del simulador		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 12: Requisito software funcional RSF090

RSF00100			
Nombre	Mostrar información escalerillas		
Descripción	El sistema mostrará información relevante sobre las escalerillas del simulador		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 13: Requisito software funcional RSF100

RSF0110			
Nombre	Mostrar información eventos simulador		
Descripción	El sistema mostrará información relevante sobre eventos ocurridos en el simulador aeroportuario.		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Deseable		

Tabla 14: Requisito software funcional RSF110

RSF0120			
Nombre	Generar reglas		
Descripción	El sistema permitirá la generación de reglas de asignación (o cancelación de asignación) de escalerillas en base a las asignaciones realizadas durante la simulación		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 15: Requisito software funcional RSF120

RSF0130			
Nombre	Cargar reglas		
Descripción	El sistema permitirá que se cargue en el simulador un conjunto de reglas generado para cancelar asignaciones o asignar de forma automática las escalerillas		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 16: Requisito software funcional RSF130

RSF0140			
Nombre	Borrar información asignaciones		
Descripción	El sistema permitirá eliminar toda información existente acerca de las asignaciones (o cancelaciones de asignaciones) de las escalerillas		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Deseable		

Tabla 17: Requisito software funcional RSF140

RSF0150			
Nombre	Movimiento aviones		
Descripción	El sistema mostrará cómo los aviones se van moviendo cuando aterrizan por la pista de aterrizaje hasta llegar a su zona de estacionamiento		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 18: Requisito software funcional RSF150

RSF0160			
Nombre	Asignación automática de zona de estacionamiento		
Descripción	El sistema asignará automáticamente la zona de estacionamiento a los aviones que aterricen en el aeropuerto simulado		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 19: Requisito software funcional RSF160

RSF0170			
Nombre	Despegue aviones		
Descripción	El sistema mostrará cómo los aviones desaparecen de la zona de estacionamiento cuando ya no tiene pasajeros a bordo		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 20: Requisito software funcional RSF170

RSF0180			
Nombre	Movimiento escalerillas		
Descripción	El sistema mostrará cómo las escalerillas se van moviendo por la pista de material para irse trasladando de unas zonas de estacionamiento a otras		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 21: Requisito software funcional RSF180

RSF0190			
Nombre	Desembarque pasajeros		
Descripción	El sistema llevará a cabo el desembarque o bajada de los pasajeros del avión cuando una escalerilla está situada en la zona de estacionamiento del avión y éste también se encuentra en el mismo		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 22: Requisito software funcional RSF190

RSF0200			
Nombre	Retraso despegue avión		
Descripción	El sistema mostrará un mensaje cuando un vuelo ha sido retrasado debido a que no se ha desembarcado a los pasajeros del avión, parándose la simulación		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 23: Requisito software funcional RSF200

3.1.3. Requisitos software no funcionales

3.1.3.1. Requisitos software de rendimiento

RSR0010			
Nombre	Rápida ejecución		
Descripción	El sistema experto debe ser capaz de empezar a ejecutarse en menos de 5 segundos desde que el usuario lo inicia.		
Prioridad	Baja	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Opcional		

Tabla 24: Requisito software de rendimiento RSR0010

RSR0020			
Nombre	Rápido cierre sistema		
Descripción	El sistema experto debe ser capaz de cerrarse de forma completa en menos de 5 segundos desde que el usuario lo cierra		
Prioridad	Baja	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Opcional		

Tabla 25: Requisito software de rendimiento RSR0020

RSR0030			
Nombre	Rápida asignación escalerillas		
Descripción	El sistema experto debe ser capaz de asignar escalerillas (o cancelar la asignación) en menos de 5 segundos desde que el usuario realiza la asignación (o cancelación)		
Prioridad	Baja	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Deseable		

Tabla 26: Requisito software de rendimiento RSR0030

3.1.3.2. Requisitos de interfaz

RSI0010			
Nombre	Diseño minimalista		
Descripción	En el sistema destacará lo esencial del sistema, no incluyendo los elementos sobrantes.		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Deseable		

Tabla 27: Requisito software de interfaz RSI0010

RSI0020			
Nombre	Independencia arquitectura y sistema operativo		
Descripción	El sistema debe ser independiente de la arquitectura y del sistema operativo		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Deseable		

Tabla 28: Requisito software de interfaz RSI0020

3.1.3.3. Requisitos de operación

RSO0010			
Nombre	Periodos críticos		
Descripción	Los periodos críticos de acceso al sistema serán en los momentos en los que el sistema deba generar nuevas reglas ya sea por cambios en las normativas en los aeropuertos, porque las reglas generadas son erróneas o porque no se ha generado ninguna regla aún.		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 29: Requisito software de operación RSO010

RSO0020			
Nombre	Configuración simulador		
Descripción	En cada aeropuerto que se use el sistema experto debe producirse la modificación del simulador para que muestre el aeropuerto en cuestión		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 30: Requisito software de operación RSO020

3.1.3.4. Requisitos de recursos

RSRC0010			
Nombre	Tiempo de respuesta base datos		
Descripción	La base de datos deberá ser capaz de atender múltiples consultas con un tiempo de respuesta inferior a 3 segundos		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 31: Requisito software de recursos RSRC0010

3.1.3.5. Requisitos de comprobación

RSC0010			
Nombre	Detección de fallos		
Descripción	El sistema debe detectar posibles fallos que ocurren durante su ejecución, como puede ser un error en la conexión con la base de datos o el no poder generar las reglas		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Esencial		

Tabla 32: Requisito software de comprobación RSC010

RSC0020			
Nombre	Mensajes de error		
Descripción	El sistema deberá mostrar mensajes de errores claros y coherentes con el fallo que se haya producido		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Deseable		

Tabla 33: Requisito software de comprobación RSC0020

3.1.3.6. Requisitos de documentación

RSD0010			
Nombre	Manual de ayuda		
Descripción	La ayuda deberá estar siempre accesible para el usuario		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Deseable		

Tabla 34: Requisito software de documentación RSD0010

3.1.3.7. Requisitos de seguridad

RSS0010			
Nombre	Cumplir leyes de seguridad en aeropuertos		
Descripción	El movimiento de aviones y escalerillas del simulador aeroportuario debe cumplir con las leyes de seguridad existentes en el aeropuerto simulado		
Prioridad	Baja	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Opcional		

Tabla 35: Requisito software de seguridad RSS0010

3.1.3.8. Requisitos de disponibilidad

RSDP0010			
Nombre	Disponibilidad con dispositivos		
Descripción	El sistema debe ser compatible con portátiles y ordenadores de sobremesa		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Alta
Necesidad	Deseable		

Tabla 36: Requisito software de disponibilidad RSDP0010

3.2. Especificación de casos de uso

3.2.1. Introducción

El objetivo de este apartado es describir los pasos que deben realizarse para llevar a cabo cada uno de los procesos que se pueden ser llevados a cabo con el sistema. Son los llamados casos de uso.

Para ello se hará uso de la siguiente tabla:

	Identificador
Nombre	
Descripción	
Actor	
Precondiciones	
Postcondiciones	
Escenario	
Escenarios alternativos	

Tabla 37: Formato de tablas de casos de uso

Donde cada campo significa lo siguiente:

- Identificador: código que identifica de forma unívoca cada caso de uso.
- Nombre: nombre descriptivo del caso de uso.
- Descripción: explicación clara y concisa del caso de uso que se está especificando.
- Actor: tipo de usuario de la aplicación.
- Precondiciones: condiciones que se deben cumplir previamente para poder realizar una determinada operación.
- Postcondiciones: estado que presenta el sistema tras la ejecución de una determinada operación.
- Escenario: ejecución del caso de uso paso a paso con un orden determinado.
- Escenarios alternativos: condiciones excepcionales que afectan al escenario y respuestas del sistema ante esas situaciones.

La especificación de diferentes casos de uso se presenta a continuación.

3.2.2. Casos de uso

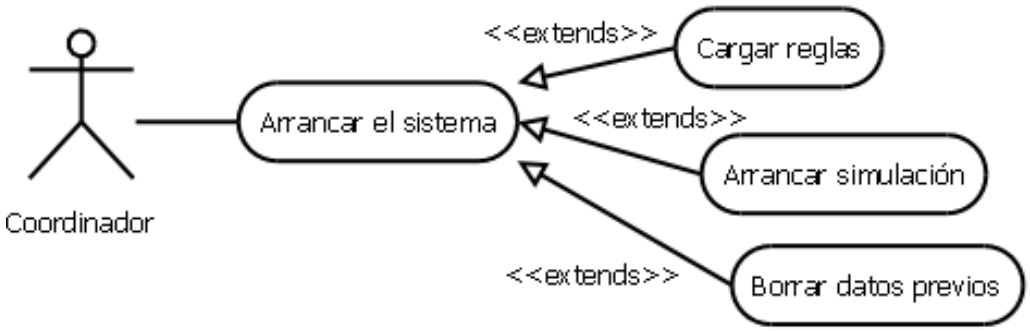


Ilustración 18: Casos de uso del arranque del sistema

	CU0010
Nombre	Arrancar el sistema
Descripción	Un coordinador quiere usar el sistema
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación
Postcondiciones	- El coordinador ve la interfaz del sistema
Escenario	- El coordinador arranca la aplicación
Escenarios alternativos	No existen

Tabla 38: Caso de uso CU0010

	CU0020
Nombre	Arrancar simulación
Descripción	Un coordinador quiere iniciar la simulación aeroportuaria
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación
Postcondiciones	- La simulación aeroportuaria comienza
Escenario	- El coordinador arranca la aplicación - El coordinador pulsa botón para iniciar simulación
Escenarios alternativos	Error de conexión con la base de datos: - La aplicación muestra un mensaje indicando el motivo por el que no se ha podido realizar conectar o consultar la base de datos de la aplicación

Tabla 39: Caso de uso CU0020

CU0030	
Nombre	Cargar reglas
Descripción	Un coordinador quiere cargar un conjunto de reglas de asignación de escalerillas
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - No haber comenzado la simulación aeroportuaria - Tener ficheros que contengan el conjunto de reglas de asignación de escalerillas
Postcondiciones	- Comienza una simulación donde se asignan y se cancelan las asignaciones de las escalerillas de forma automática
Escenario	- El coordinador arranca la aplicación - El coordinador pulsa botón para cargar reglas
Escenarios alternativos	No existe fichero de reglas: - La aplicación muestra un mensaje indicando que no se ha podido cargar el fichero de reglas, parando la simulación

Tabla 40: Caso de uso CU0030

CU0040	
Nombre	Borrar datos previos
Descripción	Un coordinador quiere borrar toda la información aprendida por el sistema experto
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - No haber comenzado la simulación aeroportuaria
Postcondiciones	- Se borra toda la información almacenada de simulaciones anteriores así como los ficheros generados con el conjunto de reglas
Escenario	- El coordinador arranca la aplicación - El coordinador pulsa botón para borrar datos previos
Escenarios alternativos	No existen datos previos: - La aplicación no muestra ningún mensaje

Tabla 41: Caso de uso CU0040

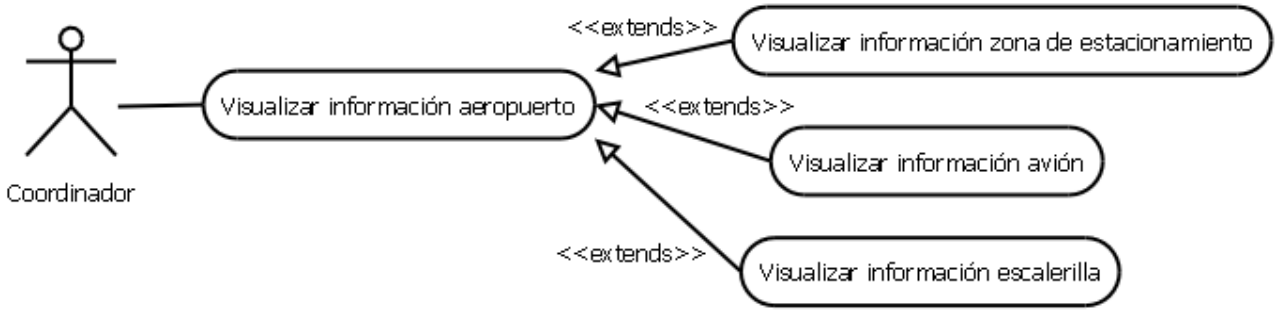


Ilustración 19: Casos de uso de visualización de información del aeropuerto

CU0050	
Nombre	Visualizar información aeropuerto
Descripción	Un coordinador quiere ver información sobre alguno de las zonas de estacionamiento, aviones o escalerillas de la simulación aeroportuaria
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - Haber comenzado la simulación aeroportuaria
Postcondiciones	- Se muestra información del elemento seleccionado
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación o carga reglas - El coordinador pincha sobre el elemento sobre el que quiere información
Escenarios alternativos	No hay ningún elemento en la selección: - La aplicación no muestra ninguna información

Tabla 42: Caso de uso CU0050

CU0060	
Nombre	Visualizar información zona de estacionamiento
Descripción	Un coordinador quiere ver información sobre alguno de las zonas de estacionamiento
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - Haber comenzado la simulación aeroportuaria
Postcondiciones	- Se muestra información de la zona de estacionamiento
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación o carga reglas - El coordinador pincha sobre una zona de estacionamiento
Escenarios alternativos	No hay ningún elemento en la selección: - La aplicación no muestra ninguna información

Tabla 43: Caso de uso CU0060

CU0070	
Nombre	Visualizar información avión
Descripción	Un coordinador quiere ver información sobre alguno de los aviones
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - Haber comenzado la simulación aeroportuaria
Postcondiciones	- Se muestra información del avión
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación o carga reglas - El coordinador pincha sobre un avión
Escenarios alternativos	No hay ningún elemento en la selección: - La aplicación no muestra ninguna información

Tabla 44: Caso de uso CU0070

CU0080	
Nombre	Visualizar información escalerilla
Descripción	Un coordinador quiere ver información sobre alguno de las escalerillas
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - Haber comenzado la simulación aeroportuaria
Postcondiciones	- Se muestra información de la escalerilla
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación o carga reglas - El coordinador pincha sobre una escalerilla
Escenarios alternativos	No hay ningún elemento en la selección: - La aplicación no muestra ninguna información

Tabla 45: Caso de uso CU0080

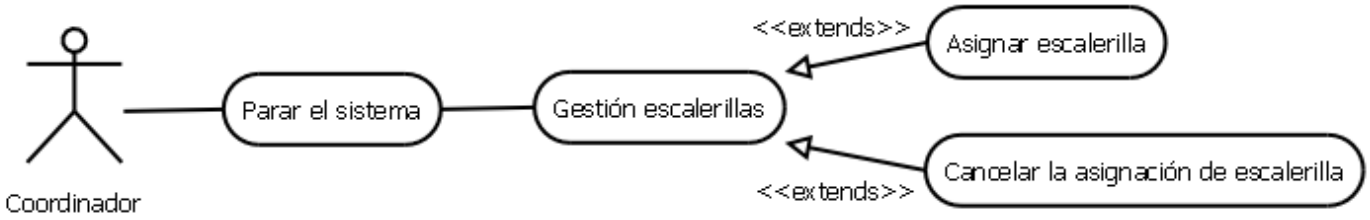


Ilustración 20: Casos de uso de la parada y gestión de escalerillas

CU0090	
Nombre	Parar el sistema
Descripción	Un coordinador quiere parar o pausar el sistema para asignar o cancelar la asignación de una escalerilla
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - Haber comenzado la simulación aeroportuaria
Postcondiciones	- Se pausa la simulación
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación - El coordinador pincha sobre botón de pausa
Escenarios alternativos	No hay

Tabla 46: Caso de uso CU0090

CU0100	
Nombre	Gestión escalerillas
Descripción	Un coordinador quiere asignar o cancelar la asignación de una escalerilla
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - Haber comenzado la simulación aeroportuaria - Haber pausado la simulación - El coordinador pincha sobre una escalerilla
Postcondiciones	- Se asigna o se cancela la asignación de la escalerilla seleccionada
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación - El coordinador pincha sobre botón de pausa - El coordinador pincha sobre una escalerilla - La aplicación muestra la opción de asignar o cancelar asignación de escalerilla
Escenarios alternativos	No hay ninguna escalerilla seleccionado: - La aplicación no muestra ninguna opción para asignar o cancelar asignación de escalerilla

Tabla 47: Caso de uso CU0100

CU0110	
Nombre	Asignar escalerilla
Descripción	Un coordinador quiere asignar una escalerilla
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - Haber comenzado la simulación aeroportuaria - Haber pausado la simulación - El coordinador pincha sobre una escalerilla
Postcondiciones	- Se asigna una escalerilla
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación - El coordinador pincha sobre botón de pausa - El coordinador pincha sobre una escalerilla - El sistema muestra la opción de asignar o cancelar asignación de escalerilla - El coordinador selecciona la opción de asignar escalerilla - El coordinador selecciona la escalerilla y la zona de estacionamiento asignada del avión al que se le va a realizar este servicio - Se confirma asignación
Escenarios alternativos	La escalerilla seleccionada está asignada a otro avión: - La asignación se encola y cuando la escalerilla termine de realizar todos los servicios previstos lleva a cabo ese servicio

Tabla 48: Caso de uso CU0110

CU0120	
Nombre	Cancelar la asignación de escalerilla
Descripción	Un coordinador quiere cancelar la asignación de una escalerilla hecha previamente
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación - Haber comenzado la simulación aeroportuaria - Haber pausado la simulación - El coordinador pincha sobre una escalerilla
Postcondiciones	- Se cancela la asignación de una escalerilla
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación - El coordinador pincha sobre botón de pausa - El coordinador pincha sobre una escalerilla - El sistema muestra la opción de asignar o cancelar asignación de escalerilla - El coordinador selecciona la opción de cancelar asignación de escalerilla - El coordinador selecciona la escalerilla y la zona de estacionamiento asignada del avión al que se le va a cancelar la realización de este servicio - Se confirma cancelación de asignación
Escenarios alternativos	La escalerilla seleccionada tiene servicios asignados aparte del cancelado: - La escalerilla pasa a realizar el siguiente servicio al cancelado La escalerilla seleccionada no está asignada: - No se puede confirmar la cancelación de asignación

Tabla 49: Caso de uso CU0120

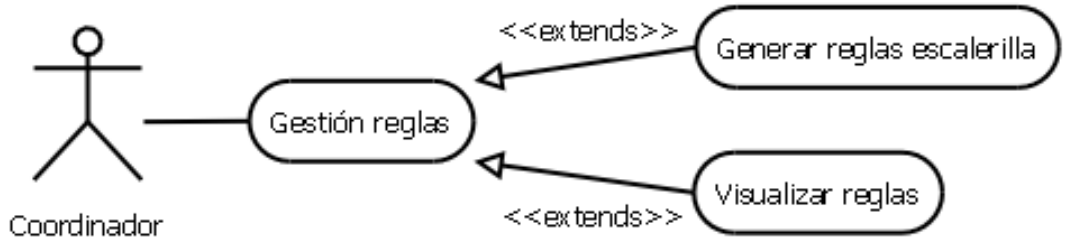


Ilustración 21: Casos de uso de la gestión de reglas

CU0130	
Nombre	Gestión reglas
Descripción	Un coordinador quiere generar el conjunto de reglas de asignación y cancelación de escalerillas o visualizar dicho conjunto de reglas
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- Arrancar la aplicación
Postcondiciones	- Se genera el conjunto de reglas o se visualiza
Escenario	- El coordinador arranca la aplicación
Escenarios alternativos	No hay

Tabla 50: Caso de uso CU0130

CU0140	
Nombre	Generar reglas escalerillas
Descripción	Un coordinador quiere generar el conjunto de reglas de asignación y cancelación de escalerillas
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- El coordinador arranca la aplicación - El coordinador arranca simulación - El coordinador realiza una serie de asignaciones o cancelaciones de asignación de las escalerillas
Postcondiciones	- Se genera el conjunto de reglas
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación - El coordinador realiza una serie de asignaciones o cancelaciones de asignación de las escalerillas - El coordinador pausa la simulación - El coordinador pulsa en botón de generar reglas
Escenarios alternativos	No se tienen datos de asignación o cancelación de escalerillas para generar el conjunto de reglas (o los datos que se tienen no son suficientes): - La aplicación muestra un mensaje indicando que no hay datos suficientes para generar el conjunto de reglas

Tabla 51: Caso de uso CU0140

CU0150	
Nombre	Generar reglas escalerillas
Descripción	Un coordinador quiere generar el conjunto de reglas de asignación y cancelación de escalerillas
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- El coordinador arranca la aplicación - El coordinador arranca simulación - El coordinador realiza una serie de asignaciones o cancelaciones de asignación de las escalerillas
Postcondiciones	- Se genera el conjunto de reglas y se guarda en un fichero
Escenario	- El coordinador arranca la aplicación - El coordinador arranca simulación - El coordinador realiza una serie de asignaciones o cancelaciones de asignación de las escalerillas - El coordinador pausa la simulación - El coordinador pulsa en botón de generar reglas
Escenarios alternativos	No se tienen datos de asignación o cancelación de escalerillas para generar el conjunto de reglas (o los datos que se tienen no son suficientes): - La aplicación muestra un mensaje indicando que no hay datos suficientes para generar el conjunto de reglas

Tabla 52: Caso de uso CU0150

CU0160	
Nombre	Visualizar reglas
Descripción	Un coordinador quiere visualizar el conjunto de reglas de asignación o cancelación de escalerillas
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- El coordinador arranca la aplicación - El sistema ha generado un conjunto de reglas de asignación o cancelación de escalerillas
Postcondiciones	- Se visualiza el conjunto de reglas
Escenario	- El coordinador arranca la aplicación - El coordinador pulsa el botón de ver reglas
Escenarios alternativos	No se ha generado ningún conjunto de reglas: - La aplicación muestra un mensaje indicando que no se puede mostrar el conjunto de reglas creado porque no existe

Tabla 53: Caso de uso CU0160

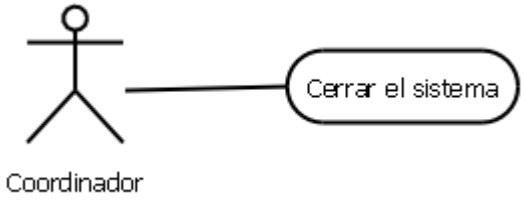


Ilustración 22: Caso de uso de cierre del sistema

CU0170	
Nombre	Cerrar el sistema
Descripción	Un coordinador quiere cerrar la aplicación o sistema
Actor	Un coordinador de servicios aeroportuarios
Precondiciones	- El coordinador arranca la aplicación
Postcondiciones	- Se cierra el sistema
Escenario	- El coordinador arranca la aplicación - El coordinador pulsa el botón para cerrar el sistema
Escenarios alternativos	No hay

Tabla 54: Caso de uso CU0170

3.3. Definición de la arquitectura del sistema

A continuación se procederá a definir la arquitectura general del sistema, así como la descomposición de la misma en distintos niveles y lo que abarca cada uno de éstos.

La arquitectura que se propone es una arquitectura de tres componentes: la vista o interfaz, el controlador y el generador de reglas. Así se consigue tenerlo todo

organizado, o sea, hacer una distinción entre la presentación y las diferentes partes de la lógica de toda la aplicación.

- El controlador: como su nombre indica es el organizador de la aplicación, decide qué hacer según interactúe el usuario con la aplicación. Es el encargado, básicamente, de responder a las acciones solicitadas por el usuario y transmitir los datos devueltos a la vista para que los presente al usuario. Es la lógica de los elementos que hay en el aeropuerto de la aplicación (cómo los aviones, cuando aterrizan, se mueven a las zonas de estacionamiento, lleva el tiempo en el que los aviones deben despegar del aeropuerto, el movimiento de las escalerillas, etc.). Esta capa realiza consultas a una base de datos para conocer algunas características importantes del aeropuerto: número y tipo de escalerillas, clase y subclase de aviones, tiempos de embarque y desembarque de los pasajeros según avión, etc.

- El generador de reglas: la parte encargada de, una vez recopilados todos los datos de la simulación, aplicar el modelo elegido (árbol de decisión J48) para obtener el conjunto de reglas que definen el razonamiento a seguir para asignar las escalerillas a los aviones que van aterrizando en el aeropuerto. Es la capa que se alimenta del API de la herramienta Weka, tanto para construir el fichero de datos con la información como la posterior construcción del modelo.

- La vista: recibe, por parte del controlador, los nuevos datos a mostrar, y los representa de forma gráfica para mejor entendimiento del usuario y para que pueda seguir interactuando con la aplicación. Es la interfaz de la aplicación (lo que ve el coordinador cuando arranca el sistema).

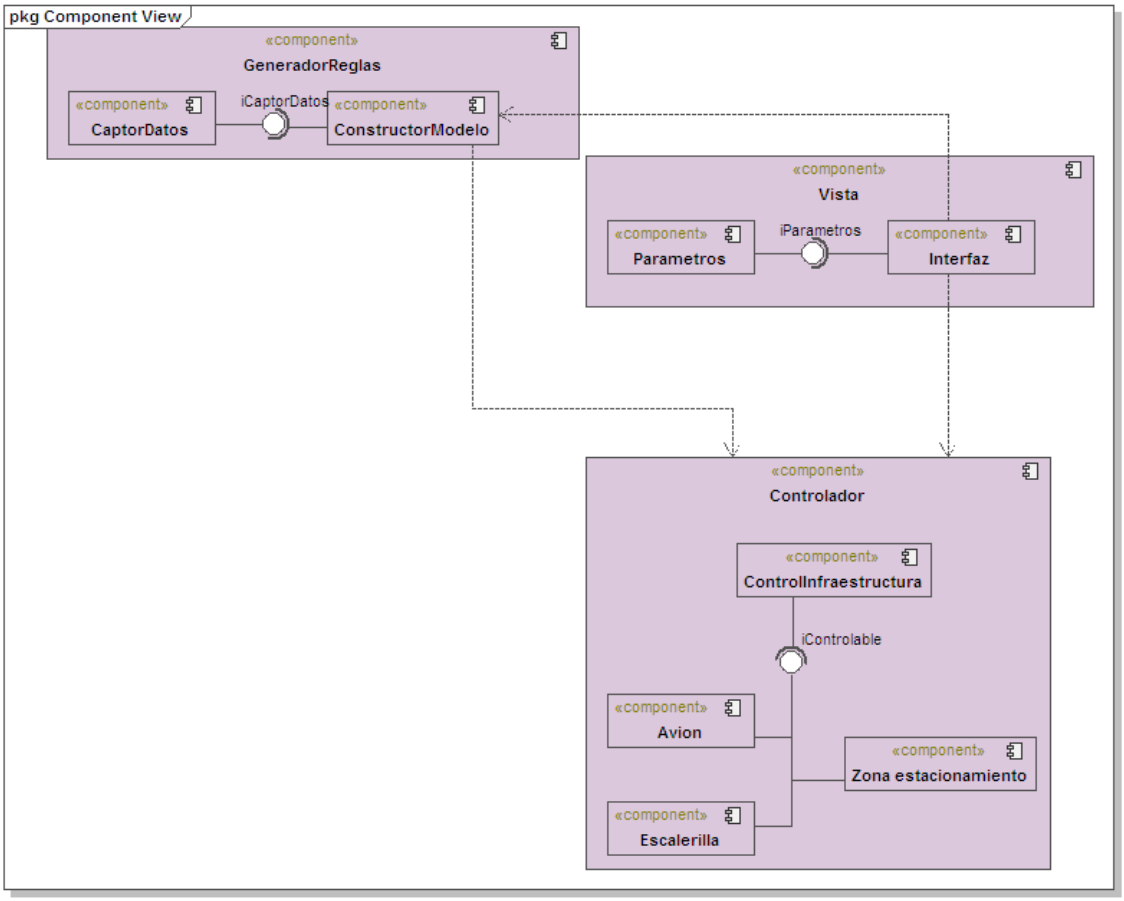


Ilustración 23: Diagrama de subsistemas

3.4. Descripción de diagramas de secuencia

A continuación se muestra una serie de diagramas de secuencia asociados a los casos de uso más representativos de nuestro sistema, procurando que el conjunto de diagramas permitan hacernos una más completa idea del funcionamiento interno del mismo.

3.4.1. Diagrama de secuencia de visualizar información

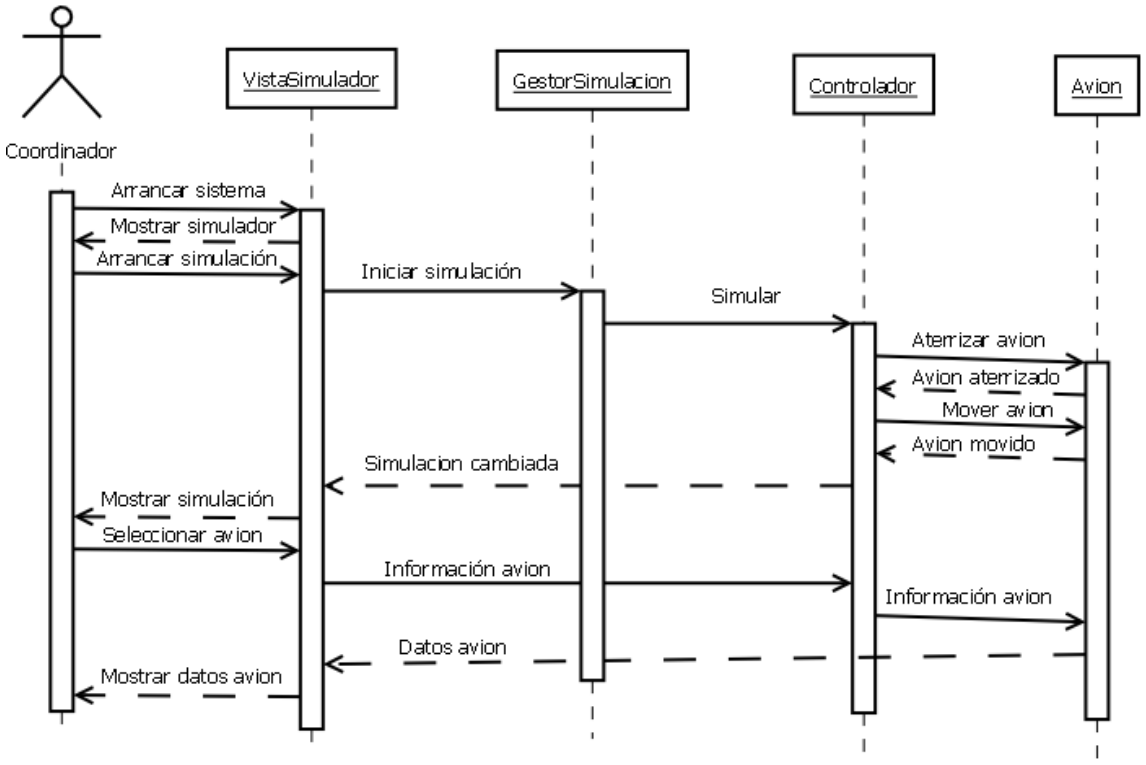


Ilustración 24: Diagrama de secuencia de visualizar información

1. El coordinador arranca el sistema y se le muestra la interfaz del simulador.

2. El coordinador arranca la simulación y el gestor de simulación da señal al controlador para que comience a dirigir el aterrizaje de los aviones, así como los movimientos de éstos por la pista para llegar a sus zonas de estacionamiento.

3. El coordinador selecciona un avión del que desea obtener información.

4. El controlador devuelve el avión a la vista, para que obtenga la información a mostrar y la enseñe al coordinador.

3.4.2. Diagrama de secuencia de asignar escalerilla

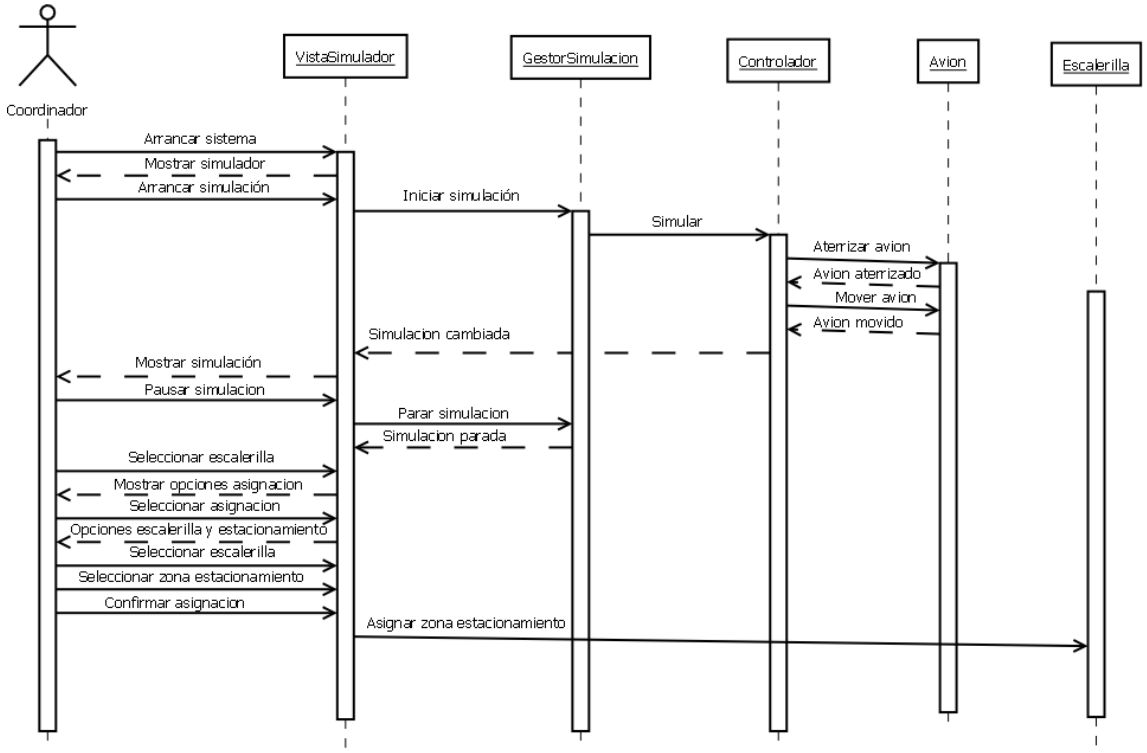


Ilustración 25: Diagrama de secuencia de asignar escalerilla

1. El coordinador arranca el sistema y se le muestra la interfaz del simulador.

2. El coordinador arranca la simulación y el gestor de simulación da señal al controlador que comience a dirigir el aterrizaje de los aviones, así como los movimientos de éstos por la pista para llegar a sus zonas de estacionamiento.

3. El coordinador pausa la simulación y el gestor de simulación no da señal al controlador de que siga con la simulación.

4. El coordinador selecciona una escalerilla, marca la opción de asignación de escalerilla e indica qué escalerilla debe ser asignada a qué zona de estacionamiento (cada zona de estacionamiento sólo puede tener asignado un avión en cada momento).

5. La asignación se encola en la lista de servicios de la escalerilla.

3.4.3. Diagrama de secuencia de cancelar escalerilla

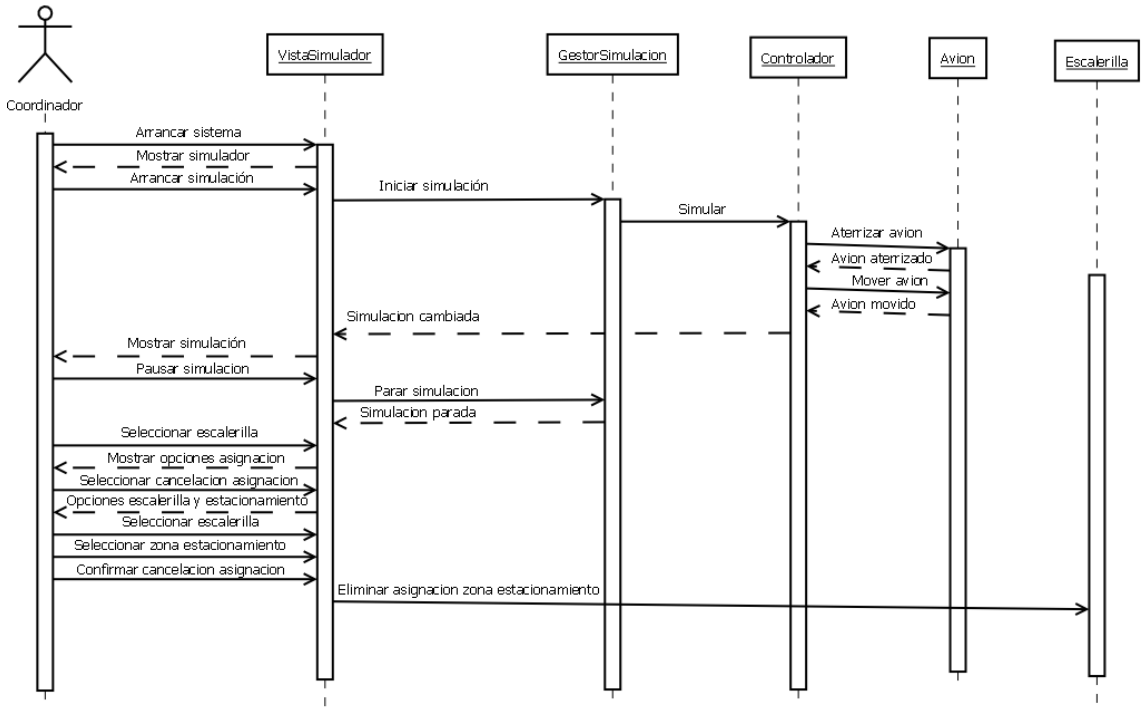


Ilustración 26: Diagrama de secuencia de cancelar escalerilla

1. El coordinador arranca el sistema y se le muestra la interfaz del simulador.

2. El coordinador arranca la simulación y el gestor de simulación da señal al controlador para que comience a dirigir el aterrizaje de los aviones, así como los movimientos de éstos por la pista para llegar a sus zonas de estacionamiento.

3. El coordinador pausa la simulación y el gestor de simulación no da señal al controlador de que siga con la simulación.

4. El coordinador selecciona una escalerilla, marca la opción de cancelación de asignación de escalerilla e indica qué asignación (indicando escalerilla y zona de estacionamiento) debe cancelarse.

3.4.4. Diagrama de secuencia de generar regla

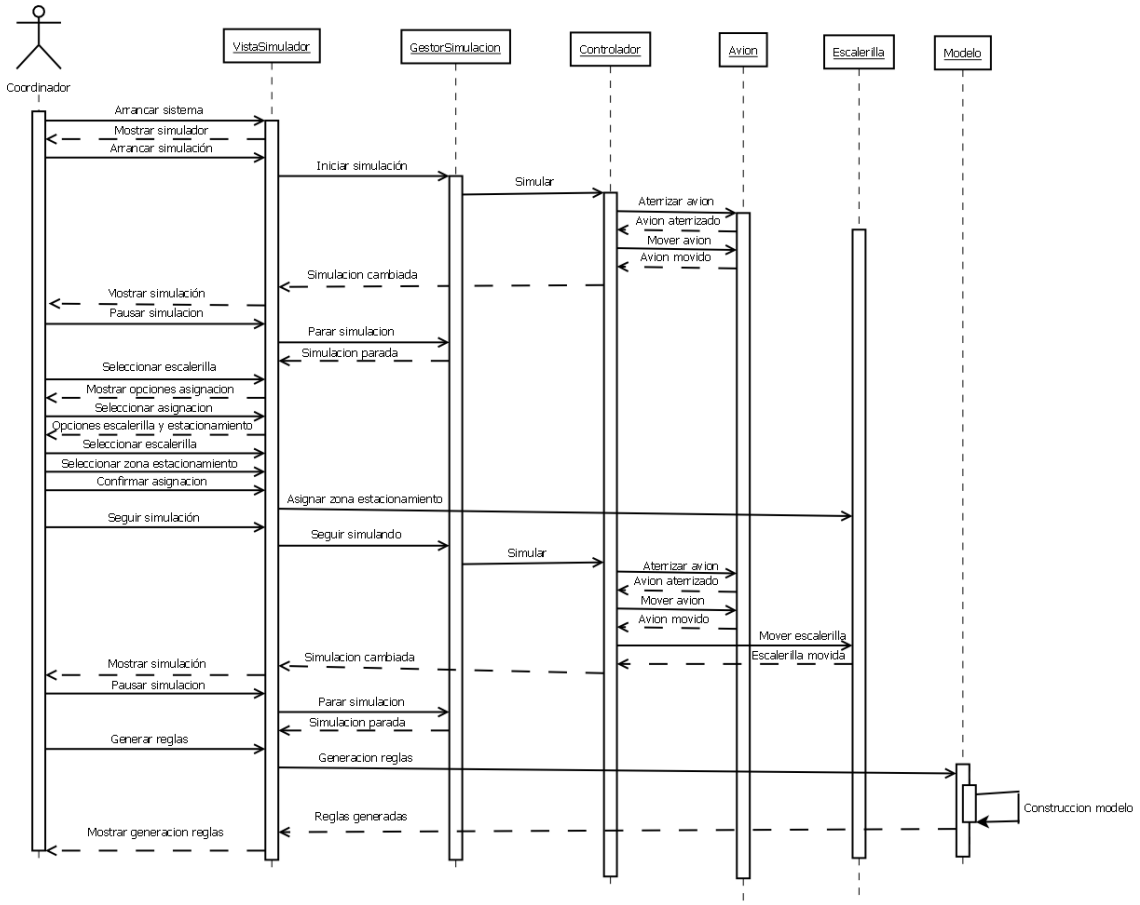


Ilustración 27: Diagrama de secuencia de generar regla

1. El coordinador arranca el sistema y se le muestra la interfaz del simulador.

2. El coordinador arranca la simulación y el gestor de simulación da señal al controlador para que comience a dirigir el aterrizaje de los aviones, así como los movimientos de éstos por la pista para llegar a sus zonas de estacionamiento.

3. El coordinador pausa la simulación y el gestor de simulación no da señal al controlador de que siga con la simulación.

4. El coordinador selecciona una escalerilla, marca la opción de asignación de escalerilla e indica qué escalerilla debe ser asignada a qué zona de estacionamiento (cada zona de estacionamiento sólo puede tener asignado un avión en cada momento).

5. La asignación se encola en la lista de servicios de la escalerilla.

6. El coordinador hace que continúe la simulación y el gestor de simulación vuelve a dar señal al controlador de que siga con la misma.

7. El coordinador pausa la simulación y el gestor de simulación vuelve a no dar señal al controlador para que siga con la simulación.

8. El coordinador pide la generación del conjunto de reglas de asignación (o cancelación de asignación) y la vista del simulador llama a la clase que crea el modelo de generación de reglas.

9. Cuando el conjunto de reglas es creado, el control se le devuelve a la vista para que indique al coordinador que las reglas ya han sido generadas.

3.4.5. Diagrama de secuencia de visualizar regla

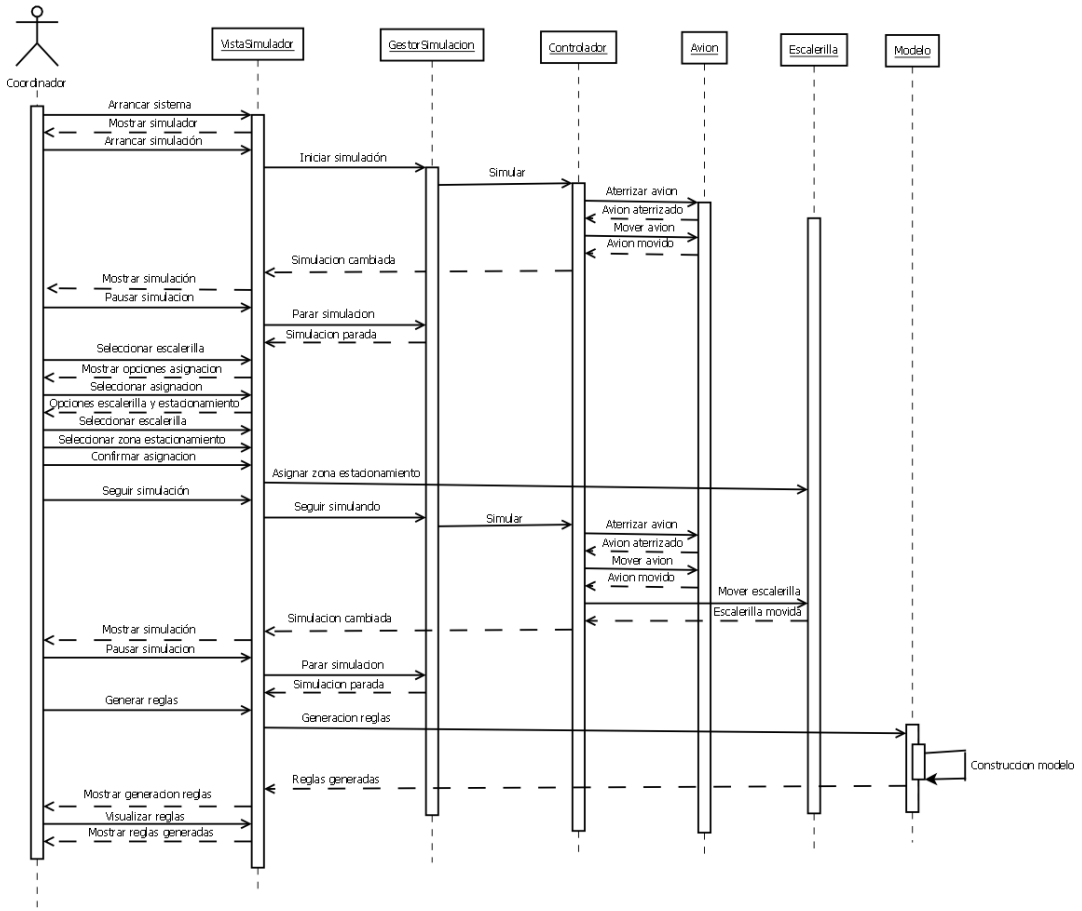


Ilustración 28: Diagrama de secuencia de visualizar regla

1. El coordinador arranca el sistema y se le muestra la interfaz del simulador.

2. El coordinador arranca la simulación y el gestor de simulación da señal al controlador para que comience a dirigir el aterrizaje de los aviones, así como los movimientos de éstos por la pista para llegar a sus zonas de estacionamiento.

3. El coordinador pausa la simulación y el gestor de simulación no da señal al controlador de que siga con la simulación.

4. El coordinador selecciona una escalerilla, marca la opción de asignación de escalerilla e indica qué escalerilla debe ser asignada a qué zona de estacionamiento (cada zona de estacionamiento sólo puede tener asignado un avión en cada momento).

5. La asignación se encola en la lista de servicios de la escalerilla.

6. El coordinador hace que continúe la simulación y el gestor de simulación vuelve a dar señal al controlador de que siga con la misma.

7. El coordinador pausa la simulación y el gestor de simulación vuelve a no dar señal al controlador para que siga con la simulación.

8. El coordinador pide la generación del conjunto de reglas de asignación (o cancelación de asignación) y la vista del simulador llama a la clase que crea el modelo de generación de reglas.

9. Cuando el conjunto de reglas se crea, el control se le devuelve a la vista para que indique al coordinador que las reglas ya han sido generadas.

10. El coordinador pide que se le muestren las reglas generadas y la vista devuelve las reglas generadas.

3.4.6. Diagrama de secuencia de cargar reglas

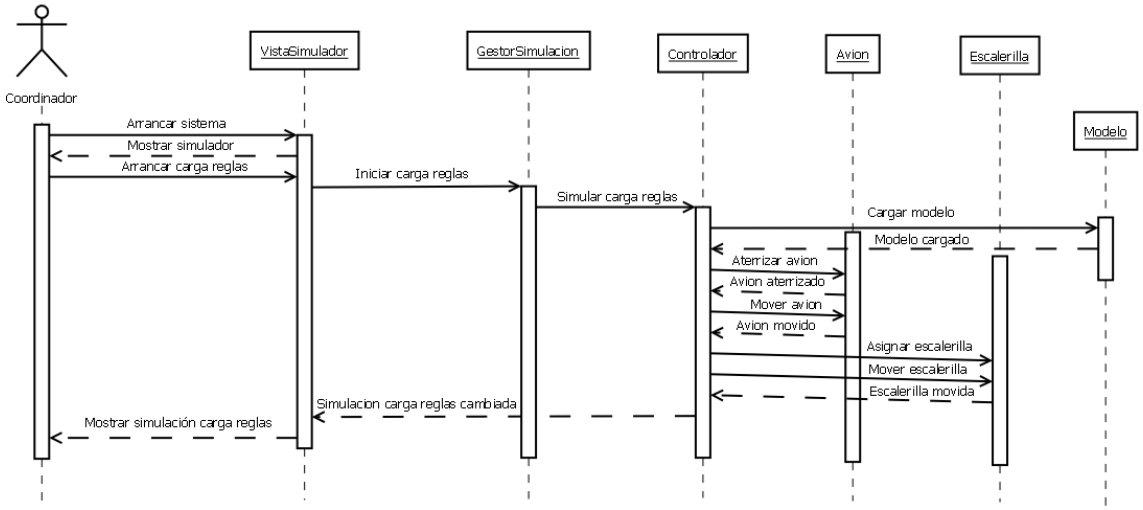


Ilustración 29: Diagrama de secuencia de cargar reglas

1. El coordinador arranca el sistema y se le muestra la interfaz del simulador.

2. El coordinador arranca la aplicación en modo de carga de datos para que se asignen de forma automática las escalerillas en función de conjunto de reglas (modelo).

3. El gestor de simulación da señal al controlador que se ha arrancado la aplicación en modo de carga de reglas.

4. El controlador carga el modelo y da comienzo con la simulación, dirigiendo el aterrizaje de los aviones, así como los movimientos de éstos por la pista para llegar a sus zonas de estacionamiento y la asignación o cancelación de la asignación de las escalerillas cuando el modelo así lo disponga.

5. El controlador devuelve a la vista la información sobre el estado del aeropuerto, para que ésta se lo vaya mostrando al coordinador a lo largo de toda la simulación.

4. Implementación

En este apartado se van a comentar algunos detalles importantes en la implementación del simulador aeroportuario, así como el preprocesamiento de los datos que ha sido necesario para construir el modelo de generación de reglas y poder obtener unos resultados lo más satisfactorios posibles. También se van a detallar algunas limitaciones que ha sido necesario adoptar en el proyecto y que repercuten en la implementación del mismo.

La aplicación de los algoritmos de minería de datos requiere la realización de una serie de actividades previas, encaminadas a preparar los datos de entrada, debido a que, en muchas ocasiones dichos datos proceden de fuentes heterogéneas, no tienen el formato adecuado o contienen ruido. Por otra parte, es necesario interpretar y evaluar los resultados obtenidos. El proceso completo consta de las siguientes etapas:

- Recopilación de datos: como el árbol de decisión J48 es una técnica de minería de datos, es fundamental primeramente recopilar un conjunto de datos para que, a partir de los mismos, podamos extraer información no trivial residente de manera implícita en los mismos.

- Preprocesamiento de los datos: transformación de los datos para que puedan ser manejados y obtener de manera más fácil información sobre los mismos.

- Fase de entrenamiento: como el árbol de decisión J48 usa un algoritmo supervisado o predictivo, es decir, predice el valor del atributo clase (etiqueta) de una instancia (conjunto de datos o atributos) a partir de los valores de otros atributos de la instancia, se le aplica un aprendizaje supervisado que consta de dos fases: entrenamiento y test. La fase de entrenamiento consiste en la construcción de un modelo (en este caso el árbol de decisión J48) usando un subconjunto de datos con etiqueta conocida.

- Fase de pruebas o test: se prueba el modelo construido comparando la clase predicha con la clase real de las instancias. Existen varias formas de realizar esta fase: usando como conjunto de datos de test el mismo que el de entrenamiento, usar otro conjunto de datos distinto al de entrenamiento, realizar validación cruzada (se divide el conjunto de datos en N partes y, por cada parte, se construye el clasificador con las N-1 partes restantes y se prueba con ésa, haciéndolo con cada una de las N

particiones) o dividir el conjunto de datos en dos grupos y con uno se entrena y con otro se prueba.

- Análisis de los resultados obtenidos: se estudian los resultados obtenidos tras la fase de pruebas o test (porcentaje de aciertos y fallos al clasificar una instancia con el modelo) para comprobar si con el modelo construido se obtienen los resultados esperados.

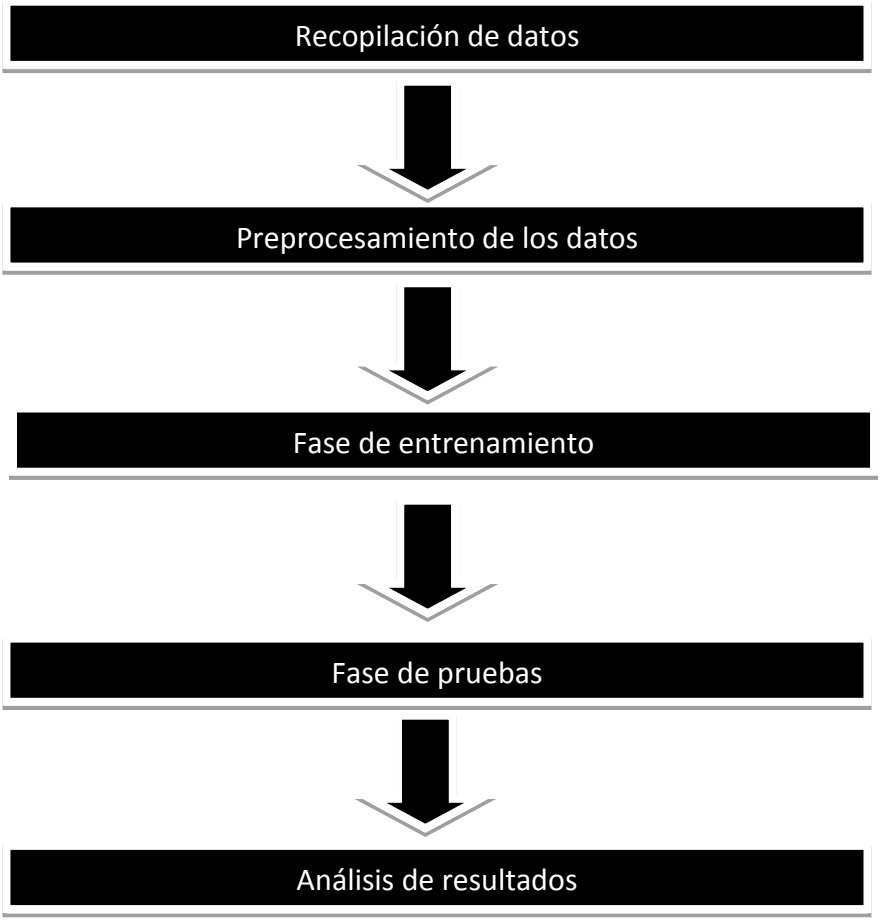


Ilustración 30: Proceso de los algoritmos de minería de datos

4.1. Elaboración de modelo de datos

La elaboración del modelo de datos consiste en identificar las estructuras de información que va a utilizar nuestro sistema. Mediante la creación del modelo de datos, se facilita el diseño posterior de la base de datos, permitiendo una identificación clara de las entidades y relaciones de las que hará uso el sistema.

Para este proyecto, la empresa Asseco Spain ha proporcionado una muestra de la base de datos que usa alguno de los sistemas existentes en el aeropuerto de Madrid-Barajas. No se expone el modelo de datos de dicha base de datos debido a la

gran complejidad que tiene (existen multitud de campos, tablas y relaciones entre tablas). Como información, en dicha base de datos está almacenada una gran cantidad de datos sobre: cintas de transporte de equipajes, posibles incidencias, clases y subclases de avión, tiempos de embarque y desembarque de pasajeros, tiempos de carreteo, situación de los hipódromos; hora de llegada, procedencia, número de pasajeros, etc. de los aviones que aterrizan en el aeropuerto; hora de salida, destino, número de pasajeros, etc. de los aviones que despegan del aeropuerto; horario del personal y datos sobre el mismo; etc.

De toda esta información, en este proyecto sólo se usará una pequeña parte de la misma ya que en el simulador aéreo sólo se tienen en cuenta los tres elementos que forman parte del entramado del servicio de asignación y cancelación de escalerillas: las propias escalerillas, los aviones y las zonas de estacionamiento. Por ello, el sistema experto se conecta con esta muestra de la base de datos para consultar tres tablas principalmente:

- T_GRUPOS_AVION_L: donde se indica el tiempo de carreteo y embarque/desembarque de pasajeros según la clase de avión y situación de la zona de estacionamiento. En total hay 7 clases de aviones diferentes en la base de datos.

- T_TIPO_AVION_L: donde se establece la relación entre las diferentes clases de avión y sus subclases. Cada subclase de avión sólo puede pertenecer a una única clase del mismo y en la base de datos hay hasta 222 subclases distintas de aviones. En la siguiente figura se muestra la relación entre esta tabla de la base de datos y la anterior.

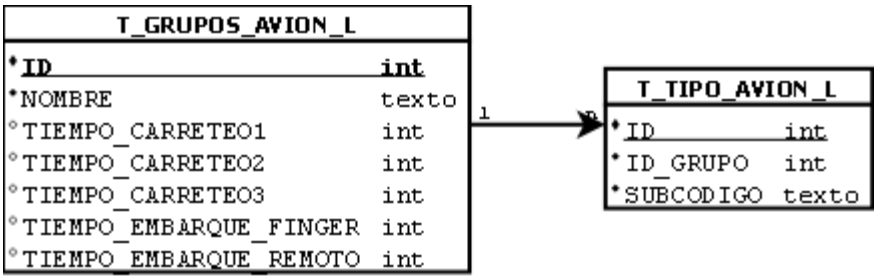


Ilustración 31: Relación grupos y tipos de avión

- T_ESCALERILLAS: tabla con el inventario de las escalerilla (cantidad y tipos de las mismas). En la base de datos hay 7 tipos de escalerillas (de 1 metro; de 1,5 metros; de 2 metros; de 2,5 metros; de 3 metros; de 3,5 metros y de 4 metros) y 20 escalerillas en total.

Ilustración 32: Tabla de escalerillas

4.2. Modelo del conocimiento

El siguiente paso, después de la elaboración del modelo de datos y antes de comenzar con la implementación del simulador, fue realizar el preprocesamiento de los datos de entrada que iba a recibir el modelo antes de la fase de entrenamiento. Este paso ha sido fundamental en este proyecto ya que, al ser un proyecto orientado a la inteligencia artificial y minería de datos, una gran parte del fracaso o éxito del proyecto se debe al hecho de realizar un buen preprocesamiento de los datos.

Lo primero que hay que tener claro para realizar el procesamiento es cuáles van a ser los datos de entrada. Todo modelo, y como tal el árbol de decisión J48, necesita de una fase de entrenamiento para poder aprender ciertos patrones que están implícitos en los datos de entrenamiento que recibe. En este proyecto, cada instancia (o conjunto de valores de variables) del modelo representa el estado del aeropuerto en un momento determinado y dicho estado debe dar la suficiente información para decidir si se debe asignar una escalerilla a una zona de estacionamiento, cancelar una asignación o no realizar ninguna acción en ese momento.

El conocimiento total del estado del aeropuerto está suministrado por la muestra de la base de datos, pero esta muestra tiene un grandísimo problema que debe ser tenido en cuenta: la gran cantidad de campos (variables) que tiene. Un número tan alto de variables provocaba que fuera a ser imposible que el modelo consiguiera aprender y generar un conjunto de reglas razonable teniendo tantísima información a tener en cuenta y con un tiempo de entrenamiento limitado. Por ello, se eligió un subconjunto de todas estas variables para que estuvieran presentes tanto en el simulador como en el conjunto de ejemplos de entrenamiento del que aprendiera el árbol de decisión J48. Para tomar la decisión de qué variables eran las más acertadas se consultó con un coordinador experto y se usó el sentido común. Las variables elegidas fueron:

- Posición de las zonas de estacionamiento

- Localización de las zonas de estacionamiento: local si la zona de estacionamiento es adyacente a una terminal del aeropuerto o remota si no lo es.

- Identificador de las escalerillas asignadas a las zonas de estacionamiento

- Posición de los aviones

- Clase de los aviones

- Subclase de los aviones

- Porcentaje de pasajeros dentro del avión

- Tiempo que le queda para despegar al avión

- Estado de los aviones (yendo a la zona de estacionamiento, en la zona de estacionamiento, sin pasajeros)

- Tiempo que llevan desembarcando los aviones

- Posición de las escalerillas

- Estado de las escalerillas (libre, yendo a zona de estacionamiento y en zona de estacionamiento)

- Tipo de escalerillas

- Identificador de la zona de estacionamiento a la que van las escalerillas

De todas estas variables había dos (porcentaje de pasajeros dentro del avión y el tiempo que llevan desembarcando los aviones) de las que el coordinador puntualizó que influían en su decisión pero en casos muy extremos, por ejemplo, en situaciones que hacían que una escalerilla que estuviera siendo usada para desembarcar pasajeros de un avión, se tomara la decisión crítica de cancelar este servicio y llevar la escalerilla a otro avión para el desembarque de sus pasajeros. Por ello, estas dos variables tampoco han sido tenidas en cuenta para la toma de decisiones. El coordinador también comentó que las cancelaciones de las asignaciones de las escalerillas eran muy poco frecuentes debido a que se gastaba un tiempo innecesario en mover la escalerilla a una zona de estacionamiento para que luego no se llevara a cabo el servicio planeado y que la situación que se debía dar para que se produjera una cancelación era la siguiente: tener una zona de estacionamiento remota a la que está

llegando un avión, que ese avión necesitara despegar pronto de nuevo y que no se tuvieran escalerillas libres compatibles con ese tipo de avión.

Una vez que ya se tenía una visión global de las variables que formaban parte de la decisión de un coordinador para la asignación o cancelación de la asignación de una escalerilla, llegaba la hora de elegir el formato con el que se iba a representar cada una de estas variables. Para ello, hay que saber que, como el árbol de decisión J48 que se va a construir se va a hacer con la ayuda de la herramienta Weka, es necesario conocer el fichero de entrada de datos de esta herramienta. Weka usa ficheros .arff como datos de entrada. Un fichero .arff ("attribute-relation file format") es un archivo de texto ASCII que describe una lista de instancias que comparten un conjunto de atributos. En este sistema, cada instancia está formada por los valores que tiene el conjunto de variables que ayudan a tomar la decisión de la asignación, más la decisión tomada por el coordinador. En cuanto a tipos de datos, Weka soporta los siguientes: numéricos (reales o enteros), nominales, cadenas de texto y fechas. A continuación se detalla por cada variable de entrada el formato y representación utilizada en principio:

VARIABLE	REPRESENTACIÓN
Posición de zonas de estacionamiento	Entero que representa su posición "x"
	Entero que representa su posición "y"
Localización de zonas de estacionamiento	Cadena de texto ("local" o "remoto")
Identificador de escalerilla asignada a zona de estacionamiento	Entero
Posición de avión	Entero que representa su posición "x"
	Entero que representa su posición "y"
Clase de avión	Cadena de texto
Subclase de avión	Cadena de texto
Tiempo queda despegar avión	Entero
Estado de avión	Cadena de texto ("yendo a parking", "en parking", "sin pasajeros")
Posición de escalerilla	Entero que representa su posición "x"
	Entero que representa su posición "y"
Estado de escalerilla	Cadena de texto ("libre", "yendo a parking", "en parking")
Tipo de escalerilla	Cadena de texto
Identificador de zona de estacionamiento a la que va escalerilla	Entero

Tabla 55: Primera representación de variables de decisión

Como se puede observar en la tabla, las posiciones de los aviones, escalerillas y zonas de estacionamiento fueron representadas con su posiciones "x" e "y" en el simulador aeroportuario.

Sin embargo, esta representación tenía dos problemas fundamentalmente:

- Los atributos “x e “y” que representan la posición de los aviones eran atributos que estaban relacionados el uno con el otro ya que estos atributos sólo tenían significado juntos.

- Las representaciones en cadena de texto no son muy aconsejables ya que el aprendizaje es más costoso

Para arreglar el primero de estos problemas se pensó que en vez de conocer la posición de aviones, escalerillas y zonas de estacionamiento mediante sus coordenadas “x” e “y”, se conociera su posición por las distancias. Por ejemplo, el hecho de que el coordinador se fijara en la posición de un avión y de la zona de estacionamiento a la que se dirige no era tanto por saber dónde estaba cada uno posicionado en el aeropuerto, sino la distancia que le quedaba al avión para llegar a la zona de estacionamiento con el fin de tener preparada la escalerilla a tiempo en esa zona de estacionamiento. Siguiendo este razonamiento, los cambios que se debían hacer en la representación para solucionar el primer problema eran:

- No tener en cuenta las posiciones “x” e “y” de los aviones

- No tener en cuenta las posiciones “x” e “y” de las zonas de estacionamiento

- No tener en cuenta las posiciones “x” e “y” de las escalerillas

- Tener en cuenta la distancia de cada avión a la zona de estacionamiento donde debe ir

- Tener en cuenta la distancia de cada escalerilla a cada una de las zonas de estacionamiento

Para solucionar parte del segundo problema lo que se hizo fue cambiar las representaciones en cadena de texto que tenían dos valores por representaciones de números enteros binarios, 0 ó 1. También, como cada clase (o tipo) de avión tiene unas subclases específicas y diferentes a la del resto de clases de avión, si se conoce la subclase del avión se conoce, por extensión, la clase a la cual pertenece. Debido a ello, el atributo o variable clase de avión no aporta ningún dato de relevancia y puede ser eliminado. En cuanto a los atributos de cadena de texto que representan los estados, también pueden ser eliminados porque la información sobre los mismos se puede obtener de otros atributos. Por ejemplo, si una escalerilla se encuentra a una distancia de 0 metros de la zona de estacionamiento asignada y el avión se encuentra a una distancia de 50 metros de la zona de estacionamiento está claro que el estado de la escalerilla sería que está en la zona de estacionamiento y el estado del avión que está yendo a la zona de estacionamiento.

Llegado a este punto, la representación usada es mostrada en la siguiente tabla.

VARIABLE	REPRESENTACIÓN
Localización de zonas de estacionamiento	Entero (0→"remoto", 1→"local")
Identificador de escalerilla asignada a zona de estacionamiento	Entero
Subclase de avión	Cadena de texto
Tiempo queda despegar avión	Entero
Tipo de escalerilla	Cadena de texto
Identificador de zona de estacionamiento a la que va escalerilla	Entero
Distancia de zona de estacionamiento a escalerilla	Entero
Distancia de zona de estacionamiento a avión	Entero

Tabla 56: Segunda representación de variables de decisión

Una vez hecho esto, había que diseñar tanto en el número de árboles de decisión J48 que se iban a generar como en la estructura de los ficheros de entrada de y salida de los mismos. Se analizaron las distintas posibilidades y se encontraron las siguientes alternativas.

4.2.1. Generación de un único árbol de decisión J48

Esta idea consistía en construir un único árbol de decisión J48 que generara reglas para asignar o cancelar la asignación de las escalerillas a las zonas de estacionamiento. Como sólo se debía generar un único árbol de decisión, con un único fichero de datos de entrada sería suficiente, y, cuando el coordinador asignara o cancelara la asignación de una escalerilla, se obtendrían los datos para generar una instancia. La instancia estaría formada por el conjunto de variables de decisión más la salida. La salida del modelo estaría formada tanto por el identificador de la zona de estacionamiento como por el identificador de la escalerilla en el caso de que fuera una asignación, o únicamente el identificador de la zona de estacionamiento en caso de que fuera una cancelación, ya que cada zona de estacionamiento sólo puede tener una escalerilla asignada en un instante de tiempo.

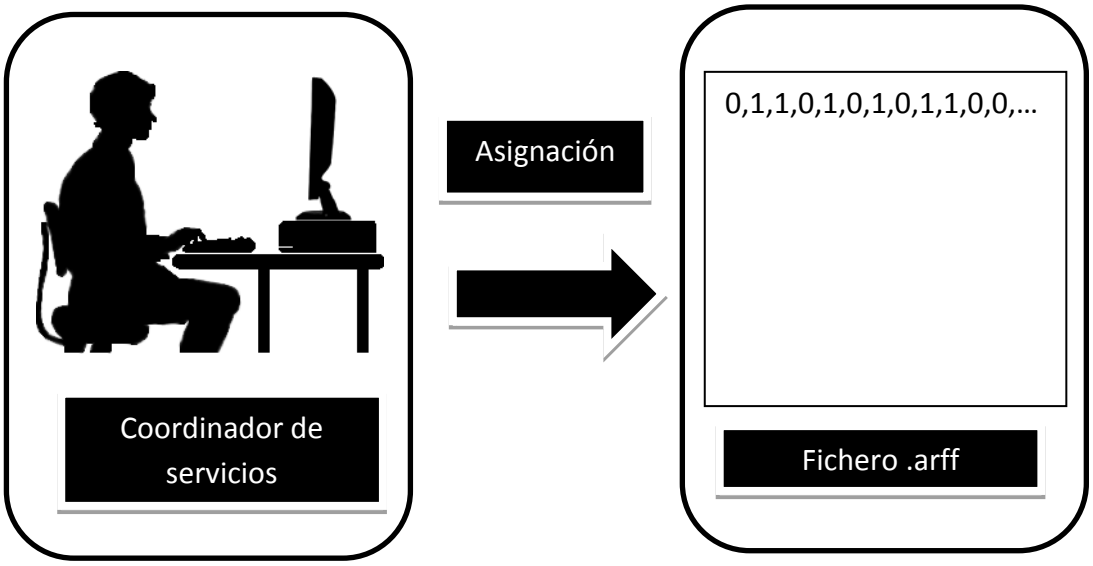


Ilustración 33: Creación del fichero ARFF con un único árbol de decisión J48

El gran inconveniente de esto era que cada instancia estaba formada por una gran cantidad de atributos y, por tanto, esto dificultaba mucho la fase de aprendizaje del modelo (una regla no escrita de la minería de datos para el aprendizaje es que es necesario como mínimo tantas instancias como 100 veces el número de atributos de una de ellas).

La representación de cada instancia en el modelo sería como sigue.

VARIABLE	REPRESENTACIÓN
Localización de zonas de estacionamiento (∀ zona de estacionamiento)	Entero (0→"remoto", 1→"local")
Identificador de escalerilla asignada a zona de estacionamiento (∀ zona de estacionamiento)	Entero (-1→"no hay escalerilla asignada, id→identificador de escalerilla asignada)
Subclase de avión (∀ zona de estacionamiento)	Cadena de texto
Tiempo queda despegar avión (∀ zona de estacionamiento)	Entero
Tipo de escalerilla (∀ escalerilla)	Cadena de texto
Identificador de zona de estacionamiento a la que va escalerilla (∀ escalerilla)	Entero
Distancia de zona de estacionamiento a escalerilla (∀ escalerilla y ∀ zona de estacionamiento)	Entero
Distancia de zona de estacionamiento a avión (∀ zona de estacionamiento)	Entero
Identificador de zona de estacionamiento e identificador de escalerilla (salida)	Cadena de texto (idZonaEstacionamiento_idEscalerilla)

Tabla 57: Representación de instancia con un único árbol de decisión J48

El número de atributos por instancia con esta representación es el siguiente.

5 x Nº zonas estacionamiento + 2 x Nº escalerillas + Nº zonas estacionamiento x Nº escalerillas + 1

4.2.2. Generación de un único árbol de decisión J48 con mejora

Debido a que el gran problema del anterior modelo era que no se tenían datos suficientes, lo que se pensó es que, en vez de que se generara una instancia cada vez que el coordinador realizara una asignación o cancelación, se generaran N instancias cada cierto tiempo (hubiera o no hubiera realizado el coordinador ninguna asignación o cancelación), siendo N el número de zonas de estacionamiento que hubiera en el aeropuerto simulado. Para poder hacer esto cada instancia estaría formada por el conjunto de variables que influyen en la decisión, más el identificador de la zona de estacionamiento, más la salida (que en este caso tomaría los valores del identificador de la escalerilla asignada, -1 en caso de que no se hubiera realizado ninguna asignación ni cancelación y -2 en caso de que fuera una cancelación). Por tanto, en cada ciclo se formarían N instancias donde lo único que varía de unas instancias a otras dentro de un ciclo es el identificador de la zona de estacionamiento y la salida.

Además, como las escalerillas sólo son usadas en las zonas de estacionamiento remotas, no es necesario que las instancias contengan información sobre las zonas de estacionamiento locales y, gracias a ello, el número de atributos de las instancias se ve reducido considerablemente.

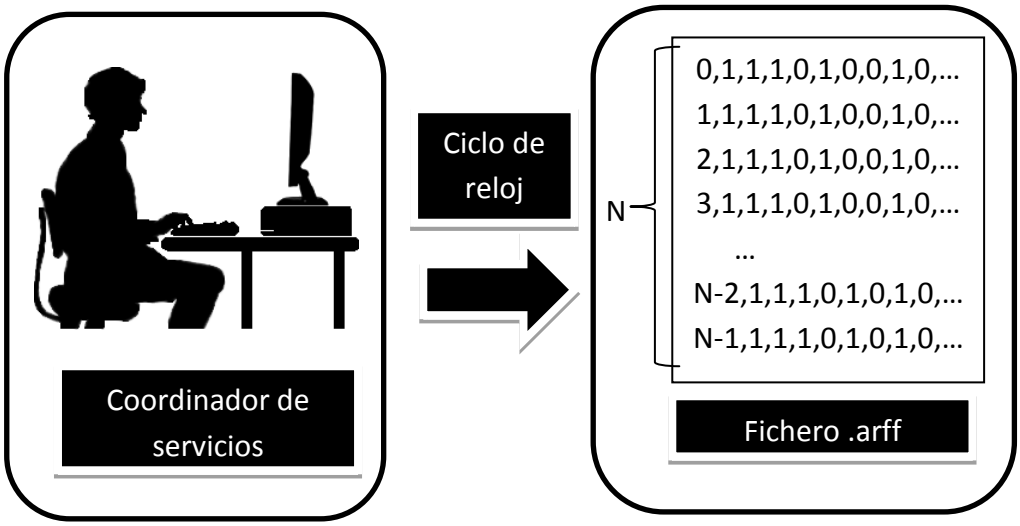


Ilustración 34: Creación del fichero ARFF con un único árbol de decisión J48 con mejora

La representación de cada instancia quedaría así.

VARIABLE	REPRESENTACIÓN
Identificador de zona de estacionamiento	Entero
Identificador de escalerilla asignada a zona de estacionamiento (∀ zona de estacionamiento remota)	Entero (-1→"no hay escalerilla asignada, id→identificador de escalerilla asignada)
Subclase de avión (∀ zona de estacionamiento remota)	Cadena de texto
Tiempo queda despegar avión (∀ zona de estacionamiento remota)	Entero
Tipo de escalerilla (∀ escalerilla)	Cadena de texto
Identificador de zona de estacionamiento a la que va escalerilla (∀ escalerilla)	Entero
Distancia de zona de estacionamiento a escalerilla (∀ escalerilla y ∀ zona de estacionamiento remota)	Entero
Distancia de zona de estacionamiento a avión (∀ zona de estacionamiento remota)	Entero
Identificador de escalerilla (salida)	Entero (-2→"cancelación", -1→"no se ha tomado decisión alguna", id→identificador de escalerilla asignada)

Tabla 58: Representación de instancia con un único árbol de decisión J48 con mejora

El número de atributos por instancia con esta representación es el siguiente.

1 + 4 x Nº zonas estacionamiento remotas + 2 x Nº escalerillas + Nº zonas estacionamiento remoto x Nº escalerillas + 1 =
2 + 4 x Nº zonas estacionamiento remotas + 2 x Nº escalerillas + Nº zonas estacionamiento remoto x Nº escalerillas

El problema de este modelo es que está muy supeditado a que éste aprenda que el atributo que representa al identificador de la zona de estacionamiento es fundamental para la asignación, ya que, si no se conoce la zona de estacionamiento a la que se le va a realizar la asignación o cancelación de una escalerilla, es imposible realizar tal acción. Una solución para este problema sería construir dos árboles de decisión J48 en los que uno tuviera como entrada el estado del aeropuerto (variables de decisión) y como salida la zona de estacionamiento; y esta salida junto con el estado del aeropuerto formarían la entrada del otro árbol de decisión J48. La salida del mismo sería el identificador de la escalerilla, -1 en caso de no tener que hacer nada o -2 en caso de que sea una cancelación.

4.2.3. Generación de N árboles de decisión J48

Esta idea consistía en construir tantos árboles de decisión J48 como zonas de estacionamiento hubiera en el aeropuerto simulado. Cada árbol de decisión J48

representaría las reglas para asignar o cancelar la asignación de las escalerillas a una zona de estacionamiento específica. Como se debían generar N árboles de decisión, se necesitarían N ficheros de datos de entrada, uno por árbol de decisión. Las instancias de cada fichero de una zona de estacionamiento estarían formadas por las variables que influyen en la decisión de asignar una escalerilla a esa zona de estacionamiento, las variables que influyen en la cancelación de asignaciones de escalerilla, más la salida (que seguiría siendo el identificador de la escalerilla, -1 si no se tuviera que hacer nada o -2 en caso de que fuera una cancelación).

Esta nueva idea tiene una grandísima ventaja que es la reducción del número de atributos por instancia, ya que, para generar el conjunto de reglas de la zona de estacionamiento 1, las instancias de este modelo sólo necesitan contemplar las variables que ayudan a tomar la decisión sobre la asignación o cancelación de escalerillas en esa zona de estacionamiento y no en todas.

Además, al centrarse cada modelo en una única zona de estacionamiento, ya no tiene sentido tener atributos de los identificadores de la escalerilla y de los identificadores y localizaciones de la zonas de estacionamiento. Estos atributos pueden ser sustituidos por otros que indiquen si la zona de estacionamiento y la escalerilla están asignadas o no. En cuanto a la creación de instancias, se crearán en cada ciclo tantas instancias como zonas de estacionamiento hubiera (una instancia por modelo).

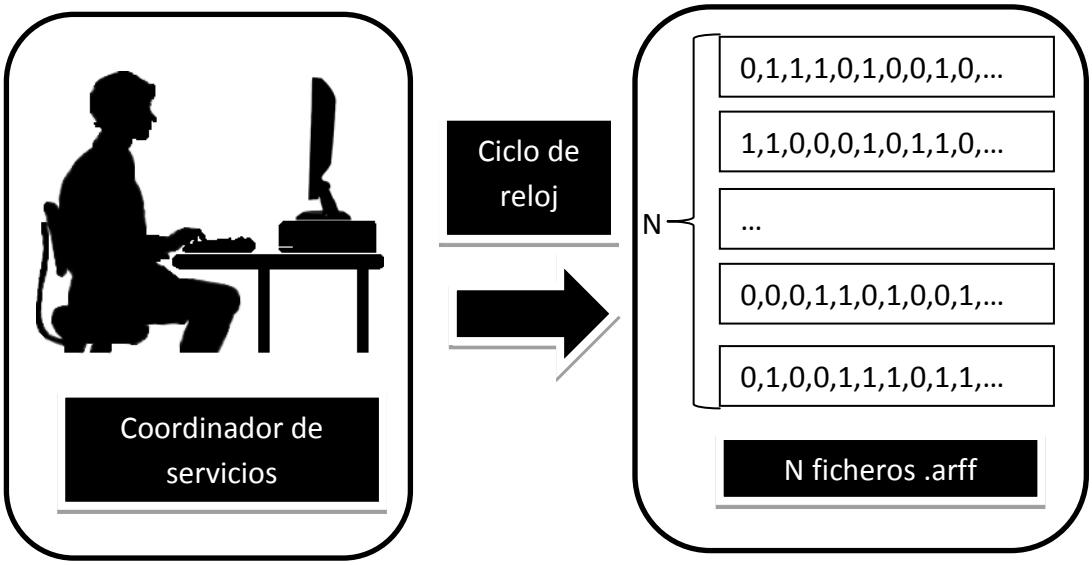


Ilustración 35: Creación de ficheros ARFF en la generación de N árboles de decisión J48

Las desventajas de este modelo es que es necesario un mayor tiempo de procesamiento porque el número de árboles de decisión que se deben construir es mayor y una mayor complejidad de gestión en caso de que se tuviera una gran cantidad de zonas de estacionamiento y, por tanto, de árboles de decisión.

A pesar de estas desventajas y gracias a las grandes ventajas que aportaba con respecto al resto de ideas que han sido explicadas anteriormente, se decidió que, para el aprendizaje del sistema experto aeroportuario, se usara esta última solución, en la que se entrenan tantos árboles de decisión como zonas de estacionamiento haya en el simulador, y cada árbol de decisión representa el conjunto de reglas de asignación y cancelación de escalerillas para esa zona de estacionamiento.

A continuación se expone el nombre de las variables en los árboles de decisión, su descripción y su representación finales.

Instancia para la zona de estacionamiento ZZZ

VARIABLE	DESCRIPCIÓN	REPRESENTACIÓN
asigParkXXX (∀ zona de estacionamiento remota)	Indica si la zona de estacionamiento con identificador XXX tiene asignada una escalerilla	Entero (0→"no asignada escalerilla", 1→"asignada escalerilla")
subclaseAvion (∀ zona de estacionamiento remota)	Indica la subclase del avión que se dirige a la zona de estacionamiento ZZZ	Cadena de texto
distAvionPark	Indica la distancia del avión a la zona de estacionamiento ZZZ a la que se dirige	Entero
tiempoSalirAvionParkXXX (∀ zona de estacionamiento remota)	Indica el tiempo que le queda para despegar sin retraso al avión de la zona de estacionamiento con identificador XXX	Entero
tipoEscXXX (∀ escalerilla)	Indica el tipo de la escalerilla con identificador XXX	Cadena de texto
asignadaEscXXX (∀ escalerilla)	Indica si la escalerilla con identificador XXX ha sido asignada	Entero (0→"no asignada escalerilla", 1→"asignada escalerilla")
distParkEscXXX (∀ escalerilla)	Indica la distancia de la escalerilla XXX con la zona de estacionamiento ZZZ	Entero
decision (salida)	Indica la decisión tomada en cuanto a la asignación	Entero (-2→"cancelación", -1→"no se ha tomado decisión alguna", idEsc→"identificador de la escalerilla asignada")

Tabla 59: Representación de instancia en la generación de N árboles de decisión J48

Por tanto, el número de atributos por instancia con la representación finalmente elegida es de:

2 x Nº zonas estacionamiento remotas + 1 + Nº zonas estacionamiento remotas + Nº escalerillas x 3 + 1 =
3 x Nº zonas estacionamiento remotas + 3 x Nº escalerillas + 2

Al realizar las pruebas se detectó un pequeño problema en la generación del conjunto de ejemplos de entrenamiento: debido a que en cada ciclo se generaba una instancia por modelo y en pocos ciclos el coordinador realizaba una asignación, el número de instancias cuya salida era -1 (no se hace nada) era muchísimo mayor que el resto de instancias, por lo que el conjunto de reglas generado tendía siempre a no asignar ni cancelar asignaciones de escalerillas. Para arreglar este problema de falta de datos en la asignación de escalerillas, lo que se hizo es repetir cada instancia en la que se asigna o se cancela una asignación N veces, eligiendo en este proyecto como valor N el triple que el número de zonas de estacionamiento hubiera en el simulador.

4.3. Análisis de clases

En este apartado se explica de forma concisa el diagrama de clases obtenido, describiendo qué elementos lo forman, su papel en dicho diagrama, así como cualquier dato relevante para el análisis. El diagrama de clases del sistema es el siguiente.

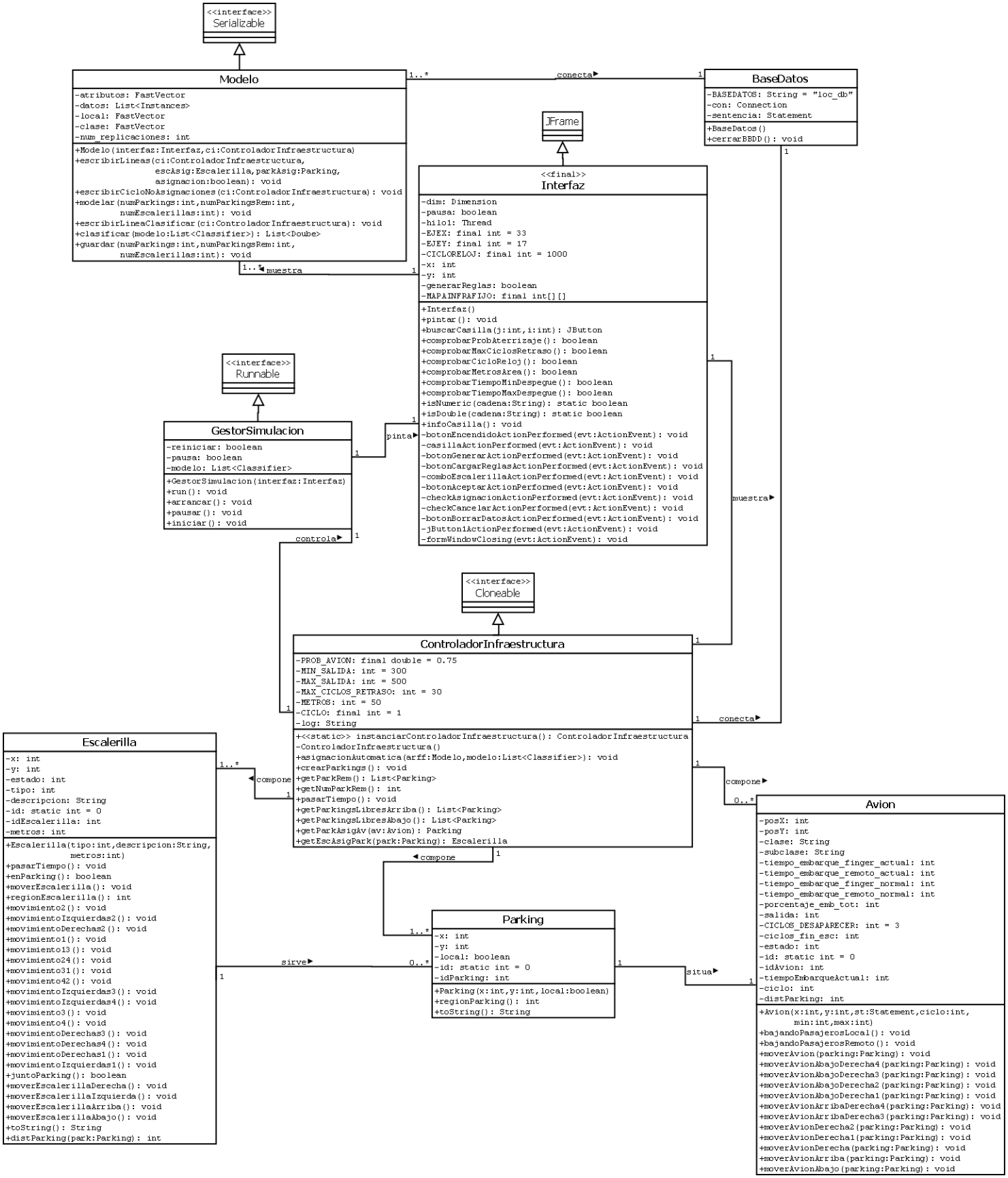


Ilustración 36: Diagrama de clases

- Modelo: clase donde se almacena la información del aeropuerto a cada instante, así como las asignaciones y cancelaciones que se van produciendo con el paso del tiempo. También es la encargada tanto de construir el árbol de decisión J48 (conjunto de reglas de asignación y cancelación de escalerillas), como de indicar cuándo hay que hacer una asignación o cancelación en el caso de que se esté en el modo de asignación automática de escalerillas.

- BaseDatos: clase encargada de realizar la conexión con la base de datos que contiene información sobre la clase y subclases de aviones, tipo y número de escalerillas, etc.

- Interfaz: como su propio nombre indica es la clase que representa la interfaz de la aplicación (botones, el simulador aeroportuario, menús de la aplicación, etc.).

- GestorSimulacion: clase que se encarga de la sincronización entre la interfaz propia de la aplicación y el controlador de la infraestructura del simulador (ControladorInfraestructura). Esta clase indica al controlador cuando ha pasado un instante de tiempo y justo después solicita que se pinte la interfaz para que el coordinador pueda ver los cambios ocurridos.

- ControladorInfraestructura: clase que controla los elementos móviles del aeropuerto simulado. Por tanto, controla a los aviones (llegada de aviones al aeropuerto simulado y cuándo se deben mover los aviones y desembarcar a los pasajeros) y a las escalerillas (cuándo deben moverse).

- Escalerilla: clase que representa cada una de las escalerillas del aeropuerto simulado y encargada del movimiento de este tipo de recursos en la simulación.

- Avion: clase que representa cada uno de los aviones del aeropuerto simulado y encargada tanto del movimiento de estos aparatos como el desembarque de los pasajeros.

- Parking: clase que representa la zona de estacionamiento donde los aviones se sitúan cuando aterrizan en el aeropuerto simulado.

4.4. Implementación del simulador

4.4.1. Pasos de implementación

Respecto a la implementación en Java del simulador, el proceso que se siguió para ir desarrollando las diferentes clases que lo componen y que ya han sido presentadas en el apartado 4.3 "Análisis de clases" fueron:

- Importación de la librería de Weka para poder usar los métodos necesarios de esta herramienta desde el propio simulador.

- Conexión del simulador con la base de datos .mdb (base de datos en Microsoft Access) para obtener información sobre los tipos de escalerillas, clases y subclases de aviones, etc. Es la clase de Java llamada BaseDatos.

- Creación de la interfaz del simulador (Interfaz.java) con las diferentes secciones que lo componen: la sección de los controles donde se sitúan los botones de arranque y parada del simulador, de generación y carga de reglas y el de borrado de toda la información almacenada durante la simulación; la de detalles donde se describen las características de los elementos que se seleccionan de la simulación; la sección de asignaciones de escalerillas donde el coordinador realiza tanto la asignación de las escalerillas como la cancelación de las mismas; la sección de log donde se detallan eventos importantes ocurridos en la simulación; y finalmente la sección del aeropuerto, donde se visualiza el aeropuerto y todos los elementos que lo componen, así como el movimiento de los mismos según pasa el tiempo (aterrizaje y despegue de los aviones, movimiento de las escalerillas, etc.). La interfaz lanza un hilo (thread) para que la parte de la interfaz y la parte lógica del sistema se ejecuten de forma independiente.

- Creación de una clase (llamada GestorSimulacion) encargada de llevar el ciclo de reloj de la simulación y de indicar a la interfaz que después de cada ciclo de reloj se pinte la misma con los cambios que se han producido en la simulación.

- Creación de la clase que representa a un avión (Avion.java) del aeropuerto, así como los métodos que hacen que éste se mueva por el aeropuerto simulado y la simulación del desembarque de los pasajeros. Uno de los atributos de la clase del avión indica el estado en el que se encuentra este avión: si este atributo tiene como valor -1 significa que el avión ha sufrido un gran retraso, si es 0 significa que está yendo a la zona de estacionamiento, si es 1 que está situado en la zona de estacionamiento y si

es 2 que los pasajeros del mismo están desembarcando. El aterrizaje en el simulador de una subclase de avión u otra es totalmente aleatorio.

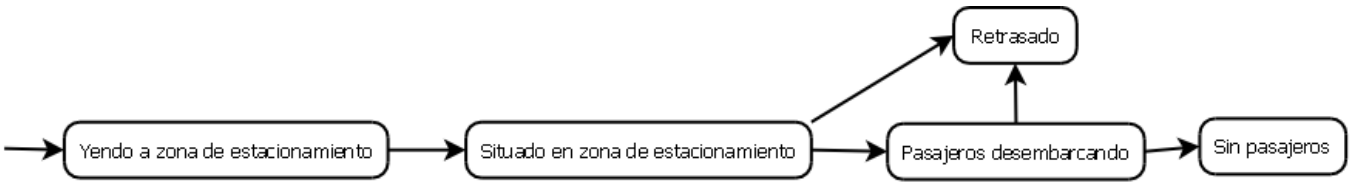


Ilustración 37: Diagrama de estados del avión

- Creación de la clase que representa a una escalerilla del aeropuerto (Escalerilla.java), así como los métodos que hacen que ésta se mueva por la pista de material del aeropuerto y que calculen la distancia de la escalerilla a una zona de estacionamiento. Esta clase tiene una lista con las diferentes zonas de estacionamiento a las que la escalerilla ha sido asignada, para que así sea posible poder realizar múltiples asignaciones a una escalerilla y que vaya realizando el servicio a cada uno de los aviones según el orden de la asignación. Uno de los atributos de la clase del avión indica el estado en el que se encuentra esta escalerilla: si este atributo tiene como valor 0 significa que la escalerilla está libre, si es 1 significa que está yendo a la zona de estacionamiento del avión asignado y si es 2 que está situado en la zona de estacionamiento del avión asignado.

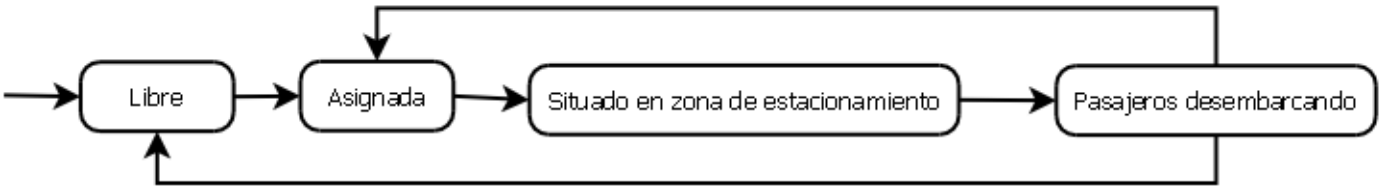


Ilustración 38: Diagrama de estados de la escalerilla

- Creación de la clase que representa a una zona de estacionamiento o parking del aeropuerto (Parking.java). Esta clase tiene un atributo en el que se referencia al avión que se dirige a esa zona de estacionamiento y otro atributo que indica si la zona de estacionamiento representada es local (adyacente a una terminal) o remota.

- Creación de la clase que controla toda la infraestructura (aviones, escalerillas y zonas de estacionamiento) del aeropuerto simulado (ControladorInfraestructura.java). Esta clase contiene listas para referenciar a todos los aviones, escalerillas y zonas de estacionamiento del aeropuerto y ha sido diseñada con un patrón singleton. Dicha clase, cuando recibe por parte del gestor de la simulación una señal de que ha pasado un ciclo de reloj, se encarga de distribuir esta información a los diferentes elementos de la infraestructura para que hagan lo que corresponda (se muevan,

realicen una determinada acción, etc.) y de la posterior coordinación de los sucesos que ocurran en la simulación. Esta clase también se encarga de decidir cuándo un avión aterriza en el aeropuerto (para ello debe haber alguna zona de estacionamiento libre y la asignación de un avión a una zona de estacionamiento es totalmente aleatoria).

- Creación de la clase encargada de manejar los métodos proporcionados por la librería de Weka importada (Modelo.java). Esta clase se encarga tanto del diseño de las instancias como de su posterior construcción cuando ha pasado un ciclo de reloj o cuando el coordinador de servicios ha realizado una asignación o cancelación. Además, esta clase es la encargada de construir el árbol de decisión J48 con el conjunto de reglas generadas; así como de guardar las instancias, el modelo y reglas generadas en ficheros.

- Cambios en el controlador infraestructura y en la clase que maneja la librería de Weka para incluir la funcionalidad de la asignación automática a partir del conjunto de reglas. El controlador de la infraestructura se encarga de ir asignando o cancelando asignaciones de las escalerillas según la salida dada por el modelo y que es manejado por la otra clase.

- Cambios en la interfaz para que se incluya tanto un apartado donde el coordinador de servicios aeroportuarios pueda cambiar algunos parámetros de la simulación (tiempos máximos y mínimos de despegue del avión, duración del ciclo de reloj, máximo retraso permitido en el despegue de un avión, probabilidad de aterrizaje de un avión en el aeropuerto y distancia entre las áreas del aeropuerto) como otro apartado donde esté la ayuda del sistema (significado de los diferentes iconos del simulador).

4.4.2. Descripción de la interfaz

Tras finalizar el proceso de implementación, la interfaz construida tiene el siguiente aspecto.

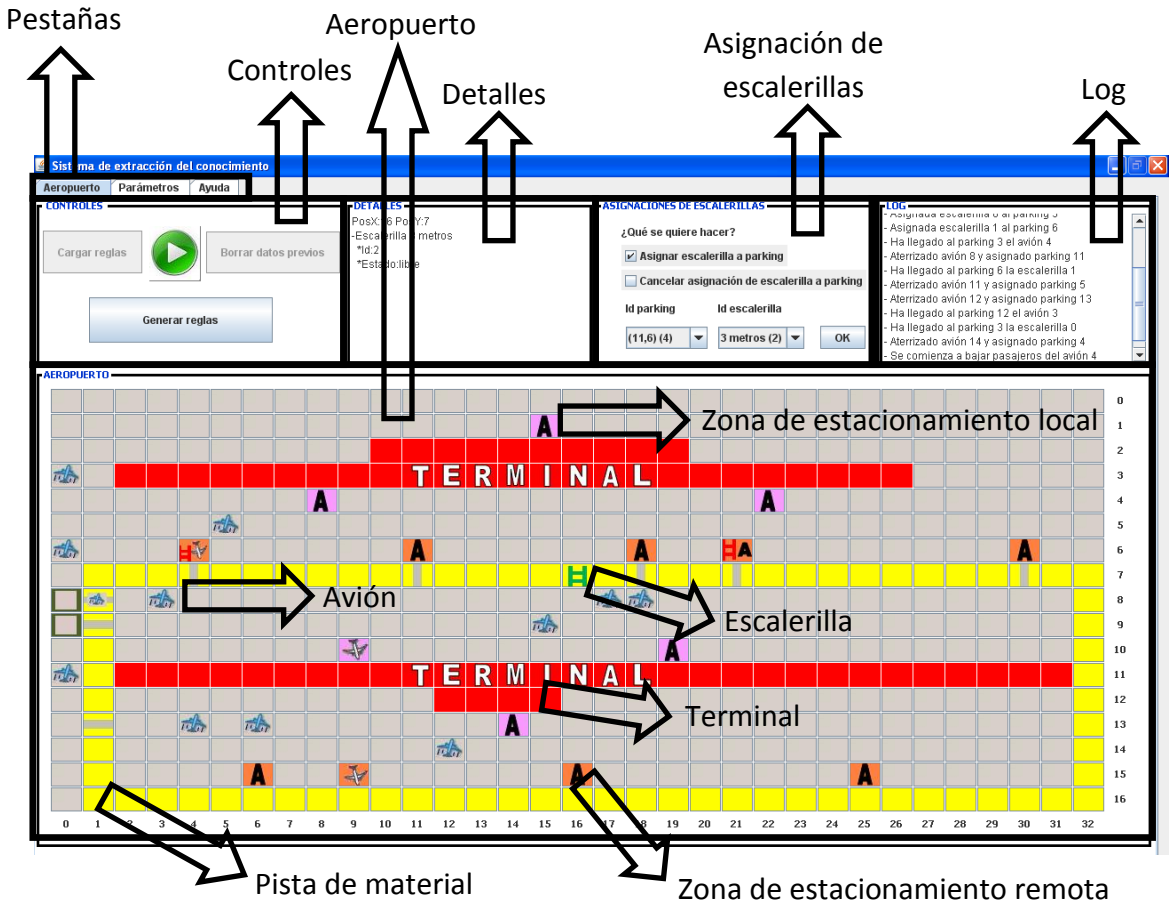


Ilustración 39: Descripción de la interfaz gráfica

Como se puede ver, la interfaz está compuesta de una serie de pestañas donde se pueden configurar algunos parámetros de la simulación, consultar la ayuda sobre los iconos que aparecen en la simulación y visualizar la simulación en cuestión. En esta última pestaña (llamada "Aeropuerto") es donde se permite al coordinador de los servicios aeroportuarios arrancar la simulación en uno de los dos modos posibles (asignación y cancelación manual o automática), ver detalles de los elementos del aeropuerto simulado (aviones, escalerillas y zonas de estacionamiento), poder asignar y cancelar asignaciones de escalerillas, ver eventos importantes ocurridos durante la simulación (log) y, por supuesto, tener una perspectiva aérea de la simulación del aeropuerto.

En dicha perspectiva aérea se pueden encontrar:

- Zonas de estacionamiento: existen de dos tipos (las locales de color rosa y las remotas de color naranja). Sólo se permite asignar escalerillas a los aviones a los que se les ha asignado una zona de estacionamiento remota.

- Aviones: que pueden estar aterrizando, moviéndose a la zona de estacionamiento asignada, situado en la zona de estacionamiento,

desembarcando a los pasajeros o, ya sin pasajeros, desapareciendo de la simulación.

- Escalerillas: el color de las escalerillas indica el estado de las mismas (verde si está libre, rojo si está asignada a algún avión, o negra si hay varias escalerillas en una misma zona del aeropuerto). Las escalerillas se mueven a lo largo de la pista de material y se encuentran paradas si no tienen asignado ningún servicio.

- Pista de material: lugar por donde se mueven las escalerillas para dirigirse de unas zonas de estacionamiento a otras. Está representada en la simulación con un color amarillo.

- Terminales: edificios, representados con un color rojo en la simulación.

Para conocer con más detalle cómo configurar el simulador aeroportuario, así como el funcionamiento del mismo, ir a la sección 8.1 “Manual de configuración del simulador aeroportuario” y 8.2 “Manual de usuario”.

5. Validación y pruebas

Para comprobar, no sólo el correcto funcionamiento del simulador sino también que el sistema experto aprende y que es capaz de generar un conjunto de reglas acorde al razonamiento llevado a cabo en la asignación y cancelación de escalerillas durante la simulación, se han realizado una serie pruebas.

Debido a que el aprendizaje del sistema lleva un tiempo considerable, ya que varias variables han de ser tenidas en cuenta y, por tanto, necesita de un conjunto de entrenamiento lo suficientemente grande para poder generar reglas lógicas, se ha tomado la decisión de realizar 6 pruebas, de las cuáles 5 han sido realizadas por la persona encargada de desarrollar este sistema y una ha sido realizada por una persona con mucha más experiencia en el sector aeroportuario y que conoce más profundamente los motivos de las asignaciones y cancelaciones de las escalerillas.

En todas las pruebas para validar el modelo generado (un árbol de decisión que representa el conjunto de reglas de asignación y cancelación de escalerillas) se ha usado la técnica de validación cruzada. Dicha técnica consiste en dividir el conjunto de datos en n partes y, por cada parte, se construye el clasificador (árbol de decisión) con las n−1 partes restantes y se prueba con ésa, siendo así por cada una de las n partes. Para todas las pruebas realizadas en este proyecto se ha usado como valor n=10.

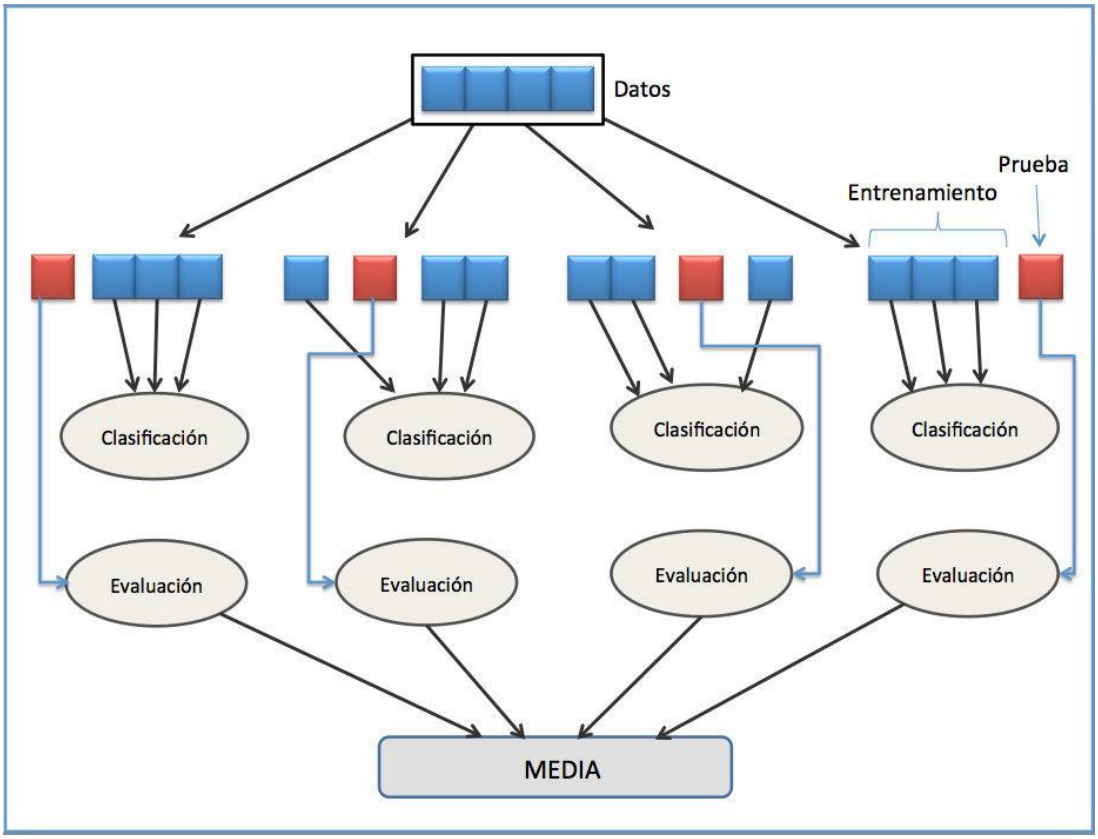


Ilustración 40: Técnica de validación cruzada

Además de esta técnica de validación, tras la generación del conjunto de reglas, en cada una de las pruebas se ha arrancado el sistema en modo de asignación automática para que el simulador asignara o cancelara las escalerillas del aeropuerto en función de dicho conjunto de reglas y así ver visualmente lo que ocurre en el aeropuerto y si concuerda con el razonamiento llevado a cabo en el aprendizaje.

Todas las pruebas han sido realizadas con un aeropuerto simulado con 2 terminales y 3 zonas de estacionamiento locales por terminal, 9 zonas de estacionamiento remotas, 3 escalerillas (una de 1 metro con identificador 0, otra de 2 metros con identificador 1 y otra de 3 metros con identificador 2), tiempo mínimo de despegue de aviones en 300 minutos y máximo en 500 minutos, un máximo de 2 aviones aterrizando al mismo tiempo y la probabilidad de que aterrice al menos un avión es del 75% y una distancia entre áreas del aeropuerto de 50 metros (cada cuadrado del aeropuerto simulado son 50 metros en realidad).

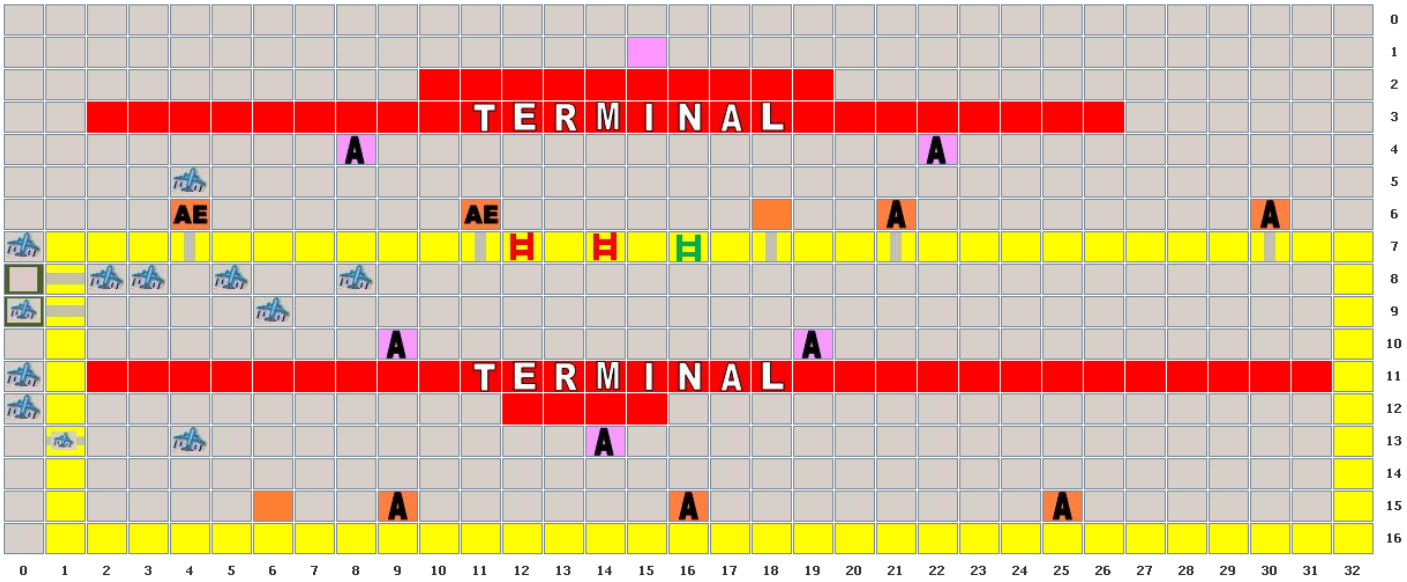


Ilustración 41: Aeropuerto simulado para las pruebas

A continuación se van a detallar un poco más cada una de las pruebas y los resultados de las mismas. Es necesario mencionar antes de continuar que lo que se va a describir ahora son unas líneas muy generales del mecanismo de asignación y cancelación de escalerillas para cada una de las pruebas y que no se profundiza en el razonamiento llevado a cabo para tomar la decisión, ya que la complejidad de dicho razonamiento es muy elevada y es imposible explicarla brevemente.

5.1. Descripción de pruebas

5.1.1. Prueba 1

En esta prueba el desarrollador del sistema ha simulado que es un coordinador de servicios aeroportuarios. El razonamiento seguido en esta prueba para la asignación y cancelación de escalerillas es que, siempre que aterrice un avión en el aeropuerto, al instante se le asigne a su zona de estacionamiento correspondiente una de las escalerillas (en caso de que la zona de estacionamiento sea remota): si la escalerilla de 1 metro está libre se le asigna ésta; si no está libre la de 1 metro pero está libre la de 2 metros, se le asigna esta segunda; si no está libre ni la de 1 metro ni la de 2 pero la de 3 metros está libre, se le asigna la de 3 metros; y, si ninguna está libre, se le asigna por defecto la de 1 metro.

5.1.2. Prueba 2

En esta prueba el desarrollador del sistema ha simulado que es un coordinador de servicios aeroportuarios. El razonamiento seguido en esta prueba para la asignación y cancelación de escalerillas es que sólo se le asigna a un avión una escalerilla cuando le quedan 200 minutos o menos para despegar y la zona de estacionamiento asociada a este avión es remota. Para elegir la escalerilla que se le va a asignar se sigue el mismo razonamiento que antes: si la escalerilla de 1 metro está libre se le asigna ésta; si no está libre la de 1 metro pero está libre la de 2 metros, se le asigna esta segunda; si no está libre ni la de 1 metro ni la de 2 pero la de 3 metros está libre, se le asigna la de 3 metros; y, si ninguna está libre, se le asigna por defecto la de 1 metro.

5.1.3. Prueba 3

En esta prueba el desarrollador del sistema ha simulado que es un coordinador de servicios aeroportuarios. El razonamiento seguido en esta prueba para la asignación y cancelación de escalerillas es que a un avión se le asigna una escalerilla u otra dependiendo de la situación de la zona de estacionamiento a la que se dirija. Para ello el aeropuerto simulado se ha dividido en tres zonas, teniendo la zona 1 dos zonas de estacionamiento remotas, la zona 2 tiene tres zonas de estacionamiento remotas y la zona 3 tiene cuatro zonas de estacionamiento remotas. A las zonas de estacionamiento remotas que están dentro de la zona 1 se les asigna siempre la escalerilla de 1 metro; a las que están dentro de la zona 2 se les asigna siempre la escalerilla de 2 metros; y las de la zona 3, la de 3 metros.

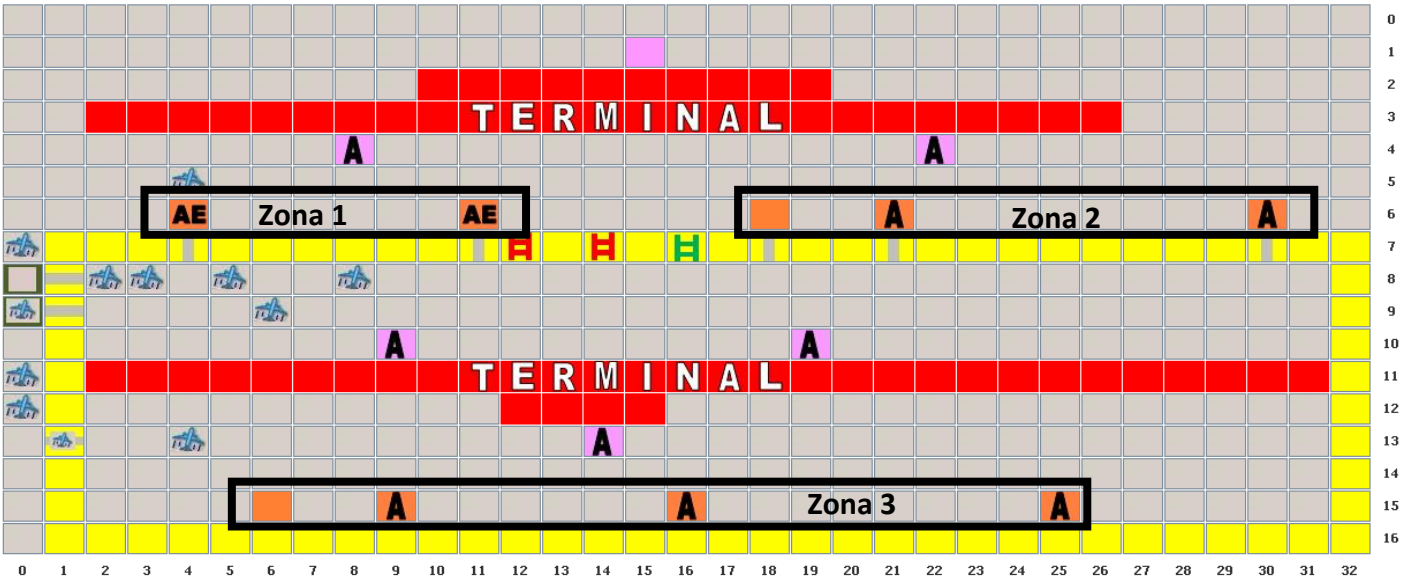


Ilustración 42: Zonas del aeropuerto simulado para la prueba 3

5.1.4. Prueba 4

En esta prueba el desarrollador del sistema ha simulado que es un coordinador de servicios aeroportuarios. El razonamiento seguido en esta prueba para la asignación y cancelación de escalerillas es que, dependiendo de la subclase del avión, se les va a asignar una escalerilla u otra: para todos los aviones que se dirijan a una zona de estacionamiento remota y el primer carácter de su subclase sea un número, se les asigna la escalerilla de 1 metro; para todos los aviones que se dirijan a una zona de estacionamiento remota y el primer carácter sea una letra comprendida entre la A y la D (ambas inclusive), se les asigna la escalerilla de 2 metros; y, para el resto de aviones, se les asigna la escalerilla de 3 metros.

5.1.5. Prueba 5

En esta prueba el desarrollador del sistema ha simulado que es un coordinador de servicios aeroportuarios. El razonamiento seguido en esta prueba para la asignación y cancelación de escalerillas es un poco más complejo que en las anteriores pruebas, ya que combina varios aspectos de éstas (zonas del aeropuerto, salida y subclases de los aviones):

- Si la zona de estacionamiento remota asociada al avión está en la zona 1:
  - Si la salida es mayor a 200 minutos:
    - Si el primer carácter de la subclase es un número → se asigna escalerilla de 1 metro
    - Si el primer carácter de la subclase es una letra comprendida entre la A y la D (ambas inclusive) → se asigna escalerilla de 2 metros

        - Para el resto de subclases → se asigna escalerilla de 3 metros
    - Si la salida es menor o igual a 200 minutos:
        - Si el primer carácter de la subclase es un número → se asigna escalerilla de 2 metros
        - Si el primer carácter de la subclase es una letra comprendida entre la A y la D (ambas inclusive) → se asigna escalerilla de 3 metros
        - Para el resto de subclases → se asigna escalerilla de 1 metro

- Si la zona de estacionamiento remota asociada al avión está en la zona 2:
    - Si la salida es mayor a 200 minutos:
        - Si el primer carácter de la subclase es un número → se asigna escalerilla de 3 metros
        - Si el primer carácter de la subclase es una letra comprendida entre la A y la D (ambas inclusive) → se asigna escalerilla de 1 metro
        - Para el resto de subclases → se asigna escalerilla de 2 metros
    - Si la salida es menor o igual a 200 minutos:
        - Si el primer carácter de la subclase es un número → se asigna escalerilla de 1 metro
        - Si el primer carácter de la subclase es una letra comprendida entre la A y la D (ambas inclusive) → se asigna escalerilla de 2 metros
        - Para el resto de subclases → se asigna escalerilla de 3 metros

- Si la zona de estacionamiento remota asociada al avión está en la zona 3:
    - Si la salida es mayor a 200 minutos:
        - Si el primer carácter de la subclase es una letra comprendida entre la A y la D (ambas inclusive) → se asigna escalerilla de 1 metro
        - Para el resto de subclases → se asigna escalerilla de 2 metros
    - Si la salida es menor o igual a 200 minutos → se asigna escalerilla de 3 metros

Se permite la cancelación de escalerillas de 3 metros en aviones que tienen un tiempo de salida mayor de 200 minutos para asignársela a aviones que deben salir en 200 minutos o menos y que soportan este tipo de escalerillas.

5.1.6. Prueba 6

Esta prueba ha sido realizada por una persona de Asseco que lleva muchos años trabajando en el sector aeroportuario y que conoce la forma en la que los coordinadores de servicios de aeropuertos realizan la asignación o cancelación de las escalerillas para el desembarque de los pasajeros. El razonamiento llevado a cabo por esta persona para la asignación y cancelación de escalerillas es el siguiente:

- Si el tiempo que le queda para despegar al avión es menor o igual a 100 minutos:
  - Si el primer carácter de la subclase es un número → se asigna escalerilla de 1 metro o 2 metros más cercana
  - Si el primer carácter de la subclase es una letra comprendida entre la A y la D (ambas inclusive) → se asigna escalerilla de 2 metros o 3 metros más cercana
  - Para el resto de subclases → se asigna escalerilla de 2 metros o 3 metros más cercana

- Si el tiempo que le queda para despegar al avión es mayor de 100 minutos y menor o igual a 400 minutos:
  - Si la distancia del avión a la zona de estacionamiento es menor o igual a 800 metros:
    - Si el primer carácter de la subclase es un número → si está libre la escalerilla de 1 metro, se asigna ésta; si no es así y está libre la de 2 metros, se asigna esta segunda; y, si por el contrario, no hay ninguna libre, se elige la que esté más cerca (la de 1 metro o 2 metros) para ponerla a la cola de asignaciones de esa escalerilla
    - Si el primer carácter de la subclase es una letra comprendida entre la A y la D (ambas inclusive) → si está libre la escalerilla de 2 metros, se asigna ésta; si no es así y está libre la de 3 metros, se asigna esta segunda; y, si por el contrario, no hay ninguna libre, se elige la que esté más cerca (la de 2 metros o 3 metros) para ponerla a la cola de asignaciones de esa escalerilla
    - Para el resto de subclases → si está libre la escalerilla de 2 metros, se asigna ésta; si no es así y está libre la de 3 metros, se asigna esta segunda; y, si por el contrario, no hay ninguna libre, se elige la que esté más cerca (la de 2 metros o 3 metros) para ponerla a la cola de asignaciones de esa escalerilla
  - Si la distancia del avión a la zona de estacionamiento es mayor a 800 metros:
    - Si el primer carácter de la subclase es un número → si está libre la escalerilla de 1 metro, se asigna ésta; si no es así y está libre la de 2 metros, se asigna esta segunda; y, si por el contrario, no hay ninguna libre, no se asigna ninguna escalerilla
    - Si el primer carácter de la subclase es una letra comprendida entre la A y la D (ambas inclusive) → si está libre la escalerilla de 2 metros, se asigna ésta; si no es así y está libre la de 3 metros, se asigna esta segunda; y, si por el contrario, no hay ninguna libre, no se asigna ninguna escalerilla

- Para el resto de subclases → si está libre la escalerilla de 2 metros, se asigna ésta; si no es así y está libre la de 3 metros, se asigna esta segunda; y, si por el contrario, no hay ninguna libre, no se asigna ninguna escalerilla

➢ Si el tiempo que le queda para despegar al avión es mayor de 400 minutos:
  - Si el primer carácter de la subclase es un número → si está libre la escalerilla de 1 metro, se asigna ésta; si no es así y está libre la de 2 metros, se asigna esta segunda; y, si por el contrario, no hay ninguna libre, no se asigna ninguna escalerilla
  - Si el primer carácter de la subclase es una letra comprendida entre la A y la D (ambas inclusive) → si está libre la escalerilla de 2 metros, se asigna ésta; si no es así y está libre la de 3 metros, se asigna esta segunda; y, si por el contrario, no hay ninguna libre, no se asigna ninguna escalerilla
  - Para el resto de subclases → si está libre la escalerilla de 2 metros, se asigna ésta; si no es así y está libre la de 3 metros, se asigna esta segunda; y, si por el contrario, no hay ninguna libre, no se asigna ninguna escalerilla

Se permite la cancelación de escalerillas en aviones que tienen un tiempo de salida mayor de 100 minutos para asignársela a aviones que deben salir en 100 minutos o menos y que soportan este tipo de escalerillas.

5.2. Resultados

Los resultados obtenidos tras realizar las pruebas descritas anteriormente son los siguientes.

	Nº instancias	Instancias bien clasificadas	% de aciertos	Instancias mal clasificadas	% de fallos	Nº reglas generadas
Prueba 1	9182	8462	92,16%	720	7,84%	51
Prueba 2	10201	8789	86,16%	1412	13,84%	54
Prueba 3	9031	8038	89%	993	11%	24
Prueba 4	9990	8435	84,43%	1555	15,57%	3524
Prueba 5	11249	9398	83,55%	1851	16,45%	5089
Prueba 6	20000	15353	76,77%	4647	23,23%	12599

Tabla 60: Resultados de las pruebas

Como se puede ver las pruebas realizadas por el desarrollador tienen un mayor porcentaje de aciertos en la asignación y cancelación de las escalerillas. Esto es normal ya que el razonamiento llevado a cabo en estas pruebas es de menor dificultad que el expuesto en la prueba 6, en la que son necesarias muchos más reglas para definir el

razonamiento seguido. Además, al estudiar las reglas generadas, se puede comprobar que la gran diferencia que existe entre el número de reglas generadas en las pruebas 4, 5 y 6 con respecto al resto, se debe a que en estas pruebas se tiene en cuenta la subclase del avión para realizar la asignación o cancelación, lo que hace que el número de reglas se multiplique muchísimo (es necesario al menos una regla por subclase y zona de estacionamiento remota y hay 222 subclases de aviones diferentes y 9 zonas de estacionamiento remotas).

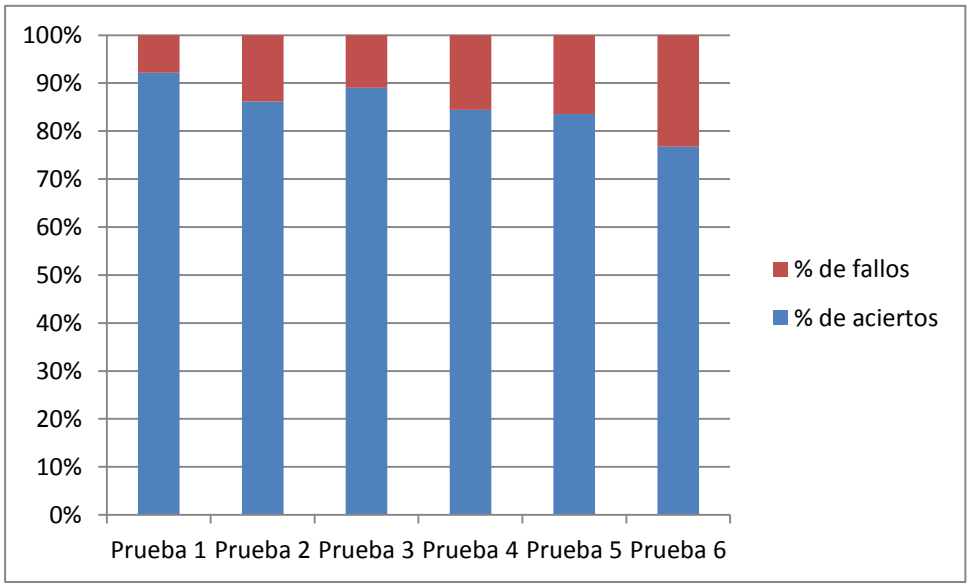


Ilustración 43: Porcentaje de aciertos y fallos en las pruebas

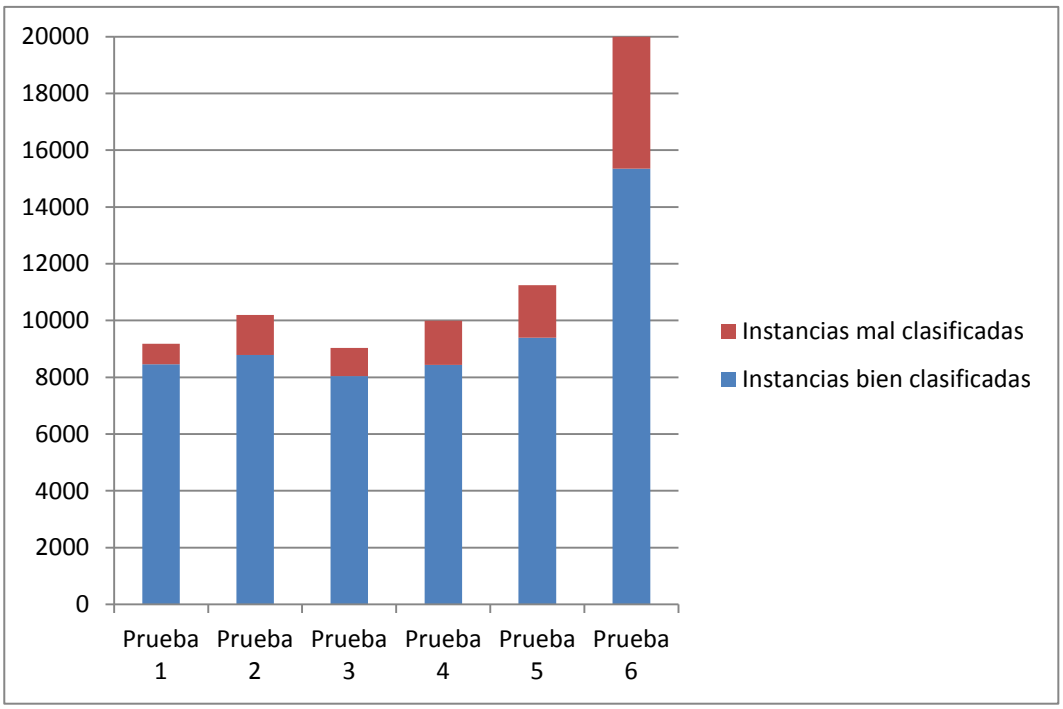


Ilustración 44: Número de instancias mal y bien clasificadas en las pruebas

En todas las pruebas se generan N reglas por cada zona de estacionamiento. El conjunto de reglas de las zonas de estacionamiento locales está formado en todas las pruebas por una única regla que indica que siempre se haga “nada”, es decir, ni se cancelen ni se asignen escalerillas. Esto es lógico ya que en las zonas de estacionamiento locales los pasajeros son desembarcados del avión por la pasarela y no por las escalerillas y, por tanto, no hace falta que se le realice tal servicio.

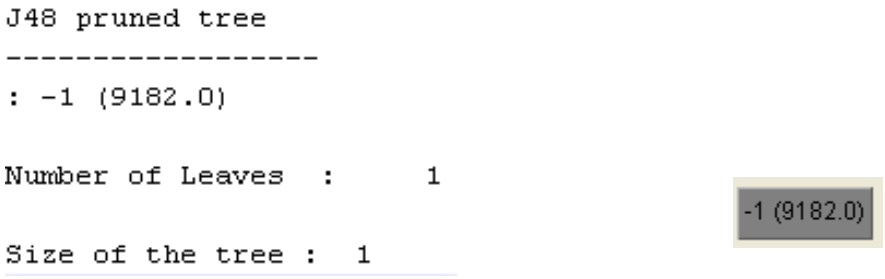


Ilustración 45: Reglas y árbol J48 de zona de estacionamiento local en prueba 1

En cambio, para las zonas de estacionamiento remotas, sí es necesario un razonamiento más complejo y, por tanto, el número de reglas por cada zona de estacionamiento remota es mayor.

En la pruebas de la 1 a la 5, tal y como se ha indicado anteriormente, el porcentaje de aciertos es muy alto (ninguna baja del 83% de aciertos e incluso la primera prueba supera el 90%), es decir, la acción a realizar aprendida por el modelo (asignar una escalerilla, cancelar la asignación de una escalerilla o no hacer nada) se asemeja mucho al razonamiento seguido durante la simulación y, por tanto, se puede concluir que el aprendizaje ha sido satisfactorio. Para corroborar esto se ha arrancado la simulación en modo de asignación y cancelación automática, basándose en el conjunto de reglas generado, y se ha comprobado que es así.

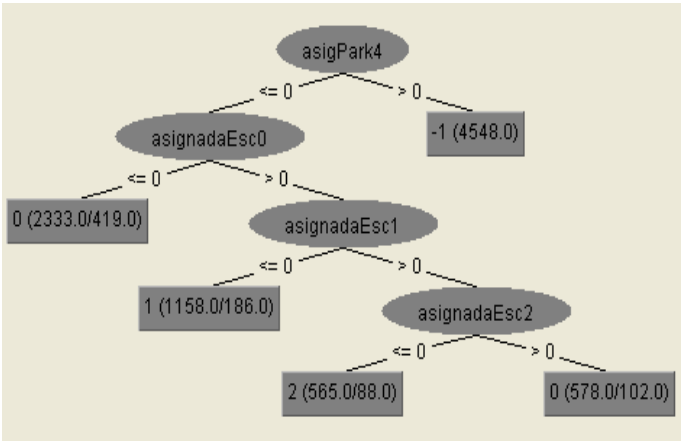
La mayoría de los fallos en la asignación, que se producen en las pruebas, son debidos a que en el proceso de entrenamiento existe ruido, es decir, no siempre se asignaba una escalerilla en el momento preciso (debido a la cantidad de variables que debía de tener en cuenta) y de ahí el porcentaje de error en la prueba.

A continuación se van a analizar un poco más a fondo las reglas construidas en las pruebas 1, 3 y 5.

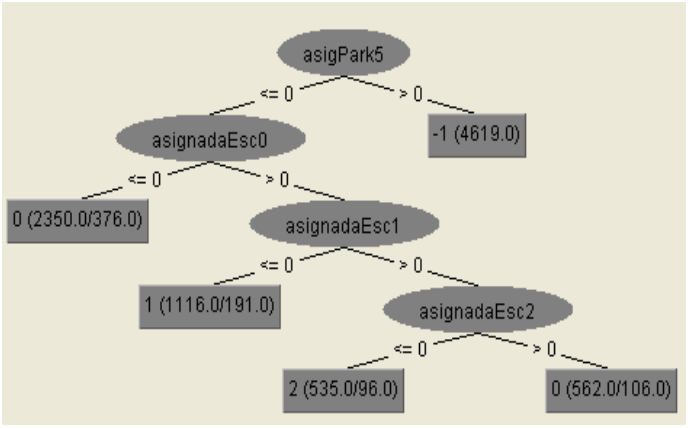
5.2.1. Reglas construidas y árboles J48 en prueba 1

```
J48 pruned tree
------------------

asigPark3 <= 0
|   asignadaEsc0 <= 0: 0 (2253.0/390.0)
|   asignadaEsc0 > 0
|   |   asignadaEsc1 <= 0: 1 (1165.0/196.0)
|   |   asignadaEsc1 > 0
|   |   |   asignadaEsc2 <= 0: 2 (592.0/105.0)
|   |   |   asignadaEsc2 > 0: 0 (530.0/94.0)
asigPark3 > 0: -1 (4642.0)

Number of Leaves  :     5

Size of the tree :  9
```

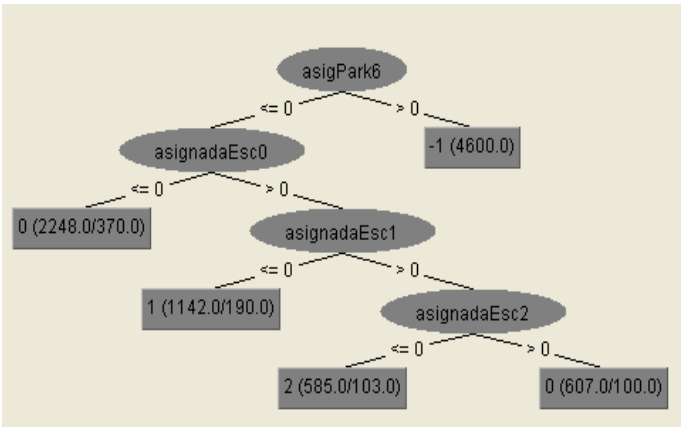

```
J48 pruned tree
------------------

asigPark4 <= 0
|   asignadaEsc0 <= 0: 0 (2333.0/419.0)
|   asignadaEsc0 > 0
|   |   asignadaEsc1 <= 0: 1 (1158.0/186.0)
|   |   asignadaEsc1 > 0
|   |   |   asignadaEsc2 <= 0: 2 (565.0/88.0)
|   |   |   asignadaEsc2 > 0: 0 (578.0/102.0)
asigPark4 > 0: -1 (4548.0)

Number of Leaves  :     5

Size of the tree :  9
```



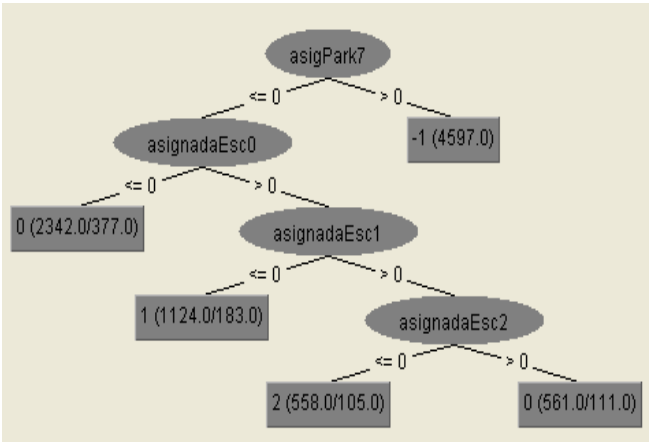

```
J48 pruned tree
------------------

asigPark5 <= 0
|   asignadaEsc0 <= 0: 0 (2350.0/376.0)
|   asignadaEsc0 > 0
|   |   asignadaEsc1 <= 0: 1 (1116.0/191.0)
|   |   asignadaEsc1 > 0
|   |   |   asignadaEsc2 <= 0: 2 (535.0/96.0)
|   |   |   asignadaEsc2 > 0: 0 (562.0/106.0)
asigPark5 > 0: -1 (4619.0)

Number of Leaves  :     5

Size of the tree :  9
```



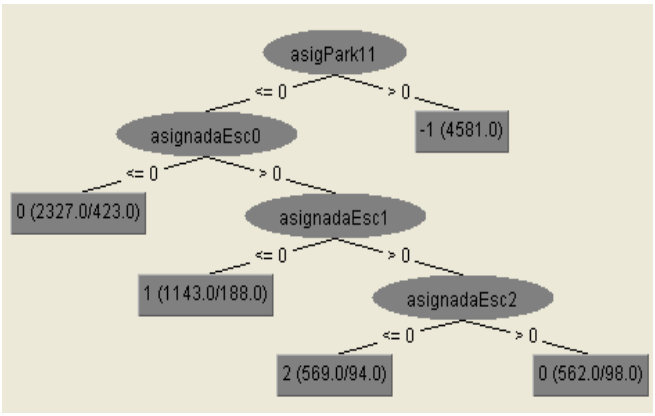
```
J48 pruned tree
------------------

asigPark6 <= 0
|   asignadaEsc0 <= 0: 0 (2248.0/370.0)
|   asignadaEsc0 > 0
|   |   asignadaEsc1 <= 0: 1 (1142.0/190.0)
|   |   asignadaEsc1 > 0
|   |   |   asignadaEsc2 <= 0: 2 (585.0/103.0)
|   |   |   asignadaEsc2 > 0: 0 (607.0/100.0)
asigPark6 > 0: -1 (4600.0)

Number of Leaves  :     5

Size of the tree :  9
```



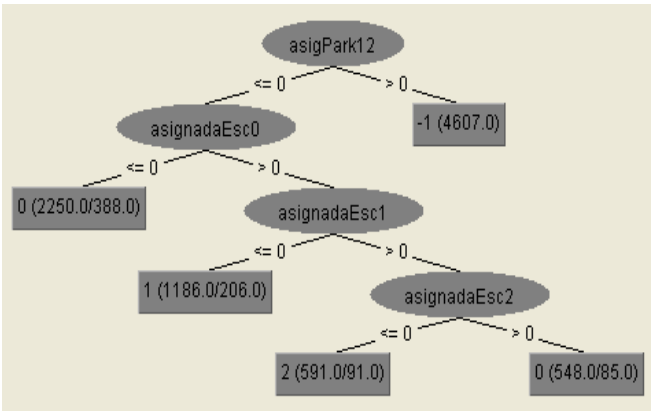

```
J48 pruned tree
------------------

asigPark7 <= 0
|   asignadaEsc0 <= 0: 0 (2342.0/377.0)
|   asignadaEsc0 > 0
|   |   asignadaEsc1 <= 0: 1 (1124.0/183.0)
|   |   asignadaEsc1 > 0
|   |   |   asignadaEsc2 <= 0: 2 (558.0/105.0)
|   |   |   asignadaEsc2 > 0: 0 (561.0/111.0)
asigPark7 > 0: -1 (4597.0)

Number of Leaves  :     5

Size of the tree :  9
```



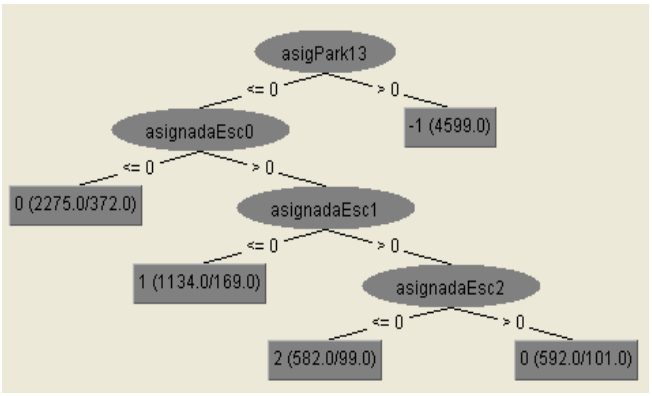

```
J48 pruned tree
------------------

asigPark11 <= 0
|   asignadaEsc0 <= 0: 0 (2327.0/423.0)
|   asignadaEsc0 > 0
|   |   asignadaEsc1 <= 0: 1 (1143.0/188.0)
|   |   asignadaEsc1 > 0
|   |   |   asignadaEsc2 <= 0: 2 (569.0/94.0)
|   |   |   asignadaEsc2 > 0: 0 (562.0/98.0)
asigPark11 > 0: -1 (4581.0)

Number of Leaves  :     5

Size of the tree :  9
```



```
J48 pruned tree
------------------

asigPark12 <= 0
|   asignadaEsc0 <= 0: 0 (2250.0/388.0)
|   asignadaEsc0 > 0
|   |   asignadaEsc1 <= 0: 1 (1186.0/206.0)
|   |   asignadaEsc1 > 0
|   |   |   asignadaEsc2 <= 0: 2 (591.0/91.0)
|   |   |   asignadaEsc2 > 0: 0 (548.0/85.0)
asigPark12 > 0: -1 (4607.0)

Number of Leaves  :     5

Size of the tree :  9
```




```
J48 pruned tree
------------------

asigPark13 <= 0
|   asignadaEsc0 <= 0: 0 (2275.0/372.0)
|   asignadaEsc0 > 0
|   |   asignadaEsc1 <= 0: 1 (1134.0/169.0)
|   |   asignadaEsc1 > 0
|   |   |   asignadaEsc2 <= 0: 2 (582.0/99.0)
|   |   |   asignadaEsc2 > 0: 0 (592.0/101.0)
asigPark13 > 0: -1 (4599.0)

Number of Leaves  :     5

Size of the tree :  9
```




```
J48 pruned tree
------------------

asigPark14 <= 0
|   asignadaEsc0 <= 0: 0 (2276.0/354.0)
|   asignadaEsc0 > 0
|   |   asignadaEsc1 <= 0: 1 (1069.0/169.0)
|   |   asignadaEsc1 > 0
|   |   |   asignadaEsc2 <= 0: 2 (549.0/90.0)
|   |   |   asignadaEsc2 > 0: 0 (608.0/107.0)
asigPark14 > 0: -1 (4680.0)

Number of Leaves  :     5

Size of the tree :  9
```

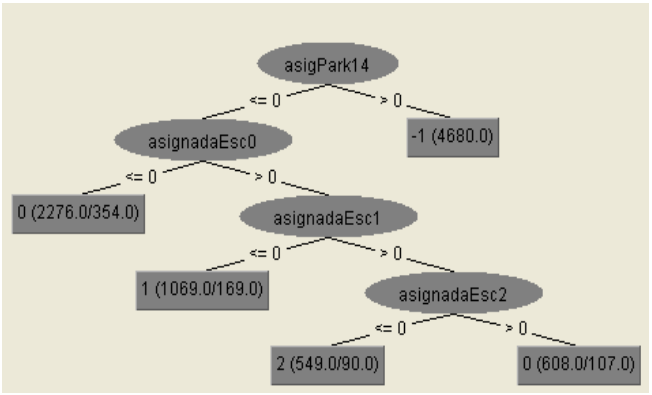


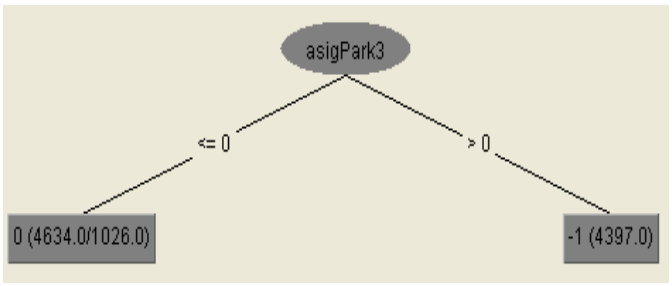
Ilustración 46 (a, b, c, d, e, f, g, h, i): Reglas y árboles J48 de zonas de estacionamiento remotas en prueba 1

En las reglas generadas para las diferentes zonas de estacionamiento remotas de la prueba 1, primero se comprueba si la zona de estacionamiento (parking) tiene ya asignada una escalerilla y, si es así, no se hace nada (asigParkXXX>0:-1); si la zona de estacionamiento no tiene asignada aún una escalerilla (asigParkXXX<=0), se verifica si la escalerilla con identificador 0 (de 1 metro) está libre (asignadaEsc0<=0) y, si es así, se le asigna esta escalerilla (asignadaEsc0<=0:0); si la de 1 metro no está libre

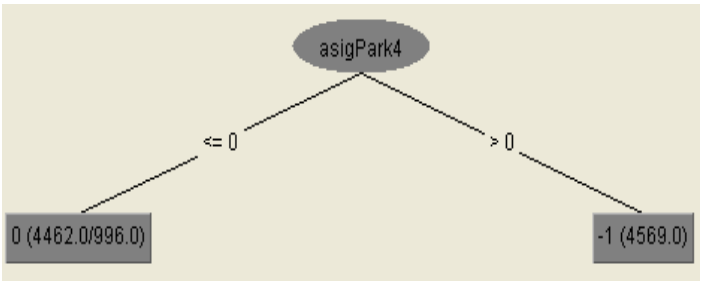
(asignadaEsc0>0), se observa si la escalerilla con identificador 1 (de 2 metros) está libre (asignadaEsc1<=0), y, si es así, se le asigna esta escalerilla (asignadaEsc1<=0:0); si la de 2 metros no está libre (asignadaEsc1>0), se ve si la escalerilla con identificador 2 (de 3 metros) está libre (asignadaEsc2<=0), y, si es así, se le asigna esta escalerilla (asignadaEsc2<=0:2); en caso contrario, se le asigna la escalerilla de 1 metro con identificador 0 (asignadaEsc2>0:0).

5.2.1. Reglas construidas y árboles J48 en prueba 3

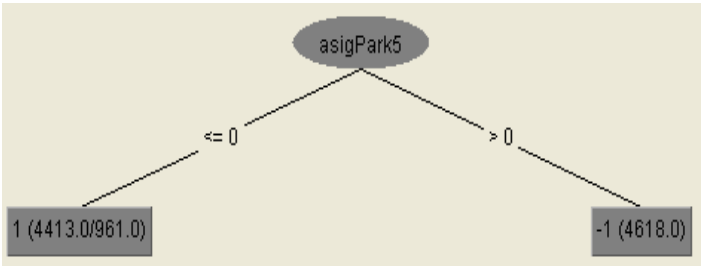
```
J48 pruned tree
------------------

asigPark3 <= 0: 0 (4634.0/1026.0)
asigPark3 > 0: -1 (4397.0)

Number of Leaves  :     2

Size of the tree :  3
```



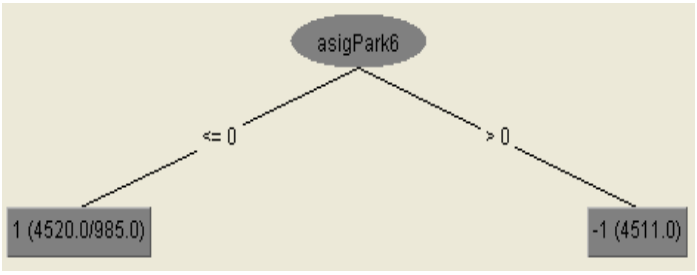

```
J48 pruned tree
------------------

asigPark4 <= 0: 0 (4462.0/996.0)
asigPark4 > 0: -1 (4569.0)

Number of Leaves  :     2

Size of the tree :  3
```



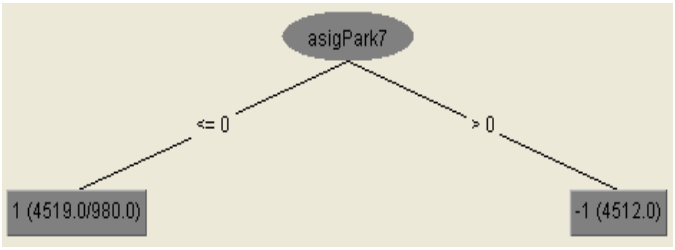

```
J48 pruned tree
------------------

asigPark5 <= 0: 1 (4413.0/961.0)
asigPark5 > 0: -1 (4618.0)

Number of Leaves  :     2

Size of the tree :  3
```



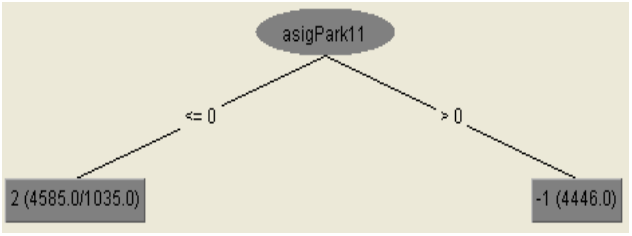

```
J48 pruned tree
------------------

asigPark6 <= 0: 1 (4520.0/985.0)
asigPark6 > 0: -1 (4511.0)

Number of Leaves  :     2

Size of the tree :  3
```



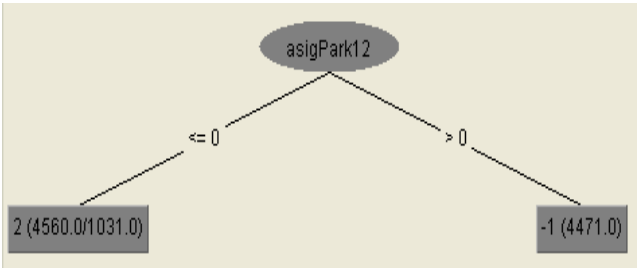
```
J48 pruned tree
------------------

asigPark7 <= 0: 1 (4519.0/980.0)
asigPark7 > 0: -1 (4512.0)

Number of Leaves  :      2

Size of the tree :  3
```




```
J48 pruned tree
------------------

asigPark11 <= 0: 2 (4585.0/1035.0)
asigPark11 > 0: -1 (4446.0)

Number of Leaves  :      2

Size of the tree :  3
```




```
J48 pruned tree
------------------

asigPark12 <= 0: 2 (4560.0/1031.0)
asigPark12 > 0: -1 (4471.0)

Number of Leaves  :      2

Size of the tree :  3
```




```
J48 pruned tree
------------------

asigPark13 <= 0: 2 (4512.0/1000.0)
asigPark13 > 0: -1 (4519.0)

Number of Leaves  :      2

Size of the tree :  3
```


```
J48 pruned tree
------------------

asigPark14 <= 0: 2 (4533.0/993.0)
asigPark14 > 0: -1 (4498.0)

Number of Leaves  :      2

Size of the tree :  3
```

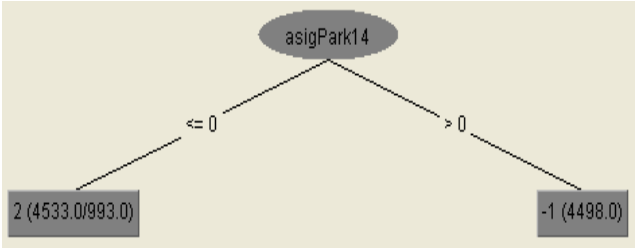


Ilustración 47 (a, b, c, d, e, f, g, h, i): Reglas y árboles J48 de zonas de estacionamiento remotas en prueba 3

En las reglas generadas para las diferentes zonas de estacionamiento remotas de la prueba 3, primero se comprueba si la zona de estacionamiento (parking) tiene ya asignada una escalerilla y, si es así, no se hace nada (asigParkXXX>0:-1); si la zona de estacionamiento no tiene asignada aún una escalerilla (asigParkXXX<=0), se asigna una escalerilla u otra dependiendo de la zona de estacionamiento: para las zonas de estacionamiento 3 y 4 se asigna la escalerilla con identificador 0, que es de 1 metro (asigParkXXX<=0:0); para las zonas de estacionamiento 5, 6 y 7 se asigna la escalerilla con identificador 1, que es de 2 metros (asigParkXXX<=0:1); y para las zonas de estacionamiento 11, 12, 13 y 14 se asigna la escalerilla con identificador 2, que es de 3 metros (asigParkXXX<=0:2).

5.2.2. Reglas construidas en prueba 5

En la explicación de esta prueba sólo se muestra el conjunto de reglas generadas de una de las zonas de estacionamiento remotas, más concretamente el de la zona de estacionamiento con identificador 13, y tampoco ninguno de los árboles de decisión J48 construidos debido al gran número de reglas que son necesarias para definir el razonamiento de las decisiones tomadas para cada zona de estacionamiento.

```
J48 pruned tree
------------------

asigPark13 <= 0
|   tiempoSalirAvionPark13 <= 200
|   |   subclaseAvion4 = 100: 2 (13.0/5.0)
|   |   subclaseAvion4 = 130: 0 (23.0/9.0)
|   |   subclaseAvion4 = 141
|   |   |   asignadaEsc2 <= 0: 2 (6.0/1.0)
|   |   |   asignadaEsc2 > 0
|   |   |   |   asigPark4 <= 0: 0 (7.0)
|   |   |   |   asigPark4 > 0
|   |   |   |   |   tiempoSalirAvionPark7 <= 160: 0 (2.0)
|   |   |   |   |   tiempoSalirAvionPark7 > 160: 2 (2.0)
|   |   subclaseAvion4 = 142
|   |   |   tiempoSalirAvionPark5 <= 32: 2 (4.0/1.0)
|   |   |   tiempoSalirAvionPark5 > 32: 0 (9.0)
|   |   subclaseAvion4 = 143
|   |   |   distAvionPark <= 549: 2 (4.0)
|   |   |   distAvionPark > 549
|   |   |   |   tiempoSalirAvionPark11 <= 18: 2 (2.0)
|   |   |   |   tiempoSalirAvionPark11 > 18: 0 (9.0)
|   |   subclaseAvion4 = 14Y: 2 (21.0/8.0)
|   |   subclaseAvion4 = 301: 0 (11.0/3.0)
|   |   subclaseAvion4 = 302: 0 (18.0/8.0)
|   |   subclaseAvion4 = 312
|   |   |   asigPark6 <= 0: 2 (6.0/1.0)
|   |   |   asigPark6 > 0: 0 (9.0/2.0)
|   |   subclaseAvion4 = 313: 2 (15.0/5.0)
```

```
|   |   subclaseAvion4 = 313: 2 (15.0/5.0)
|   |   subclaseAvion4 = 314
|   |   |   tiempoSalirAvionPark6 <= 182
|   |   |   |   asigPark7 <= 0
|   |   |   |   |   distParkEsc2 <= 65: 0 (4.0)
|   |   |   |   |   distParkEsc2 > 65: 2 (2.0)
|   |   |   |   asigPark7 > 0: 0 (5.0)
|   |   |   tiempoSalirAvionPark6 > 182: 2 (8.0)
|   |   subclaseAvion4 = 318
|   |   |   distParkEsc0 <= 138: 2 (10.0)
|   |   |   distParkEsc0 > 138: 0 (7.0/1.0)
|   |   subclaseAvion4 = 319: 2 (20.0/8.0)
|   |   subclaseAvion4 = 31A
|   |   |   asigPark7 <= 0: 0 (8.0/3.0)
|   |   |   asigPark7 > 0: 2 (11.0/4.0)
|   |   subclaseAvion4 = 31B: 0 (6.0)
|   |   subclaseAvion4 = 31D: 0 (19.0/9.0)
|   |   subclaseAvion4 = 320
|   |   |   tiempoSalirAvionPark3 <= 208
|   |   |   |   asigPark5 <= 0
|   |   |   |   |   tiempoSalirAvionPark3 <= 141: 0 (2.0)
|   |   |   |   |   tiempoSalirAvionPark3 > 141: 2 (3.0)
|   |   |   |   asigPark5 > 0: 0 (7.0)
|   |   |   tiempoSalirAvionPark3 > 208: 2 (6.0)
|   |   subclaseAvion4 = 321: 0 (20.0/9.0)
|   |   subclaseAvion4 = 322
|   |   |   asigPark11 <= 0: 2 (7.0/1.0)
|   |   |   asigPark11 > 0: 0 (7.0/1.0)
|   |   subclaseAvion4 = 32A
|   |   |   asigPark14 <= 0: 0 (5.0)
|   |   |   asigPark14 > 0
|   |   |   |   tiempoSalirAvionPark3 <= 170: 2 (8.0)
|   |   |   |   tiempoSalirAvionPark3 > 170: 0 (4.0/1.0)
|   |   subclaseAvion4 = 32B
|   |   |   asigPark12 <= 0: 0 (7.0/2.0)
|   |   |   asigPark12 > 0: 2 (6.0)
|   |   subclaseAvion4 = 32D: 0 (18.0/9.0)
|   |   subclaseAvion4 = 32S: 2 (18.0/5.0)
|   |   subclaseAvion4 = 330: 0 (21.0/9.0)
|   |   subclaseAvion4 = 332
|   |   |   distParkEsc0 <= 151
|   |   |   |   distParkEsc2 <= 166: 0 (12.0)
|   |   |   |   distParkEsc2 > 166: 2 (2.0)
|   |   |   distParkEsc0 > 151: 2 (5.0)
|   |   subclaseAvion4 = 333: 0 (20.0/10.0)
|   |   subclaseAvion4 = 340: 2 (26.0/11.0)
|   |   subclaseAvion4 = 342
|   |   |   asigPark7 <= 0: 2 (9.0)
|   |   |   asigPark7 > 0: 0 (16.0/4.0)
|   |   subclaseAvion4 = 343: 0 (25.0/10.0)
|   |   subclaseAvion4 = 345
|   |   |   asignadaEsc1 <= 0: 0 (6.0)
|   |   |   asignadaEsc1 > 0
|   |   |   |   distParkEsc0 <= 101: 2 (5.0)
|   |   |   |   distParkEsc0 > 101: 0 (6.0/1.0)
```

```
|   |   subclaseAvion4 = 346
|   |   |   asigPark14 <= 0: 2 (7.0/1.0)
|   |   |   asigPark14 > 0
|   |   |   |   distParkEsc2 <= 165: 0 (8.0)
|   |   |   |   distParkEsc2 > 165: 2 (2.0)
|   |   subclaseAvion4 = 702
|   |   |   distParkEsc0 <= 45: 0 (5.0)
|   |   |   distParkEsc0 > 45: 2 (15.0/2.0)
|   |   subclaseAvion4 = 703
|   |   |   tiempoSalirAvionPark13 <= 100: 0 (12.0/2.0)
|   |   |   tiempoSalirAvionPark13 > 100: 2 (6.0)
|   |   subclaseAvion4 = 704
|   |   |   asigPark7 <= 0: 2 (4.0/1.0)
|   |   |   asigPark7 > 0: 0 (10.0/1.0)
|   |   subclaseAvion4 = 70F
|   |   |   asigPark3 <= 0
|   |   |   |   distParkEsc1 <= 22: 0 (2.0)
|   |   |   |   distParkEsc1 > 22: 2 (8.0)
|   |   |   asigPark3 > 0: 0 (7.0/1.0)
|   |   subclaseAvion4 = 70Y: 0 (16.0/1.0)
|   |   subclaseAvion4 = 712: 2 (18.0/6.0)
|   |   subclaseAvion4 = 717
|   |   |   asigPark4 <= 0: 0 (5.0)
|   |   |   asigPark4 > 0
|   |   |   |   tiempoSalirAvionPark4 <= 20: 0 (2.0)
|   |   |   |   tiempoSalirAvionPark4 > 20: 2 (5.0)
|   |   subclaseAvion4 = 721: 2 (21.0/10.0)
|   |   subclaseAvion4 = 722
|   |   |   tiempoSalirAvionPark14 <= 123: 2 (8.0)
|   |   |   tiempoSalirAvionPark14 > 123
|   |   |   |   distParkEsc0 <= 27: 2 (4.0)
|   |   |   |   distParkEsc0 > 27: 0 (10.0/1.0)
|   |   subclaseAvion4 = 72A: 0 (22.0/11.0)
|   |   subclaseAvion4 = 72S: 0 (20.0/7.0)
|   |   subclaseAvion4 = 72X
|   |   |   asigPark4 <= 0: 2 (8.0)
|   |   |   asigPark4 > 0
|   |   |   |   asignadaEsc0 <= 0: 0 (5.0/1.0)
|   |   |   |   asignadaEsc0 > 0: 2 (3.0)
|   |   subclaseAvion4 = 72Y
|   |   |   distParkEsc0 <= 112
|   |   |   |   asigPark6 <= 0: 2 (5.0)
|   |   |   |   asigPark6 > 0: 0 (3.0/1.0)
|   |   |   distParkEsc0 > 112: 0 (9.0)
|   |   subclaseAvion4 = 732: 0 (14.0/4.0)
|   |   subclaseAvion4 = 733
|   |   |   asigPark4 <= 0
|   |   |   |   tiempoSalirAvionPark7 <= 55: 2 (2.0)
|   |   |   |   tiempoSalirAvionPark7 > 55: 0 (8.0)
|   |   |   asigPark4 > 0: 2 (4.0/1.0)
|   |   subclaseAvion4 = 734
|   |   |   asignadaEsc0 <= 0
|   |   |   |   distAvionPark <= 152: 2 (2.0)
|   |   |   |   distAvionPark > 152: 0 (11.0/1.0)
|   |   |   asignadaEsc0 > 0: 2 (11.0/1.0)
```

```
|   |   subclaseAvion4 = 735: 2 (17.0/7.0)
|   |   subclaseAvion4 = 736: 0 (15.0/5.0)
|   |   subclaseAvion4 = 737
|   |   |   asignadaEscl <= 0
|   |   |   |   tiempoSalirAvionPark6 <= 100: 0 (2.0)
|   |   |   |   tiempoSalirAvionPark6 > 100: 2 (3.0)
|   |   |   asignadaEscl > 0: 0 (5.0)
|   |   subclaseAvion4 = 738
|   |   |   tiempoSalirAvionPark4 <= 155: 0 (8.0)
|   |   |   tiempoSalirAvionPark4 > 155
|   |   |   |   tiempoSalirAvionPark4 <= 252: 2 (5.0)
|   |   |   |   tiempoSalirAvionPark4 > 252: 0 (2.0)
|   |   subclaseAvion4 = 739
|   |   |   distParkEscl <= 143
|   |   |   |   asigPark12 <= 0
|   |   |   |   |   distParkEscl <= 57: 0 (4.0)
|   |   |   |   |   distParkEscl > 57: 2 (2.0)
|   |   |   |   asigPark12 > 0: 0 (6.0)
|   |   |   distParkEscl > 143: 2 (6.0)
|   |   subclaseAvion4 = 73A
|   |   |   asigPark5 <= 0
|   |   |   |   tiempoSalirAvionPark11 <= 253: 2 (5.0)
|   |   |   |   tiempoSalirAvionPark11 > 253: 0 (5.0/1.0)
|   |   |   asigPark5 > 0: 0 (8.0)
|   |   subclaseAvion4 = 73G
|   |   |   asigPark3 <= 0
|   |   |   |   asignadaEsc2 <= 0: 0 (3.0)
|   |   |   |   asignadaEsc2 > 0: 2 (4.0/1.0)
|   |   |   asigPark3 > 0: 2 (5.0)
|   |   subclaseAvion4 = 73H
|   |   |   tiempoSalirAvionPark13 <= 76: 2 (5.0/1.0)
|   |   |   tiempoSalirAvionPark13 > 76: 0 (8.0)
|   |   subclaseAvion4 = 73X
|   |   |   asigPark7 <= 0
|   |   |   |   tiempoSalirAvionPark6 <= 228: 0 (4.0)
|   |   |   |   tiempoSalirAvionPark6 > 228: 2 (2.0)
|   |   |   asigPark7 > 0: 2 (5.0)
|   |   subclaseAvion4 = 73Y
|   |   |   tiempoSalirAvionPark14 <= 100: 2 (3.0)
|   |   |   tiempoSalirAvionPark14 > 100
|   |   |   |   tiempoSalirAvionPark3 <= 242: 0 (8.0)
|   |   |   |   tiempoSalirAvionPark3 > 242: 2 (2.0)
|   |   subclaseAvion4 = 741: 0 (17.0/5.0)
|   |   subclaseAvion4 = 742
|   |   |   tiempoSalirAvionPark3 <= 219: 2 (9.0/3.0)
|   |   |   tiempoSalirAvionPark3 > 219: 0 (6.0/1.0)
|   |   subclaseAvion4 = 743: 2 (27.0/12.0)
|   |   subclaseAvion4 = 744: 0 (19.0/8.0)
|   |   subclaseAvion4 = 747: 0 (19.0/7.0)
|   |   subclaseAvion4 = 74C: 2 (20.0/9.0)
|   |   subclaseAvion4 = 74D
|   |   |   tiempoSalirAvionPark12 <= 187: 0 (14.0/2.0)
|   |   |   tiempoSalirAvionPark12 > 187: 2 (4.0)
|   |   subclaseAvion4 = 74E: 0 (16.0/6.0)
```

```
|   |   subclaseAvion4 = 74F
|   |   |   asigPark12 <= 0: 0 (8.0)
|   |   |   asigPark12 > 0: 2 (9.0/3.0)
|   |   subclaseAvion4 = 74L
|   |   |   tiempoSalirAvionPark4 <= 129
|   |   |   |   asigPark4 <= 0
|   |   |   |   |   asigPark11 <= 0: 0 (2.0)
|   |   |   |   |   asigPark11 > 0: 2 (3.0/1.0)
|   |   |   |   asigPark4 > 0: 2 (4.0)
|   |   |   tiempoSalirAvionPark4 > 129: 0 (10.0)
|   |   subclaseAvion4 = 74M: 2 (20.0/7.0)
|   |   subclaseAvion4 = 74X
|   |   |   asigPark7 <= 0
|   |   |   |   tiempoSalirAvionPark14 <= 260: 2 (4.0)
|   |   |   |   tiempoSalirAvionPark14 > 260: 0 (3.0)
|   |   |   asigPark7 > 0: 0 (7.0)
|   |   subclaseAvion4 = 74Y
|   |   |   asigPark6 <= 0
|   |   |   |   tiempoSalirAvionPark7 <= 237: 0 (9.0)
|   |   |   |   tiempoSalirAvionPark7 > 237: 2 (3.0/1.0)
|   |   |   asigPark6 > 0: 2 (8.0/1.0)
|   |   subclaseAvion4 = 752
|   |   |   asigPark7 <= 0
|   |   |   |   asigPark3 <= 0: 2 (7.0/1.0)
|   |   |   |   asigPark3 > 0: 0 (4.0)
|   |   |   asigPark7 > 0: 0 (7.0)
|   |   subclaseAvion4 = 753
|   |   |   asigPark7 <= 0: 0 (10.0/2.0)
|   |   |   asigPark7 > 0
|   |   |   |   asigPark6 <= 0
|   |   |   |   |   distParkEsc2 <= 53: 0 (3.0)
|   |   |   |   |   distParkEsc2 > 53: 2 (2.0)
|   |   |   |   asigPark6 > 0: 2 (5.0)
|   |   subclaseAvion4 = 757: 2 (18.0/8.0)
|   |   subclaseAvion4 = 75F
|   |   |   tipoEscl = 1 metro: 0 (1.0)
|   |   |   tipoEscl = 1.50 metros: 2 (2.0)
|   |   |   tipoEscl = 2 metros: 0 (3.0)
|   |   |   tipoEscl = 2.50 metros
|   |   |   |   asignadaEsc0 <= 0: 0 (2.0)
|   |   |   |   asignadaEsc0 > 0: 2 (3.0)
|   |   |   tipoEscl = 3.50 metros: 0 (0.0)
|   |   |   tipoEscl = 4 metros: 2 (1.0)
|   |   |   tipoEscl = 3 metros: 0 (2.0/1.0)
|   |   subclaseAvion4 = 762
|   |   |   asigPark11 <= 0: 2 (4.0)
|   |   |   asigPark11 > 0
|   |   |   |   tiempoSalirAvionPark6 <= 175: 2 (5.0/1.0)
|   |   |   |   tiempoSalirAvionPark6 > 175: 0 (5.0)
|   |   subclaseAvion4 = 763
|   |   |   asigPark14 <= 0: 2 (9.0/2.0)
|   |   |   asigPark14 > 0: 0 (9.0/1.0)
```

```
|   |   subclaseAvion4 = 76A
|   |   |   tipoEsc0 = 1 metro: 2 (2.0)
|   |   |   tipoEsc0 = 1.50 metros: 0 (2.0)
|   |   |   tipoEsc0 = 2 metros: 0 (0.0)
|   |   |   tipoEsc0 = 2.50 metros: 0 (2.0)
|   |   |   tipoEsc0 = 3.50 metros: 2 (2.0)
|   |   |   tipoEsc0 = 4 metros: 0 (2.0)
|   |   |   tipoEsc0 = 3 metros
|   |   |   |   tiempoSalirAvionPark3 <= 208: 0 (3.0)
|   |   |   |   tiempoSalirAvionPark3 > 208: 2 (3.0)
|   |   subclaseAvion4 = 772: 0 (24.0/10.0)
|   |   subclaseAvion4 = 773
|   |   |   asigPark7 <= 0: 0 (4.0/1.0)
|   |   |   asigPark7 > 0: 2 (6.0/1.0)
|   |   subclaseAvion4 = A24
|   |   |   asigPark14 <= 0
|   |   |   |   distParkEsc2 <= 167: 0 (9.0)
|   |   |   |   distParkEsc2 > 167: 2 (3.0/1.0)
|   |   |   asigPark14 > 0: 2 (4.0)
|   |   subclaseAvion4 = A26
|   |   |   asigPark4 <= 0: 0 (12.0/1.0)
|   |   |   asigPark4 > 0: 2 (7.0/1.0)
|   |   subclaseAvion4 = A4F: 0 (19.0/6.0)
|   |   subclaseAvion4 = AB2
|   |   |   tiempoSalirAvionPark13 <= 50: 0 (4.0)
|   |   |   tiempoSalirAvionPark13 > 50
|   |   |   |   distAvionPark <= 1402: 2 (9.0)
|   |   |   |   distAvionPark > 1402: 0 (2.0)
|   |   subclaseAvion4 = AB3
|   |   |   asigPark11 <= 0: 2 (8.0/1.0)
|   |   |   asigPark11 > 0: 0 (6.0/1.0)
|   |   subclaseAvion4 = AB4
|   |   |   tiempoSalirAvionPark11 <= 184
|   |   |   |   tiempoSalirAvionPark3 <= 102: 2 (5.0/1.0)
|   |   |   |   tiempoSalirAvionPark3 > 102: 0 (6.0)
|   |   |   tiempoSalirAvionPark11 > 184: 2 (6.0)
|   |   subclaseAvion4 = AB6: 0 (18.0/6.0)
|   |   subclaseAvion4 = ABB
|   |   |   asigPark4 <= 0: 0 (6.0/1.0)
|   |   |   asigPark4 > 0: 2 (4.0)
|   |   subclaseAvion4 = ABF
|   |   |   tiempoSalirAvionPark3 <= 78: 0 (6.0/2.0)
|   |   |   tiempoSalirAvionPark3 > 78: 2 (7.0/2.0)
|   |   subclaseAvion4 = ABX
|   |   |   distParkEsc0 <= 88: 0 (6.0)
|   |   |   distParkEsc0 > 88: 2 (10.0/1.0)
|   |   subclaseAvion4 = AN2
|   |   |   asigPark3 <= 0: 0 (7.0)
|   |   |   asigPark3 > 0: 2 (9.0/3.0)
|   |   subclaseAvion4 = AN4: 2 (16.0/7.0)
|   |   subclaseAvion4 = AN6: 2 (26.0/10.0)
|   |   subclaseAvion4 = AN7: 2 (24.0/9.0)
|   |   subclaseAvion4 = ANF
|   |   |   tiempoSalirAvionPark4 <= 175: 2 (10.0)
|   |   |   tiempoSalirAvionPark4 > 175
|   |   |   |   tiempoSalirAvionPark5 <= 141: 2 (2.0)
|   |   |   |   tiempoSalirAvionPark5 > 141: 0 (6.0)
```

```
|   |   subclaseAvion4 = AR1
|   |   |   asigPark11 <= 0
|   |   |   |   tiempoSalirAvionPark13 <= 93: 0 (2.0)
|   |   |   |   tiempoSalirAvionPark13 > 93: 2 (5.0)
|   |   |   asigPark11 > 0: 0 (9.0/1.0)
|   |   subclaseAvion4 = AR7
|   |   |   asigPark11 <= 0: 0 (9.0/1.0)
|   |   |   asigPark11 > 0: 2 (5.0)
|   |   subclaseAvion4 = AR8
|   |   |   distParkEsc2 <= 166: 2 (14.0/3.0)
|   |   |   distParkEsc2 > 166: 0 (4.0)
|   |   subclaseAvion4 = AT1
|   |   |   asigPark12 <= 0: 0 (10.0/1.0)
|   |   |   asigPark12 > 0: 2 (7.0/1.0)
|   |   subclaseAvion4 = AT3
|   |   |   asignadaEsc1 <= 0
|   |   |   |   tiempoSalirAvionPark14 <= 135: 2 (4.0)
|   |   |   |   tiempoSalirAvionPark14 > 135: 0 (7.0/1.0)
|   |   |   asignadaEsc1 > 0: 2 (6.0)
|   |   subclaseAvion4 = AT4: 2 (19.0/8.0)
|   |   subclaseAvion4 = AT5: 0 (13.0/5.0)
|   |   subclaseAvion4 = AT7: 2 (18.0/8.0)
|   |   subclaseAvion4 = ATP: 0 (14.0/6.0)
|   |   subclaseAvion4 = B14: 0 (21.0/9.0)
|   |   subclaseAvion4 = B15: 0 (18.0/6.0)
|   |   subclaseAvion4 = B20
|   |   |   asigPark12 <= 0: 0 (6.0)
|   |   |   asigPark12 > 0
|   |   |   |   tiempoSalirAvionPark6 <= 187: 2 (2.0)
|   |   |   |   tiempoSalirAvionPark6 > 187: 0 (2.0)
|   |   subclaseAvion4 = B72: 0 (28.0/13.0)
|   |   subclaseAvion4 = BE1: 0 (30.0/9.0)
|   |   subclaseAvion4 = BE2: 2 (21.0/9.0)
|   |   subclaseAvion4 = BE9: 2 (19.0/9.0)
|   |   subclaseAvion4 = BEH: 0 (21.0/10.0)
|   |   subclaseAvion4 = BEK
|   |   |   asigPark5 <= 0
|   |   |   |   asigPark4 <= 0
|   |   |   |   |   asignadaEsc2 <= 0: 2 (6.0)
|   |   |   |   |   asignadaEsc2 > 0
|   |   |   |   |   |   distAvionPark <= 295: 2 (2.0)
|   |   |   |   |   |   distAvionPark > 295: 0 (2.0)
|   |   |   |   asigPark4 > 0: 0 (5.0/1.0)
|   |   |   asigPark5 > 0: 0 (4.0)
|   |   subclaseAvion4 = BES: 0 (17.0/8.0)
|   |   subclaseAvion4 = BET: 0 (15.0/6.0)
|   |   subclaseAvion4 = BH2: 2 (21.0/9.0)
|   |   subclaseAvion4 = CL6
|   |   |   tiempoSalirAvionPark5 <= 82: 0 (5.0)
|   |   |   tiempoSalirAvionPark5 > 82: 2 (13.0/5.0)
|   |   subclaseAvion4 = CN1
|   |   |   asigPark7 <= 0: 0 (5.0/1.0)
|   |   |   asigPark7 > 0: 2 (5.0)
|   |   subclaseAvion4 = CN2
|   |   |   tipoEscl = 1 metro: 2 (3.0)
|   |   |   tipoEscl = 1.50 metros: 2 (1.0)
|   |   |   tipoEscl = 2 metros: 0 (1.0)
```

```
|   |   |   tipoEsc1 = 2.50 metros: 2 (4.0/1.0)
|   |   |   tipoEsc1 = 3.50 metros: 2 (3.0)
|   |   |   tipoEsc1 = 4 metros
|   |   |   |   distAvionPark <= 1110: 0 (3.0)
|   |   |   |   distAvionPark > 1110: 2 (2.0)
|   |   |   tipoEsc1 = 3 metros: 0 (3.0)
|   |   subclaseAvion4 = CNA: 2 (22.0/6.0)
|   |   subclaseAvion4 = CNJ: 0 (22.0/6.0)
|   |   subclaseAvion4 = CR1
|   |   |   asigPark5 <= 0
|   |   |   |   asigPark7 <= 0: 2 (3.0)
|   |   |   |   asigPark7 > 0: 0 (6.0/1.0)
|   |   |   asigPark5 > 0: 2 (7.0)
|   |   subclaseAvion4 = CR2: 0 (22.0/7.0)
|   |   subclaseAvion4 = CR6
|   |   |   tipoEsc0 = 1 metro: 0 (3.0/1.0)
|   |   |   tipoEsc0 = 1.50 metros: 2 (1.0)
|   |   |   tipoEsc0 = 2 metros: 0 (2.0/1.0)
|   |   |   tipoEsc0 = 2.50 metros: 2 (3.0/1.0)
|   |   |   tipoEsc0 = 3.50 metros: 0 (4.0)
|   |   |   tipoEsc0 = 4 metros
|   |   |   |   tiempoSalirAvionPark3 <= 185: 0 (3.0)
|   |   |   |   tiempoSalirAvionPark3 > 185: 2 (2.0)
|   |   |   tipoEsc0 = 3 metros: 2 (3.0)
|   |   subclaseAvion4 = CR7: 2 (26.0/9.0)
|   |   subclaseAvion4 = CR9
|   |   |   asigPark6 <= 0: 2 (2.0)
|   |   |   asigPark6 > 0: 0 (6.0)
|   |   subclaseAvion4 = CRB: 0 (16.0/7.0)
|   |   subclaseAvion4 = CRJ: 2 (24.0/10.0)
|   |   subclaseAvion4 = CRS: 2 (22.0/10.0)
|   |   subclaseAvion4 = CS2: 2 (15.0/6.0)
|   |   subclaseAvion4 = CS5: 0 (20.0/10.0)
|   |   subclaseAvion4 = CVY
|   |   |   asigPark4 <= 0
|   |   |   |   asigPark3 <= 0: 0 (4.0/1.0)
|   |   |   |   asigPark3 > 0: 2 (8.0/1.0)
|   |   |   asigPark4 > 0: 0 (9.0/1.0)
|   |   subclaseAvion4 = CWC
|   |   |   tiempoSalirAvionPark12 <= 205
|   |   |   |   asigPark3 <= 0: 2 (3.0/1.0)
|   |   |   |   asigPark3 > 0: 0 (7.0)
|   |   |   tiempoSalirAvionPark12 > 205: 2 (4.0)
|   |   subclaseAvion4 = D11
|   |   |   tiempoSalirAvionPark4 <= 258
|   |   |   |   asignadaEsc2 <= 0: 0 (11.0)
|   |   |   |   asignadaEsc2 > 0
|   |   |   |   |   tiempoSalirAvionPark6 <= 221: 2 (2.0)
|   |   |   |   |   tiempoSalirAvionPark6 > 221: 0 (5.0)
|   |   |   tiempoSalirAvionPark4 > 258: 2 (3.0)
|   |   subclaseAvion4 = D1C
|   |   |   tiempoSalirAvionPark6 <= 150: 2 (6.0/1.0)
|   |   |   tiempoSalirAvionPark6 > 150: 0 (9.0/1.0)
|   |   subclaseAvion4 = D1Y: 0 (11.0/4.0)
```

```
|   |   subclaseAvion4 = D38
|   |   |   asigPark14 <= 0
|   |   |   |   tiempoSalirAvionPark5 <= 143: 0 (6.0)
|   |   |   |   tiempoSalirAvionPark5 > 143: 2 (7.0/1.0)
|   |   |   asigPark14 > 0: 2 (5.0)
|   |   subclaseAvion4 = D3F: 2 (13.0/6.0)
|   |   subclaseAvion4 = D6F
|   |   |   asigPark12 <= 0: 0 (7.0/1.0)
|   |   |   asigPark12 > 0
|   |   |   |   asigPark3 <= 0: 2 (6.0)
|   |   |   |   asigPark3 > 0
|   |   |   |   |   asigPark4 <= 0: 2 (2.0)
|   |   |   |   |   asigPark4 > 0: 0 (2.0)
|   |   subclaseAvion4 = D85
|   |   |   asignadaEscl <= 0: 2 (8.0/1.0)
|   |   |   asignadaEscl > 0: 0 (11.0/4.0)
|   |   subclaseAvion4 = D86
|   |   |   asigPark3 <= 0
|   |   |   |   tiempoSalirAvionPark3 <= 245: 2 (5.0)
|   |   |   |   tiempoSalirAvionPark3 > 245: 0 (2.0)
|   |   |   asigPark3 > 0: 0 (8.0/1.0)
|   |   subclaseAvion4 = D8A: 2 (15.0/3.0)
|   |   subclaseAvion4 = D8C
|   |   |   tiempoSalirAvionPark7 <= 228: 2 (4.0)
|   |   |   tiempoSalirAvionPark7 > 228: 0 (2.0)
|   |   subclaseAvion4 = D8P
|   |   |   tiempoSalirAvionPark14 <= 269
|   |   |   |   tiempoSalirAvionPark14 <= 25: 2 (3.0)
|   |   |   |   tiempoSalirAvionPark14 > 25: 0 (11.0)
|   |   |   tiempoSalirAvionPark14 > 269: 2 (4.0)
|   |   subclaseAvion4 = D8U: 0 (20.0/10.0)
|   |   subclaseAvion4 = D8X
|   |   |   asigPark3 <= 0: 0 (10.0)
|   |   |   asigPark3 > 0: 2 (10.0/4.0)
|   |   subclaseAvion4 = D92
|   |   |   distAvionPark <= 443: 2 (6.0)
|   |   |   distAvionPark > 443: 0 (10.0/2.0)
|   |   subclaseAvion4 = D93: 0 (11.0/3.0)
|   |   subclaseAvion4 = D94: 2 (19.0/9.0)
|   |   subclaseAvion4 = D95
|   |   |   asigPark7 <= 0
|   |   |   |   asigPark3 <= 0
|   |   |   |   |   tiempoSalirAvionPark3 <= 54: 0 (2.0)
|   |   |   |   |   tiempoSalirAvionPark3 > 54: 2 (2.0)
|   |   |   |   asigPark3 > 0: 0 (5.0)
|   |   |   asigPark7 > 0
|   |   |   |   tiempoSalirAvionPark11 <= 148: 0 (2.0)
|   |   |   |   tiempoSalirAvionPark11 > 148: 2 (5.0)
|   |   subclaseAvion4 = DAM: 0 (16.0/7.0)
|   |   subclaseAvion4 = DC3
|   |   |   tipoEscl = 1 metro: 0 (1.0)
|   |   |   tipoEscl = 1.50 metros: 2 (2.0)
|   |   |   tipoEscl = 2 metros: 0 (4.0/1.0)
|   |   |   tipoEscl = 2.50 metros
|   |   |   |   asigPark4 <= 0: 0 (2.0)
|   |   |   |   asigPark4 > 0: 2 (3.0)
|   |   |   tipoEscl = 3.50 metros: 0 (1.0)
|   |   |   tipoEscl = 4 metros: 0 (2.0/1.0)
```

```
|   |   |   tipoEsc1 = 3 metros: 2 (2.0)
|   |   subclaseAvion4 = DC4
|   |   |   tipoEsc2 = 1 metro: 2 (0.0)
|   |   |   tipoEsc2 = 1.50 metros: 2 (1.0)
|   |   |   tipoEsc2 = 2 metros: 2 (5.0)
|   |   |   tipoEsc2 = 2.50 metros: 2 (2.0)
|   |   |   tipoEsc2 = 3.50 metros: 0 (2.0)
|   |   |   tipoEsc2 = 4 metros: 0 (2.0)
|   |   |   tipoEsc2 = 3 metros: 2 (0.0)
|   |   subclaseAvion4 = DF1: 2 (23.0/11.0)
|   |   subclaseAvion4 = DF2
|   |   |   asigPark11 <= 0
|   |   |   |   asignadaEsc2 <= 0
|   |   |   |   |   distAvionPark <= 1199: 2 (5.0)
|   |   |   |   |   distAvionPark > 1199: 0 (2.0)
|   |   |   |   asignadaEsc2 > 0: 0 (2.0)
|   |   |   asigPark11 > 0: 0 (5.0)
|   |   subclaseAvion4 = DF5: 0 (13.0/5.0)
|   |   subclaseAvion4 = DF9
|   |   |   tiempoSalirAvionPark4 <= 165: 0 (6.0)
|   |   |   tiempoSalirAvionPark4 > 165
|   |   |   |   asigPark14 <= 0: 2 (5.0)
|   |   |   |   asigPark14 > 0
|   |   |   |   |   distAvionPark <= 947: 2 (2.0)
|   |   |   |   |   distAvionPark > 947: 0 (2.0)
|   |   subclaseAvion4 = DH1: 0 (17.0/8.0)
|   |   subclaseAvion4 = DH3
|   |   |   tiempoSalirAvionPark4 <= 252: 2 (17.0/3.0)
|   |   |   tiempoSalirAvionPark4 > 252: 0 (5.0)
|   |   subclaseAvion4 = DHT
|   |   |   tiempoSalirAvionPark14 <= 221
|   |   |   |   asignadaEsc1 <= 0: 2 (7.0)
|   |   |   |   asignadaEsc1 > 0
|   |   |   |   |   tiempoSalirAvionPark7 <= 202: 2 (3.0)
|   |   |   |   |   tiempoSalirAvionPark7 > 202: 0 (3.0)
|   |   |   tiempoSalirAvionPark14 > 221: 0 (6.0)
|   |   subclaseAvion4 = DO2
|   |   |   tiempoSalirAvionPark12 <= 121: 2 (5.0)
|   |   |   tiempoSalirAvionPark12 > 121: 0 (5.0)
|   |   subclaseAvion4 = E70: 2 (17.0/7.0)
|   |   subclaseAvion4 = E75: 2 (19.0/7.0)
|   |   subclaseAvion4 = EM1: 0 (16.0/8.0)
|   |   subclaseAvion4 = EM2
|   |   |   asigPark3 <= 0: 2 (5.0)
|   |   |   asigPark3 > 0
|   |   |   |   tiempoSalirAvionPark5 <= 157: 2 (5.0/1.0)
|   |   |   |   tiempoSalirAvionPark5 > 157: 0 (5.0)
|   |   subclaseAvion4 = EMB
|   |   |   asigPark11 <= 0: 0 (4.0)
|   |   |   asigPark11 > 0
|   |   |   |   distAvionPark <= 1036: 2 (5.0)
|   |   |   |   distAvionPark > 1036: 0 (2.0)
|   |   subclaseAvion4 = ER3: 2 (24.0/11.0)
|   |   subclaseAvion4 = ER4: 2 (21.0/8.0)
|   |   subclaseAvion4 = F10: 0 (18.0/8.0)
|   |   subclaseAvion4 = F21
|   |   |   asigPark14 <= 0: 2 (10.0)
```

```
|   |   |   asigPark14 > 0
|   |   |   |   tiempoSalirAvionPark3 <= 172: 2 (8.0/1.0)
|   |   |   |   tiempoSalirAvionPark3 > 172: 0 (5.0)
|   |   subclaseAvion4 = F22
|   |   |   asigPark11 <= 0: 2 (11.0/3.0)
|   |   |   asigPark11 > 0: 0 (8.0/1.0)
|   |   subclaseAvion4 = F24: 2 (20.0/9.0)
|   |   subclaseAvion4 = F26: 2 (20.0/9.0)
|   |   subclaseAvion4 = F2A
|   |   |   asigPark11 <= 0: 2 (8.0/1.0)
|   |   |   asigPark11 > 0: 0 (16.0/5.0)
|   |   subclaseAvion4 = F2E
|   |   |   asignadaEsc1 <= 0
|   |   |   |   asignadaEsc2 <= 0: 0 (3.0)
|   |   |   |   asignadaEsc2 > 0: 2 (3.0/1.0)
|   |   |   asignadaEsc1 > 0: 2 (3.0)
|   |   subclaseAvion4 = F2F
|   |   |   asigPark11 <= 0: 0 (5.0)
|   |   |   asigPark11 > 0
|   |   |   |   asigPark14 <= 0: 2 (4.0)
|   |   |   |   asigPark14 > 0
|   |   |   |   |   distParkEsc1 <= 31: 2 (2.0)
|   |   |   |   |   distParkEsc1 > 31: 0 (6.0)
|   |   subclaseAvion4 = F2S: 2 (16.0/4.0)
|   |   subclaseAvion4 = F50: 0 (18.0/8.0)
|   |   subclaseAvion4 = F70
|   |   |   tipoEsc2 = 1 metro: 2 (1.0)
|   |   |   tipoEsc2 = 1.50 metros: 2 (5.0/1.0)
|   |   |   tipoEsc2 = 2 metros: 0 (0.0)
|   |   |   tipoEsc2 = 2.50 metros: 0 (4.0)
|   |   |   tipoEsc2 = 3.50 metros: 0 (0.0)
|   |   |   tipoEsc2 = 4 metros: 0 (2.0/1.0)
|   |   |   tipoEsc2 = 3 metros: 0 (5.0/1.0)
|   |   subclaseAvion4 = GR1
|   |   |   tipoEsc0 = 1 metro: 2 (0.0)
|   |   |   tipoEsc0 = 1.50 metros
|   |   |   |   asigPark3 <= 0: 2 (3.0)
|   |   |   |   asigPark3 > 0: 0 (2.0)
|   |   |   tipoEsc0 = 2 metros: 0 (2.0)
|   |   |   tipoEsc0 = 2.50 metros: 2 (3.0)
|   |   |   tipoEsc0 = 3.50 metros: 0 (4.0/1.0)
|   |   |   tipoEsc0 = 4 metros: 2 (3.0)
|   |   |   tipoEsc0 = 3 metros: 2 (2.0)
|   |   subclaseAvion4 = GRC
|   |   |   distParkEsc0 <= 110: 0 (5.0)
|   |   |   distParkEsc0 > 110: 2 (5.0/1.0)
|   |   subclaseAvion4 = GRJ: 0 (20.0/9.0)
|   |   subclaseAvion4 = H25
|   |   |   asigPark6 <= 0: 0 (6.0/1.0)
|   |   |   asigPark6 > 0
|   |   |   |   tiempoSalirAvionPark5 <= 45: 0 (2.0)
|   |   |   |   tiempoSalirAvionPark5 > 45: 2 (4.0)
|   |   subclaseAvion4 = HS1
|   |   |   asigPark5 <= 0: 0 (5.0)
|   |   |   asigPark5 > 0
|   |   |   |   tiempoSalirAvionPark13 <= 60: 2 (4.0)
|   |   |   |   tiempoSalirAvionPark13 > 60: 0 (6.0/1.0)
```

```
|   |   subclaseAvion4 = HS2
|   |   |   asigPark14 <= 0
|   |   |   |   tiempoSalirAvionPark4 <= 198: 2 (5.0)
|   |   |   |   tiempoSalirAvionPark4 > 198: 0 (2.0)
|   |   |   asigPark14 > 0: 0 (8.0/1.0)
|   |   subclaseAvion4 = I93
|   |   |   tipoEsc2 = 1 metro: 2 (2.0)
|   |   |   tipoEsc2 = 1.50 metros: 2 (4.0/1.0)
|   |   |   tipoEsc2 = 2 metros
|   |   |   |   tiempoSalirAvionPark12 <= 92: 2 (2.0)
|   |   |   |   tiempoSalirAvionPark12 > 92: 0 (2.0)
|   |   |   tipoEsc2 = 2.50 metros: 0 (0.0)
|   |   |   tipoEsc2 = 3.50 metros: 0 (2.0)
|   |   |   tipoEsc2 = 4 metros: 0 (2.0)
|   |   |   tipoEsc2 = 3 metros: 0 (0.0)
|   |   subclaseAvion4 = IL6
|   |   |   tiempoSalirAvionPark13 <= 113: 2 (10.0/3.0)
|   |   |   tiempoSalirAvionPark13 > 113: 0 (9.0)
|   |   subclaseAvion4 = IL7
|   |   |   tiempoSalirAvionPark14 <= 145: 0 (7.0)
|   |   |   tiempoSalirAvionPark14 > 145: 2 (7.0/1.0)
|   |   subclaseAvion4 = IL8: 0 (21.0/8.0)
|   |   subclaseAvion4 = IL9
|   |   |   asignadaEsc0 <= 0: 0 (7.0)
|   |   |   asignadaEsc0 > 0
|   |   |   |   asigPark14 <= 0: 2 (4.0)
|   |   |   |   asigPark14 > 0
|   |   |   |   |   asigPark3 <= 0: 2 (3.0/1.0)
|   |   |   |   |   asigPark3 > 0: 0 (5.0)
|   |   subclaseAvion4 = ILW: 0 (12.0/6.0)
|   |   subclaseAvion4 = L11
|   |   |   tipoEsc1 = 1 metro: 2 (1.0)
|   |   |   tipoEsc1 = 1.50 metros: 0 (4.0/1.0)
|   |   |   tipoEsc1 = 2 metros: 0 (6.0/1.0)
|   |   |   tipoEsc1 = 2.50 metros: 2 (2.0)
|   |   |   tipoEsc1 = 3.50 metros: 2 (1.0)
|   |   |   tipoEsc1 = 4 metros: 0 (2.0)
|   |   |   tipoEsc1 = 3 metros: 2 (5.0/1.0)
|   |   subclaseAvion4 = L15
|   |   |   tiempoSalirAvionPark14 <= 117
|   |   |   |   tiempoSalirAvionPark13 <= 127: 2 (5.0)
|   |   |   |   tiempoSalirAvionPark13 > 127: 0 (2.0)
|   |   |   tiempoSalirAvionPark14 > 117: 0 (9.0)
|   |   subclaseAvion4 = L1A
|   |   |   asignadaEsc2 <= 0: 2 (5.0)
|   |   |   asignadaEsc2 > 0: 0 (7.0/1.0)
|   |   subclaseAvion4 = L0F: 0 (9.0/1.0)
|   |   subclaseAvion4 = L0H: 0 (20.0/8.0)
|   |   subclaseAvion4 = LRJ: 0 (24.0/9.0)
|   |   subclaseAvion4 = LT4: 2 (14.0/5.0)
|   |   subclaseAvion4 = M11
|   |   |   tiempoSalirAvionPark11 <= 171: 2 (6.0/1.0)
|   |   |   tiempoSalirAvionPark11 > 171: 0 (9.0)
|   |   subclaseAvion4 = M1F: 0 (22.0/8.0)
|   |   subclaseAvion4 = M81: 0 (19.0/6.0)
|   |   subclaseAvion4 = M82: 0 (15.0/6.0)
```

```
|   |   subclaseAvion4 = M83
|   |   |   tiempoSalirAvionPark11 <= 136: 0 (4.0/1.0)
|   |   |   tiempoSalirAvionPark11 > 136: 2 (6.0)
|   |   subclaseAvion4 = M87
|   |   |   asigPark11 <= 0: 0 (6.0)
|   |   |   asigPark11 > 0
|   |   |   |   asigPark4 <= 0: 2 (8.0/1.0)
|   |   |   |   asigPark4 > 0
|   |   |   |   |   distParkEsc1 <= 36: 2 (2.0)
|   |   |   |   |   distParkEsc1 > 36: 0 (5.0)
|   |   subclaseAvion4 = M88: 0 (20.0/9.0)
|   |   subclaseAvion4 = M8S
|   |   |   asignadaEsc2 <= 0
|   |   |   |   asigPark12 <= 0
|   |   |   |   |   tiempoSalirAvionPark5 <= 81: 0 (4.0)
|   |   |   |   |   tiempoSalirAvionPark5 > 81: 2 (6.0/1.0)
|   |   |   |   asigPark12 > 0: 0 (3.0)
|   |   |   asignadaEsc2 > 0: 2 (5.0)
|   |   subclaseAvion4 = M90: 0 (15.0/6.0)
|   |   subclaseAvion4 = M93
|   |   |   asigPark5 <= 0: 2 (5.0)
|   |   |   asigPark5 > 0: 0 (8.0/1.0)
|   |   subclaseAvion4 = NDC
|   |   |   asigPark12 <= 0: 0 (7.0/1.0)
|   |   |   asigPark12 > 0: 2 (6.0/1.0)
|   |   subclaseAvion4 = PA2: 0 (25.0/10.0)
|   |   subclaseAvion4 = PAS
|   |   |   asigPark3 <= 0
|   |   |   |   asigPark12 <= 0: 2 (7.0)
|   |   |   |   asigPark12 > 0: 0 (3.0)
|   |   |   asigPark3 > 0
|   |   |   |   tiempoSalirAvionPark13 <= 41: 2 (2.0)
|   |   |   |   tiempoSalirAvionPark13 > 41: 0 (11.0)
|   |   subclaseAvion4 = S20: 2 (20.0/8.0)
|   |   subclaseAvion4 = SF3: 0 (16.0/4.0)
|   |   subclaseAvion4 = SH5
|   |   |   tiempoSalirAvionPark4 <= 184: 0 (15.0)
|   |   |   tiempoSalirAvionPark4 > 184: 2 (2.0)
|   |   subclaseAvion4 = SSC
|   |   |   distParkEsc2 <= 69: 2 (3.0)
|   |   |   distParkEsc2 > 69: 0 (12.0/1.0)
|   |   subclaseAvion4 = SW4
|   |   |   distParkEsc1 <= 81: 0 (7.0)
|   |   |   distParkEsc1 > 81: 2 (7.0/1.0)
|   |   subclaseAvion4 = SWM: 0 (20.0/10.0)
|   |   subclaseAvion4 = T20: 2 (17.0/3.0)
|   |   subclaseAvion4 = TU3: 0 (15.0/6.0)
|   |   subclaseAvion4 = TU5
|   |   |   tipoEsc2 = 1 metro: 2 (4.0)
|   |   |   tipoEsc2 = 1.50 metros: 2 (1.0)
|   |   |   tipoEsc2 = 2 metros: 0 (2.0/1.0)
|   |   |   tipoEsc2 = 2.50 metros: 0 (5.0/1.0)
|   |   |   tipoEsc2 = 3.50 metros: 0 (2.0)
|   |   |   tipoEsc2 = 4 metros: 0 (2.0/1.0)
|   |   |   tipoEsc2 = 3 metros: 2 (3.0)
```

```
|   |   subclaseAvion4 = XXX
|   |   |   asignadaEsc1 <= 0
|   |   |   |   tiempoSalirAvionPark13 <= 155: 0 (8.0/1.0)
|   |   |   |   tiempoSalirAvionPark13 > 155: 2 (4.0)
|   |   |   asignadaEsc1 > 0: 0 (10.0)
|   |   subclaseAvion4 = YK2
|   |   |   tiempoSalirAvionPark12 <= 52: 0 (6.0)
|   |   |   tiempoSalirAvionPark12 > 52
|   |   |   |   tiempoSalirAvionPark6 <= 202: 2 (10.0)
|   |   |   |   tiempoSalirAvionPark6 > 202
|   |   |   |   |   tiempoSalirAvionPark4 <= 171: 0 (3.0)
|   |   |   |   |   tiempoSalirAvionPark4 > 171: 2 (2.0)
|   |   subclaseAvion4 = YK4: 0 (20.0/10.0)
|   |   subclaseAvion4 = no hay: 0 (0.0)
|   tiempoSalirAvionPark13 > 200: -1 (1856.0)
asigPark13 > 0: -1 (5541.0)

Number of Leaves  :     493

Size of the tree :  709
```

Ilustración 48: Reglas de zona de estacionamiento remota 13 en prueba 5

Aparte de la gran cantidad de reglas generadas en esta prueba por cada zona de estacionamiento, se puede comprobar al analizarlas que no siempre representan el razonamiento seguido en la toma de decisiones y, de ahí, que el porcentaje de error sea mayor que en otras pruebas. De todos modos, para esta prueba el porcentaje de aciertos (83%) se puede considerar alto a pesar de la complejidad en la asignación y cancelación de escalerillas.

6. Historia del proyecto

Este capítulo da información sobre los avances del proyecto respecto al periodo de tiempo en el que se enmarca, así como el coste total de la realización del mismo.

6.1. Plan de trabajo

En este apartado se definen las fases en las que se ha dividido el proyecto, así como el tiempo planificado para cada una de estas fases y el tiempo real que han durado finalmente las mismas. Toda esta información es de utilidad para calcular un precio fiable sobre el producto.

En primer lugar se presenta la línea temporal en la que se embarca todo el proyecto, el cuál comenzó el día 17 de Mayo de 2011 y ha finalizado el 30 de Abril de 2012.

Al realizar la estimación del desarrollo del proyecto, se dividió éste en las siguientes 5 fases o tareas: viabilidad, análisis, diseño, implementación y pruebas.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Viabilidad**	**74 días**	**mar 17/05/11**	**vie 26/08/11**	
26	**Análisis**	**55 días**	**lun 29/08/11**	**vie 11/11/11**	
32	**Diseño**	**35 días**	**lun 14/11/11**	**vie 30/12/11**	**26**
42	**Implementación**	**50 días**	**lun 02/01/12**	**vie 09/03/12**	**32**
48	**Pruebas**	**20 días**	**lun 12/03/12**	**vie 06/04/12**	**42**

Ilustración 49: Fases del plan de trabajo

6.1.1. Viabilidad

Esta tarea es la inicial del proyecto y está formado por las siguientes subtareas:

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Viabilidad**	**74 días**	**mar 17/05/11**	**vie 26/08/11**	
2	Alcance del sistema	4 días	mar 17/05/11	vie 20/05/11	
3	**Profundizar dominio**	**10 días**	**lun 23/05/11**	**vie 03/06/11**	
7	**Similitud con otros sistemas**	**10 días**	**lun 06/06/11**	**vie 17/06/11**	
10	**Algoritmos extracción conocimiento**	**15 días**	**lun 20/06/11**	**vie 08/07/11**	
15	**Algoritmo generación reglas**	**15 días**	**lun 11/07/11**	**vie 29/07/11**	
20	**Estudio de herramientas**	**10 días**	**lun 01/08/11**	**vie 12/08/11**	
25	Documentación estado del arte	10 días	lun 15/08/11	vie 26/08/11	

Ilustración 50: Subtareas de la fase de viabilidad

6.1.1.1. Alcance del sistema

Se detalla y se define de manera clara en qué va a consistir el proyecto y el objetivo final del mismo.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
2	Alcance del sistema	4 días	mar 17/05/11	vie 20/05/11	

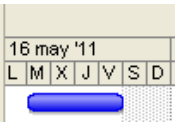


Ilustración 51: Subtarea de alcance del sistema

6.1.1.2. Profundizar en el dominio

Se consigue una serie de conocimientos sobre los aeropuertos y palabras técnicas del este entorno mediante la lectura de libros y búsquedas en internet.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
3	**Profundizar dominio**	**10 días**	**lun 23/05/11**	**vie 03/06/11**	
4	Conocimiento general entorno aeroportuario	4 días	lun 23/05/11	jue 26/05/11	
5	Conocimiento específico servicios aeropuerto	6 días	jue 26/05/11	jue 02/06/11	
6	Documentar motivación y objetivos	2 días	jue 02/06/11	vie 03/06/11	

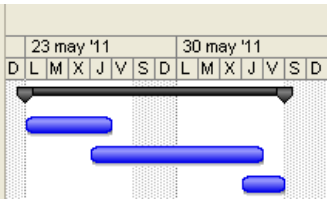


Ilustración 52: Subtarea de profundizar en el dominio

6.1.1.3. Similitud con otros sistemas

Se estudian sistemas ya existentes con una funcionalidad similar o parecida al que se desea construir.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
7	**Similitud con otros sistemas**	**10 días**	**lun 06/06/11**	**vie 17/06/11**	
8	Recopilar información sistemas expertos parecidos	5 días	lun 06/06/11	vie 10/06/11	
9	Análisis sistemas expertos parecidos	5 días	lun 13/06/11	vie 17/06/11	8

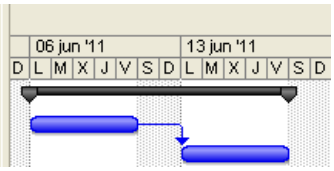


Ilustración 53: Subtarea de similitud con otros sistemas

6.1.1.4. Algoritmos de extracción del conocimiento

Se estudian algoritmos o técnicas de extracción del conocimiento, se analizan y se elige uno de los mismos para ser profundamente analizado y usado en el proyecto.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
10	**Algoritmos extracción conocimiento**	**15 días**	**lun 20/06/11**	**vie 08/07/11**	
11	Recopilar información algoritmos extracción	5 días	lun 20/06/11	vie 24/06/11	
12	Análisis algoritmos extracción	3 días	lun 27/06/11	mié 29/06/11	11
13	Selección algoritmo extracción	1 día	jue 30/06/11	jue 30/06/11	12
14	Análisis exhaustivo algoritmo	4 días	mar 05/07/11	vie 08/07/11	13

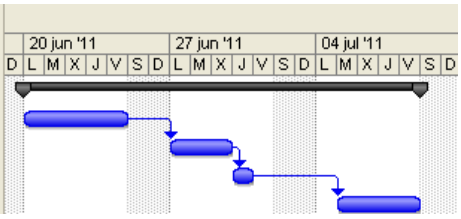


Ilustración 54: Subtarea de algoritmos de extracción del conocimiento

6.1.1.5. Algoritmos de generación de reglas

Se estudian algoritmos de generación de reglas, se analizan y se elige uno de los mismos para ser profundamente analizado y usado en el proyecto.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
15	**Algoritmo generación reglas**	**15 días**	**lun 11/07/11**	**vie 29/07/11**	
16	Recopilar información algoritmo generación reglas	4 días	lun 11/07/11	jue 14/07/11	
17	Análisis algoritmo generación reglas	4 días	vie 15/07/11	mié 20/07/11	16
18	Selección algoritmo generación reglas	1 día	jue 21/07/11	jue 21/07/11	17
19	Análisis exhaustivo generación reglas	6 días	vie 22/07/11	vie 29/07/11	18

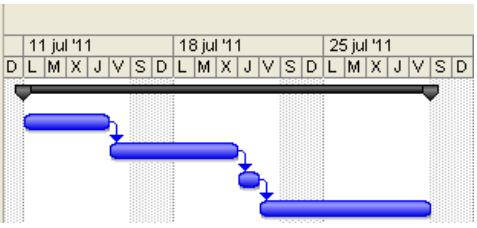


Ilustración 55: Subtarea de algoritmos de generación de reglas

6.1.1.6. Estudio de herramientas

Se estudian las posibles plataformas de desarrollo así como distintas herramientas que pueden ser de utilidad en la construcción del sistema.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
20	**Estudio de herramientas**	**10 días**	**lun 01/08/11**	**vie 12/08/11**	
21	Búsqueda de herramientas	4 días	lun 01/08/11	jue 04/08/11	
22	Análisis herramientas	3 días	vie 05/08/11	mar 09/08/11	21
23	Selección de herramientas	1 día	mié 10/08/11	mié 10/08/11	22
24	Configuración de herramientas	2 días	jue 11/08/11	vie 12/08/11	23

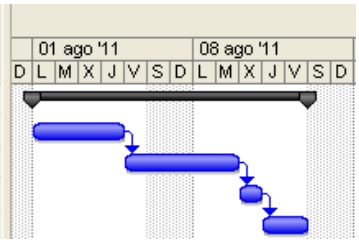


Ilustración 56: Subtarea de estudio de herramientas

6.1.1.7. Documentación del estado del arte

Con toda la documentación recopilada, se lleva a cabo la escritura del estado del arte del proyecto.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
25	Documentación estado del arte	10 días	lun 15/08/11	vie 26/08/11	

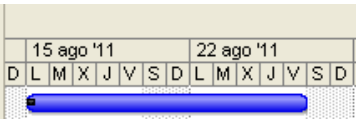
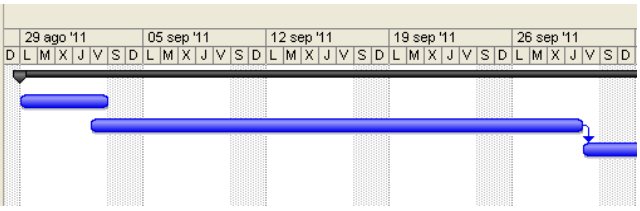


Ilustración 57: Subtarea de documentación del estado del arte

6.1.2. Análisis

En esta tarea se identifican los usuarios que van a usar el sistema, se crea una lista con los requisitos del mismo, se produce la validación de dichos requisitos por parte del cliente, se analizan los casos de uso existentes y se documenta todo ello.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
26	**Análisis**	**55 días**	**lun 29/08/11**	**vie 11/11/11**	
27	Identificacion usuarios	5 días	lun 29/08/11	vie 02/09/11	
28	Establecer requisitos	20 días	vie 02/09/11	jue 29/09/11	
29	Validacion requisitos	15 días	vie 30/09/11	jue 20/10/11	28
30	Casos de uso	10 días	jue 20/10/11	mié 02/11/11	
31	Documentación análisis	7 días	jue 03/11/11	vie 11/11/11	



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
26	**Análisis**	**55 días**	**lun 29/08/11**	**vie 11/11/11**	
27	Identificacion usuarios	5 días	lun 29/08/11	vie 02/09/11	
28	Establecer requisitos	20 días	vie 02/09/11	jue 29/09/11	
29	Validacion requisitos	15 días	vie 30/09/11	jue 20/10/11	28
30	Casos de uso	10 días	jue 20/10/11	mié 02/11/11	
31	Documentación análisis	7 días	jue 03/11/11	vie 11/11/11	

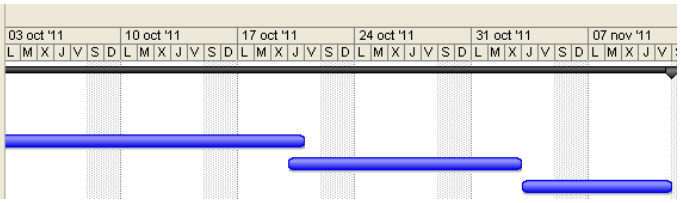


Ilustración 58 (a y b): Fase de análisis

6.1.3. Diseño

En esta tarea se analizan los subsistemas y la arquitectura del sistema, se realiza un diagrama de clases, se analiza el modelo de datos y las interfaces de usuario, y se documenta todo ello.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
32	**Diseño**	**35 días**	**lun 14/11/11**	**vie 30/12/11**	26
33	Subsistemas del sistema	6 días	lun 14/11/11	lun 21/11/11	
34	Arquitectura del sistema	5 días	lun 21/11/11	vie 25/11/11	
35	Diagramas de secuencia del sistema	3 días	lun 28/11/11	mié 30/11/11	34
36	**Clases del sistema**	**5 días**	**jue 01/12/11**	**mié 07/12/11**	
37	Clases	3 días	jue 01/12/11	lun 05/12/11	
38	Asociaciones	2 días	mar 06/12/11	mié 07/12/11	37
39	Diseño modelo de datos	5 días	jue 08/12/11	mié 14/12/11	
40	Definicion interfaces de usuario	5 días	mié 14/12/11	mar 20/12/11	
41	Documentación diseño	9 días	mar 20/12/11	vie 30/12/11	

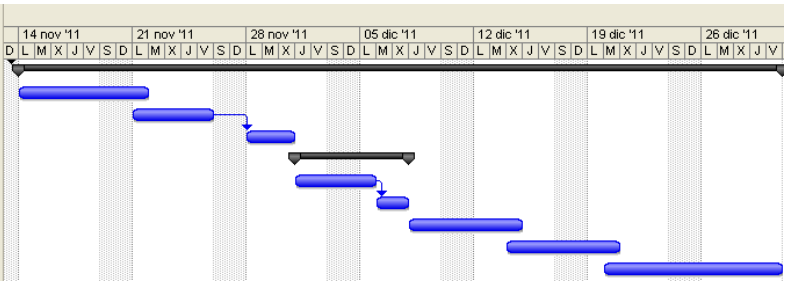
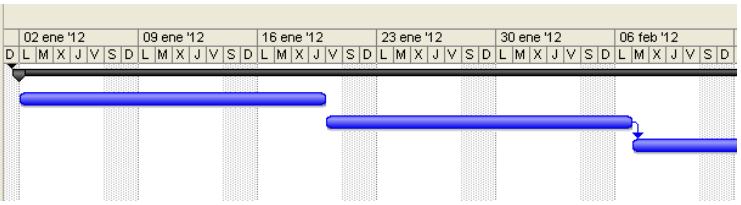


Ilustración 59: Fase de diseño

6.1.4. Implementación

Se escribe el código necesario para hacer la interfaz gráfica del sistema, el motor de extracción de conocimiento, la generación de las reglas oportunas y el modo de asignación automática. Antes de finalizar esta tarea se documenta sobre la implementación de todo ello.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
42	**Implementación**	**50 días**	**lun 02/01/12**	**vie 09/03/12**	32
43	Implementacion interfaz grafica	14 días	lun 02/01/12	jue 19/01/12	
44	Implementacion extraccion conocimiento	12 días	vie 20/01/12	lun 06/02/12	
45	Implementacion generacion reglas	10 días	mar 07/02/12	lun 20/02/12	44
46	Implementacion asignacion automática	7 días	mar 21/02/12	mié 29/02/12	
47	Documentación implementación	7 días	jue 01/03/12	vie 09/03/12	



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
42	**Implementación**	**50 días**	**lun 02/01/12**	**vie 09/03/12**	32
43	Implementacion interfaz grafica	14 días	lun 02/01/12	jue 19/01/12	
44	Implementacion extraccion conocimiento	12 días	vie 20/01/12	lun 06/02/12	
45	Implementacion generacion reglas	10 días	mar 07/02/12	lun 20/02/12	44
46	Implementacion asignacion automática	7 días	mar 21/02/12	mié 29/02/12	
47	Documentación implementación	7 días	jue 01/03/12	vie 09/03/12	

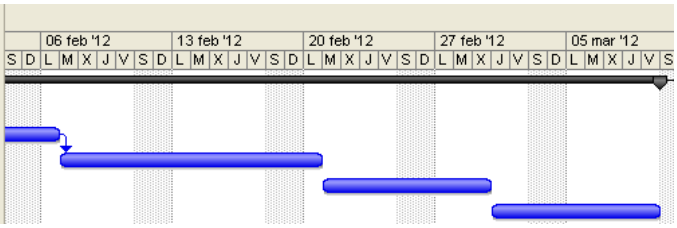
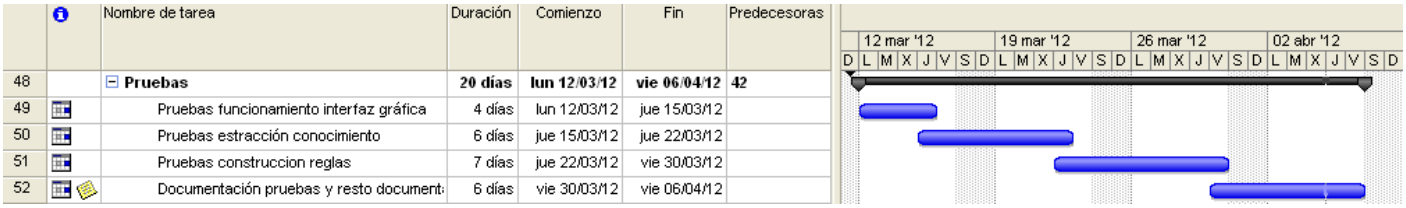


Ilustración 60 (a y b): Fase de implementación

6.1.5. Pruebas

Se realizan pruebas para comprobar el correcto funcionamiento del sistema y que cumple los objetivos marcados al principio del proyecto. Finalmente se termina de escribir la documentación sobre el proyecto.



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
48	**Pruebas**	**20 días**	**lun 12/03/12**	**vie 06/04/12**	**42**
49	Pruebas funcionamiento interfaz gráfica	4 días	lun 12/03/12	jue 15/03/12	
50	Pruebas estracción conocimiento	6 días	jue 15/03/12	jue 22/03/12	
51	Pruebas construccion reglas	7 días	jue 22/03/12	vie 30/03/12	
52	Documentación pruebas y resto document	6 días	vie 30/03/12	vie 06/04/12	

Ilustración 61: Fase de pruebas

En la siguiente tabla se muestra las fechas estimadas y las reales de cada una de las fases del proyecto, permitiendo comprobar si se han cumplido con los plazos.

TAREA	INICIO (ESTIMADO)	INICIO (REAL)		FIN (ESTIMADO)	FIN (REAL)	
Viabilidad	17/05/2011	17/05/2011		26/08/2012	23/08/2011	
Análisis	29/08/2011	24/08/2011		11/11/2011	14/11/2011	
Diseño	14/11/2011	15/11/2011		30/12/2011	05/01/2012	
Implementación	02/01/2012	04/01/2012		09/03/2012	20/03/2012	
Pruebas	09/03/2012	22/03/2012		06/04/2012	30/04/2012	

Tabla 61: Fechas estimadas y reales del proyecto

Para un correcto presupuesto es necesario conocer el esfuerzo detallado por cada tarea, además esto servirá como información de referencia para futuros proyectos de similar índole. La jornada laboral de este proyecto corresponde a 4 horas diarias todos los días excepto fines de semana, por lo que su correspondiente desglose de horas por tarea es mostrado en la siguiente tabla.

TAREA	DÍAS ESTIMADOS	HORAS ESTIMADAS	DÍAS REALES	HORAS REALES
Viabilidad	74 días	296 horas	71 días	284 horas
Análisis	55 días	220 horas	58 días	232 horas
Diseño	35 días	140 horas	38 días	152 horas
Implementación	50 días	200 horas	55 días	220 horas
Pruebas	20 días	80 horas	28 días	112 horas
Total	234 días	936 horas	250 días	1000 horas

Tabla 62: Días y horas estimadas y reales del proyecto

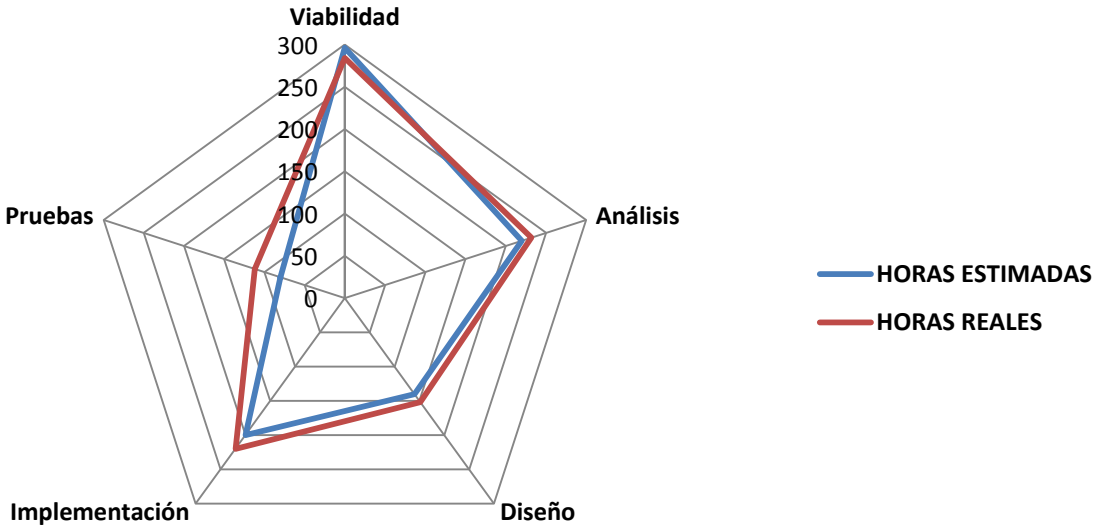


Ilustración 62: Esfuerzo por tarea en horas totales

Como se puede comprobar en el gráfico, la estimación inicial realizada ha sido bastante exacta excepto, quizás, en la tarea de pruebas, donde se preveía un esfuerzo menor del que fue necesario. Esto fue debido a que el sistema experto necesita de un conjunto de ejemplos de entrenamiento bastante elevado para aprender y, por tanto, las pruebas requerían más tiempo del previsto inicialmente.

6.2. Presupuesto

El coste total del proyecto es de 43768 euros, tal y como se muestra en el siguiente desglose de los diferentes gastos del proyecto.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:	Javier Fernández Pérez
2.- Departamento:	Informática, Inteligencia Artificial
3.- Descripción del Proyecto:	
- Titulo	Creación de un simulador aeroportuario. Sistema experto de asignación en el desembarque de pasajeros y generación de reglas a partir del conocimiento
- Duración (meses)	11
Tasa de costes Indirectos:	20%

4.- Presupuesto total del Proyecto (valores en Euros):

Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Fernández Pérez, Javier		Ingeniero	11	2.694,39	29.638,29	
Villena Román, Julio		Ingeniero Senior	0,5	4.289,54	2.144,77	
Borches Juzgado, Pedro Daniel		Ingeniero Senior	1	4.289,54	4.289,54	
					0,00	
					0,00	
		Hombres mes 12,5		Total	36.072,60	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Equipo de trabajo	580,00	100	11	60	106,33
Licencia Office Professional	500,00	100	11	60	91,67
Licencia Visio	330,00	100	11	60	60,50
Licencia Project	775,00	100	11	60	142,08
					0,00
					0,00
				Total	400,58

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	36.073
Amortización	401
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	7.295
Total	43.768

Ilustración 63: Presupuesto del proyecto

7. Conclusiones y trabajos futuros

7.1. Conclusiones

Uno de los aspectos más críticos en todo aeropuerto es la asignación tanto de sus recursos materiales como humanos a los aviones que van aterrizando en el mismo para realizar una serie de servicios (desembarque de pasajeros, transporte de los mismos y del equipaje, etc.) de la forma más óptima posible y que así no se sufran retrasos. Por ello surge la necesidad de crear un sistema experto que sea capaz de captar y representar el conocimiento de los prestadores de estos servicios para poder automatizar este proceso y reducir costes en compensaciones económicas por retrasos o cancelaciones, pero sin perder la fiabilidad.

En este proyecto se ha presentado que los juegos o simulaciones son un medio muy efectivo para poder extraer el conocimiento de los prestadores de estos servicios, debido a la gran cantidad de variables que son tenidas en cuenta por éstos para la toma de decisiones. Además, se ha constatado la importancia de la minería de datos para abordar esta necesidad, tanto en la difícil pero fundamental fase de preprocesamiento de los datos e información recogida como en la fase de entrenamiento, en la que se ha entrenado una serie de árboles de decisión J48 que representan el conjunto de reglas que son aplicadas para asignar y cancelar las asignaciones de escalerillas que ayudan al desembarque de los pasajeros de los aviones. A la vez, se ha constatado que el uso de herramientas que permiten tanto tratar los datos como construir modelos y que son de fácil integración (como la herramienta Weka) ayuda de manera notable al desarrollo del sistema experto.

Sin lugar a dudas, a la vista de las pruebas y de los resultados obtenidos, se ha confirmado que el sistema, por lo general, genera un conjunto de reglas de asignación y cancelación de escalerillas bastante acorde al razonamiento llevado a cabo durante la simulación, aunque, cuanto más complejo es el método de asignación y cancelación, el prestador de los servicios debe estar simulando o entrenando al modelo mayor tiempo para tener más datos y que el porcentaje de aciertos y, por tanto, la fiabilidad no se vea disminuida.

Finalmente, aunque los resultados obtenidos en este proyecto hayan sido bastante satisfactorios, no hay que perder de vista que, a lo largo del desarrollo del mismo, se han ido imponiendo una serie de limitaciones que hace que sea necesario un análisis y estudio posterior y que complemente a éste, para poder conocer aún con mayor precisión hasta dónde se puede llegar en la captación y extracción del conocimiento de un prestador de servicios aeroportuarios.

7.2. Trabajos futuros

Como ya se ha mencionado anteriormente, en este proyecto se han tenido que poner una serie de limitaciones para que el desarrollo del mismo no se extendiera demasiado en el tiempo. Dichas limitaciones convendrían ser estudiadas en futuros trabajos para que, algún día, el proyecto pueda ser vendido a alguna empresa y así sea rentable.

En primer lugar, se deja para el futuro mejorar el simulador aeroportuario de tal forma que se permita asignar varias escalerillas a un único avión y así, en el caso de que fuera un avión grande con una gran cantidad de pasajeros, el desalojo de los mismos fuera mucho más rápido que si sólo se asignara una única escalerilla.

Además, se deja para posteriores trabajos que el simulador aeroportuario construido tenga en cuenta más servicios aeroportuarios (como el transporte de los pasajeros del avión a la terminal y viceversa, la limpieza del avión, etc.) para que el conjunto de reglas generado sea mucho más extenso y abarque también al resto de servicios. Para ello, haría falta que se integraran en el simulador los recursos materiales necesarios para que se lleven a cabo esos servicios, así como delimitar el movimiento de éstos por el aeropuerto simulado.

Otro aspecto bastante interesante sería que se generaran reglas no sólo para asignar y cancelar asignaciones de los recursos materiales de los servicios, sino también de los recursos de personal del servicio, es decir, que el simulador tuviera en cuenta que también hay un personal encargado de mover y manejar los recursos materiales para que los servicios aeroportuarios se lleven a cabo. Por ejemplo, las jardineras que transportan a los pasajeros desde el avión a la terminal y viceversa son conducidas por unos operarios específicos que deben de poseer un carnet que les acredite para conducir estos autobuses. El sistema aprendería qué operarios pueden realizar qué servicios para así poder realizar la asignación de los operarios.

Como ya se ha dicho anteriormente, en este proyecto se ha usado el árbol de decisión J48 para generar el conjunto de reglas, pero existen muchos más modelos (o combinaciones híbridas de los mismos) que podrían servir igual que el mencionado árbol, ya que éstos también generan conjuntos de reglas. Algunos de ellos han sido ya expuestos en el punto “Estado del arte” de este proyecto, así que otra posible investigación futura sería probar con estos modelos para ver si se mejora en algo o no los resultados que se han obtenido usando el árbol de decisión J48.

Otra forma de validar el conjunto de reglas construido, diferente a la que se ha usado en este proyecto y que sería una gran idea el llevarla a cabo en un futuro, es que el simulador mostrara lo que está pasando realmente en el aeropuerto en un momento determinado y, a partir de ese momento, comparar las asignaciones y

cancelaciones de asignaciones que realiza el simulador apoyado en el conjunto de reglas generado y las que se llevan a cabo realmente en el aeropuerto decididas por el coordinador de los servicios aeroportuarios.

Para terminar con los trabajos futuros, se van a nombrar dos mejoras que se podrían desarrollar del simulador para que éste fuera mucho más versátil y adaptable a distintos aeropuertos y situaciones. La primera mejora sería que existiera un archivo de configuración y en este archivo de configuración se indicaran la cantidad y posición de las terminales del aeropuerto que se va a simular, cantidad y posiciones de zonas de estacionamiento, etc., para que así se pudiera simular cualquier aeropuerto, y no sólo una única configuración, como está actualmente. La segunda mejora sería que el modelo de generación de reglas tuviera en cuenta variables que el coordinador de servicios no pudiera controlar pero que sí afectan en su decisión final de qué servicio y qué asignaciones darán comienzo, como son, por ejemplo, el tiempo que hace (si nieva o hace mucho frío es necesario echar anticongelante a los aviones y a la pista) o imprevistos, como que falle algún motor al arrancar el avión y se deba llevar a los pasajeros a otro avión.

Anexos

Manual de configuración del simulador aeroportuario

Para poder arrancar el simulador aeroportuario, primero es necesario comprobar una serie de requisitos y realizar una serie de operaciones para que éste funcione de manera correcta. En primer lugar es necesario verificar que se tiene instalado en el ordenador donde se va a ejecutar el simulador el JRE de Java [38], que es un conjunto de utilidades que permite la ejecución de programas Java. En segundo lugar, si se quiere visualizar y conectarse a la base de datos, es necesario tener instalado Microsoft Access [39]. Otro requisito para la conexión a la base de datos es tener configurado correctamente el controlador ODBC de Microsoft Access. Para ello hay que seguir los siguientes pasos (para un Windows XP).

- Hacer click en: "Inicio" → "Panel de control" → "Herramientas administrativas" → "Orígenes de datos (ODBC)"

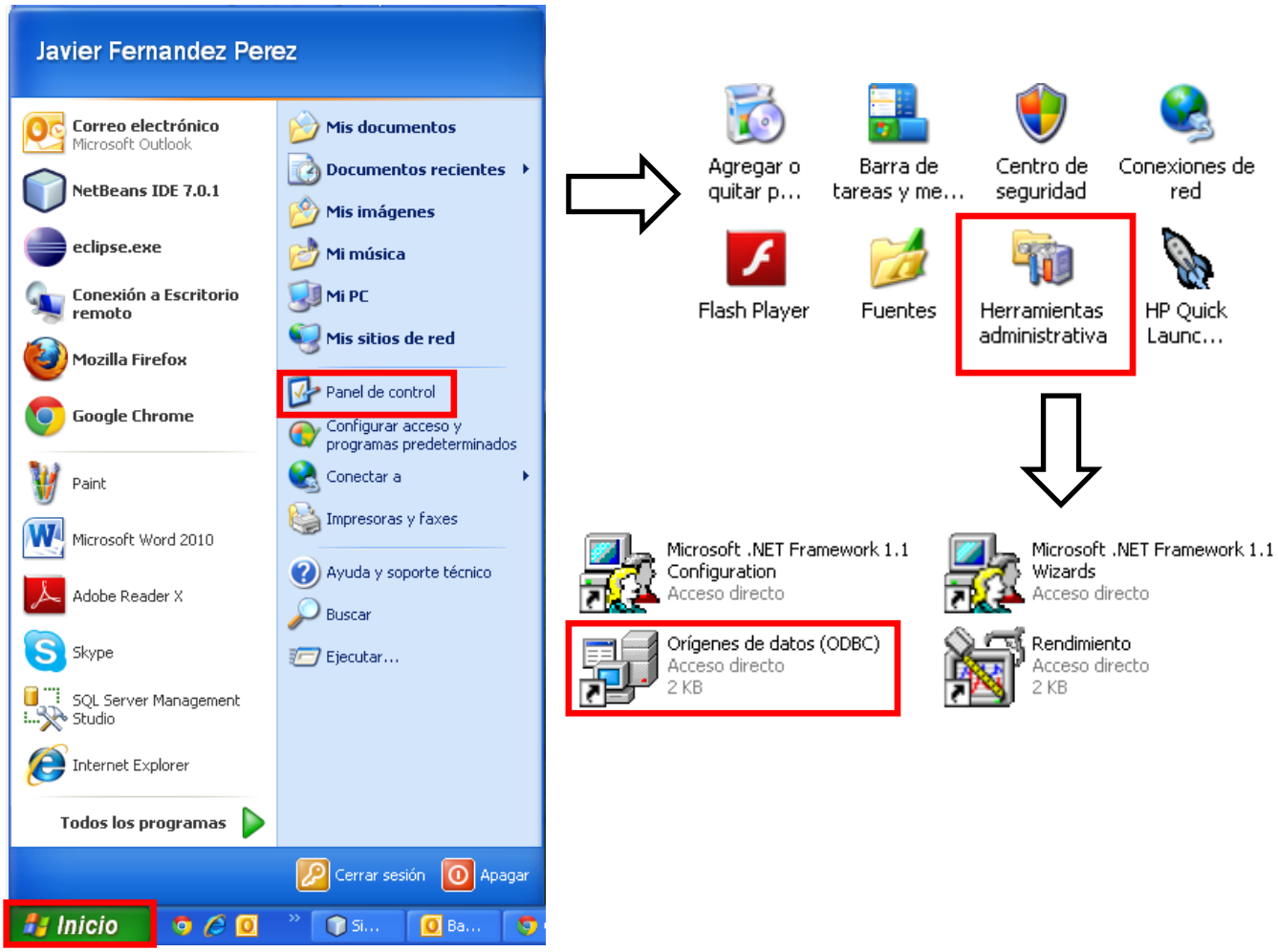


Ilustración 64: Orígenes de datos (ODBC) en Windows XP

- En la ventana que aparece ir a la pestaña de “DNS de usuario” y hacer click en “Agregar”

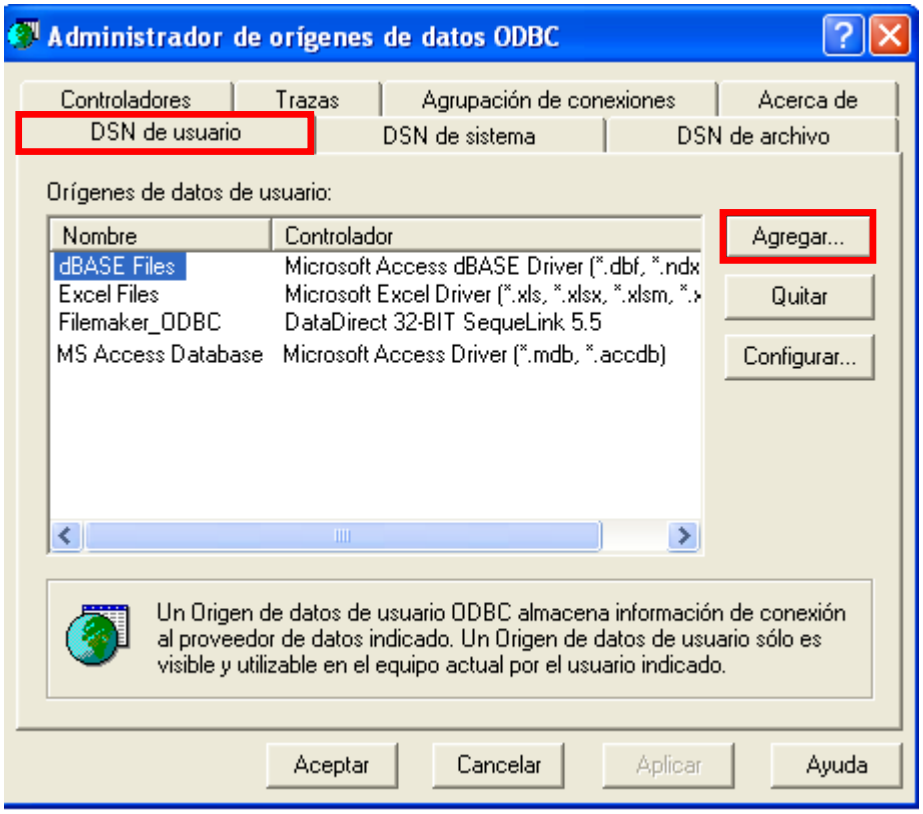


Ilustración 65: Administrador de orígenes de datos ODBC

- En la nueva ventana seleccionar el “Driver o Controlador de Microsoft Access (*.mdb)” y hacer click en “Finalizar”

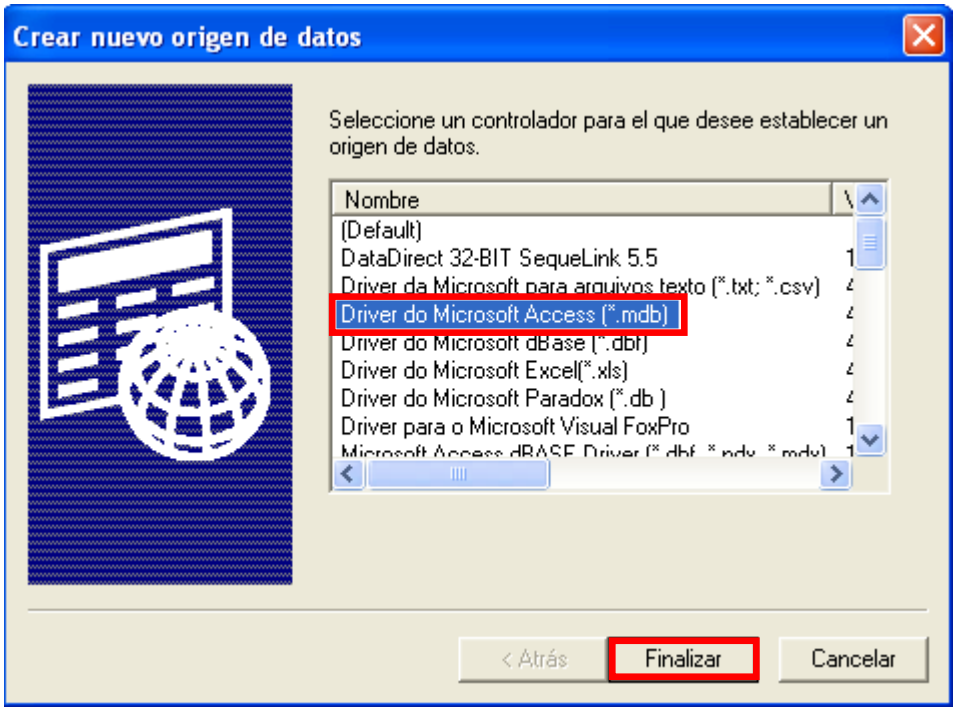


Ilustración 66: Crear nuevo origen de datos en Windows XP

- En la nueva ventana que aparece escribir loc_db (sólo se puede llamar así) en la sección de "Nombre del origen de datos" y dar al botón "Seleccionar…"

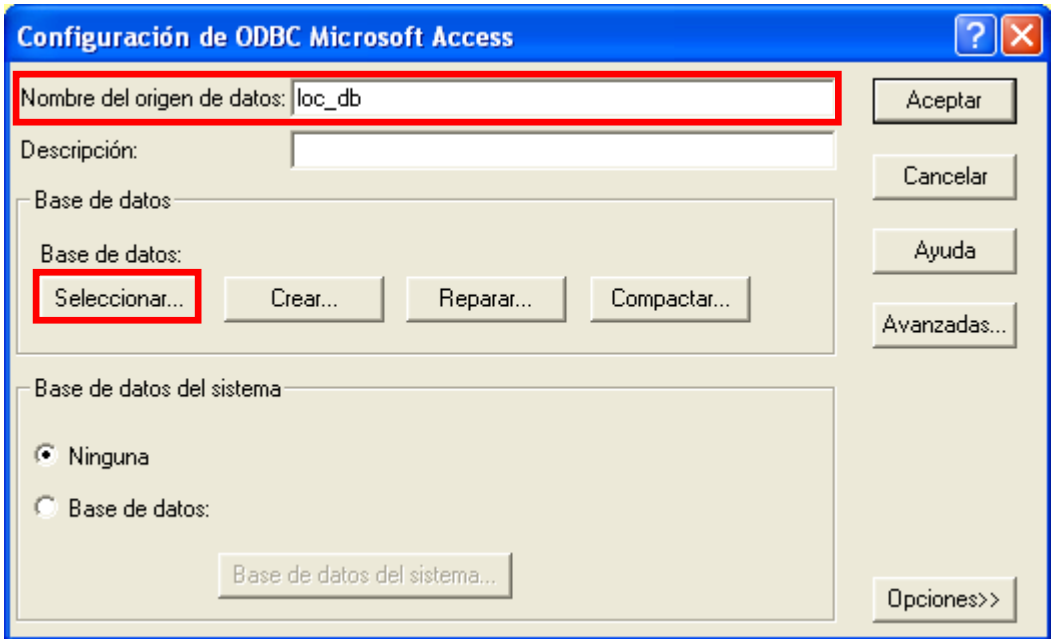


Ilustración 67: Configuración de ODBC Microsoft Access

- Buscar en el directorio de archivos donde esté situado la base de datos del simulador aeroportuario (llamado loc_db.mdb), seleccionarlo y dar al botón "Aceptar"

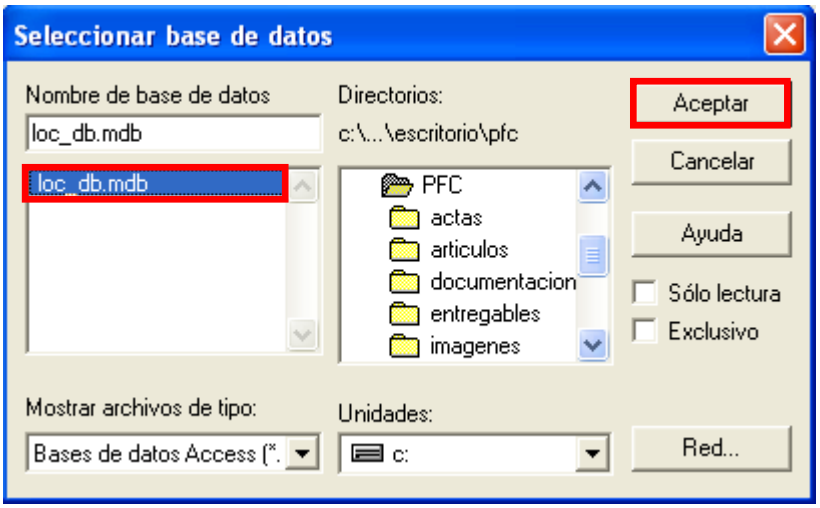


Ilustración 68: Seleccionar base de datos en configuración ODBC

- Se agregará el nuevo origen de datos de usuario ODBC a la lista y finalmente se pulsa en el botón “Aceptar”

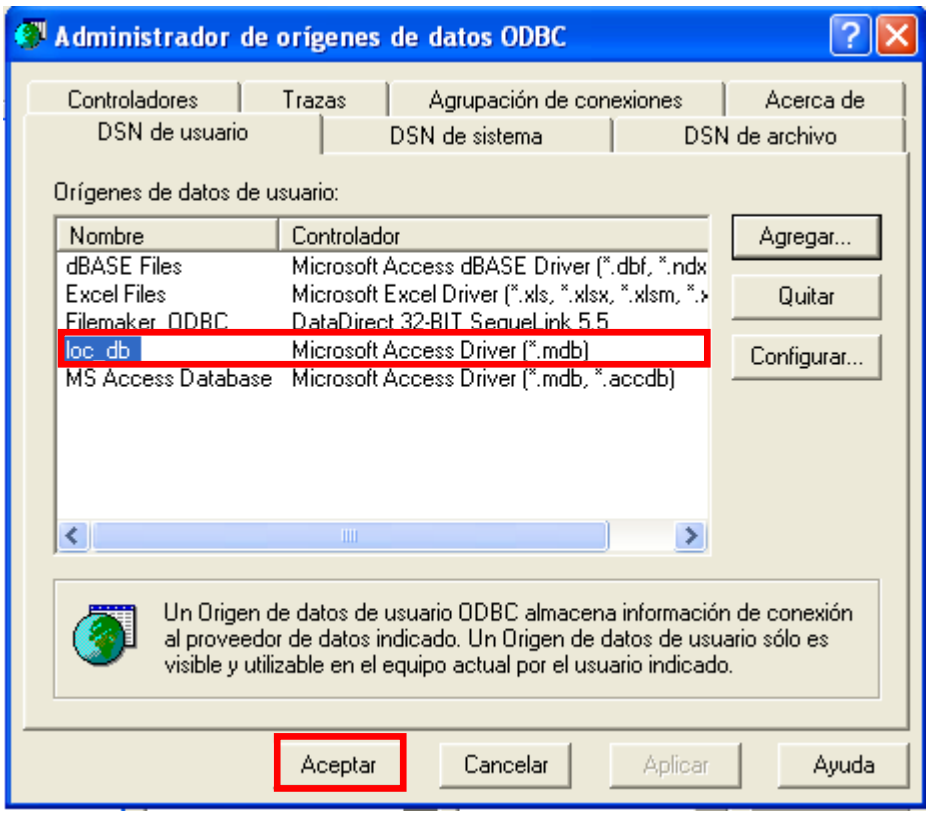


Ilustración 69: Administrador de orígenes de datos ODBC

Con estos pasos ya se está listo para poder arrancar el simulador aeroportuario.

Manual de usuario

Para poder arrancar el simulador aeroportuario (o sistema experto de extracción del conocimiento) se debe ir a la carpeta donde esté tanto la carpeta “lib” como el archivo “SimuladorAeroportuario.jar” y hacer click dos veces sobre este último.

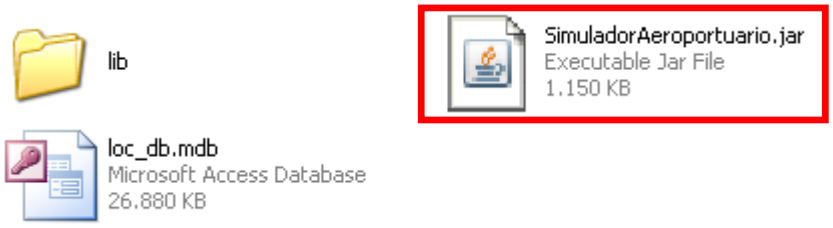


Ilustración 70: Archivos necesarios para arrancar el simulador aeroportuario

Tras esto aparecerá una ventana donde se puede ver en un plano aéreo el aeropuerto simulado y tres pestañas: “Aeropuerto”, “Parámetros” y “Ayuda”.

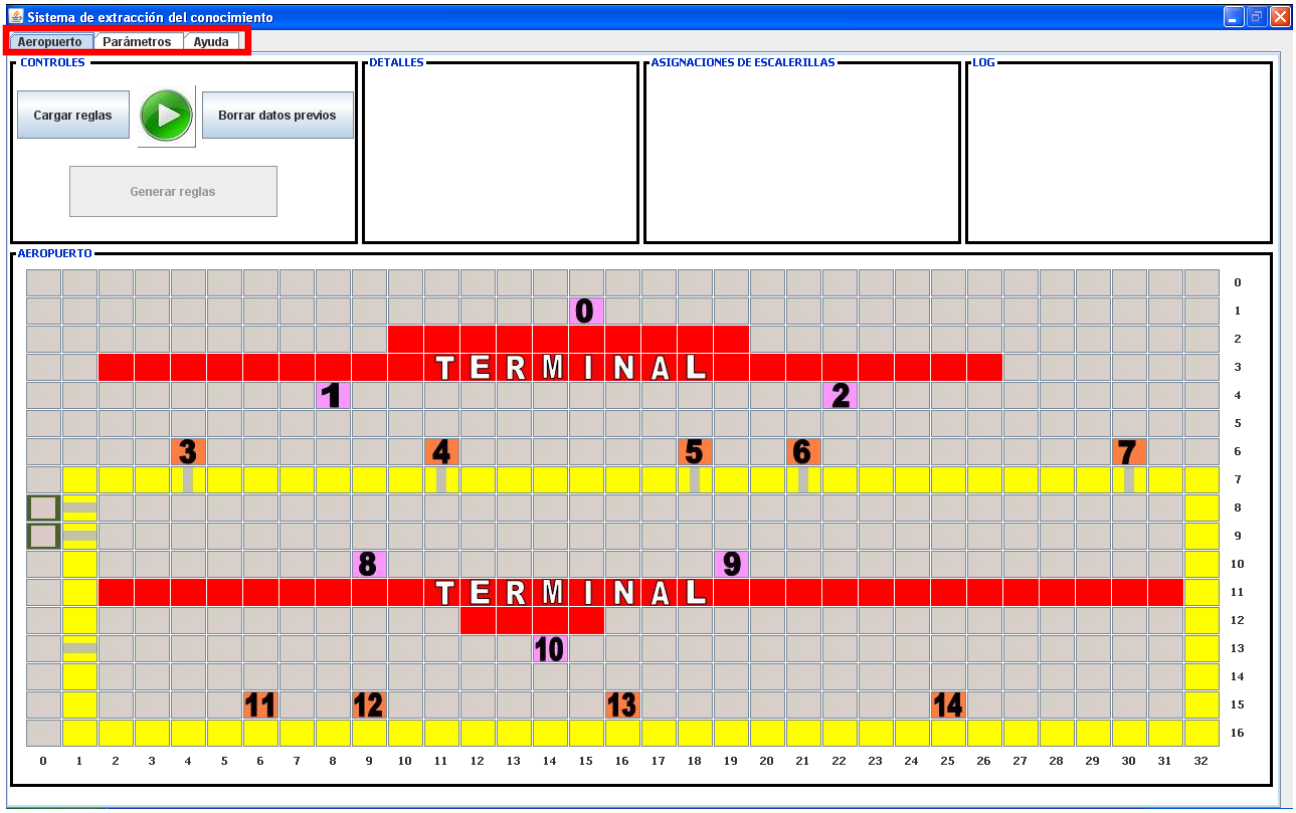


Ilustración 71: Pantalla inicial del simulador aeroportuario

Pestaña ayuda

Si se pincha sobre la pestaña “Ayuda” aparecen los símbolos que pueden ir surgiendo en la simulación así como su significado.

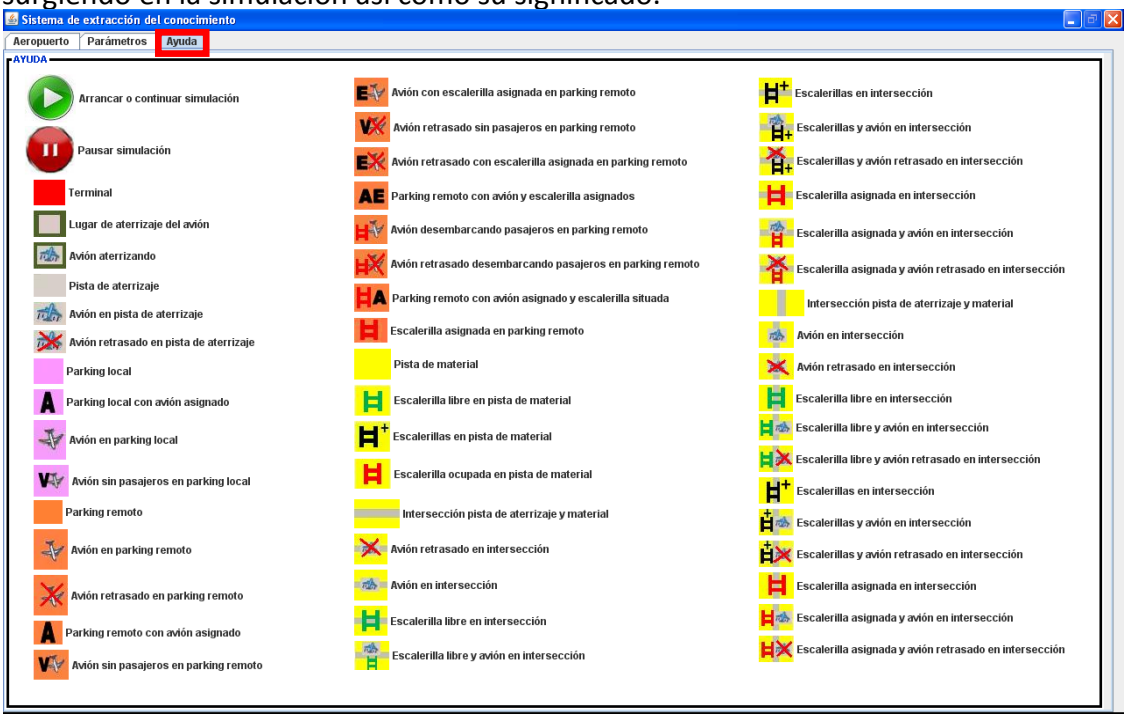


Ilustración 72: Ayuda del simulador aeroportuario

Pestaña parámetros

En la pestaña “Parámetros” se pueden cambiar algunos parámetros de la simulación, ya sea antes de arrancar la misma o durante la misma. Para confirmar los cambios realizados es necesario pulsar sobre el botón “Guardar”. Los parámetros que se pueden cambiar son los siguientes.

- Tiempo mínimo de despegue de los aviones: es el mínimo tiempo posible que está un avión en el aeropuerto (desde que aterriza hasta que se marcha del mismo)

- Tiempo máximo de despegue de los aviones: es el máximo tiempo posible que está un avión en el aeropuerto (desde que aterriza hasta que se marcha del mismo)

- Ciclo de reloj: es el intervalo de tiempo que tiene que pasar para actualizar la simulación

- Número máximo de ciclos permitidos de retraso en el despegue de los aviones: número máximo de ciclos que se permiten que se retrasen los aviones desde su salida inicialmente prevista. Si un avión se retrasa más de estos ciclos la simulación se para y muestra un mensaje

- Probabilidad de aterrizaje de los aviones: número mayor que 0 y menor o igual a 1 que indica el flujo de aviones que aterrizan en el aeropuerto. Cuanto más cercano sea a 0, menos aviones aterrizarán en el mismo, y cuanto más cercano a 1, más aviones aterrizarán

- Distancia simulada de cada área del aeropuerto: número de metros reales que representa cada una de las áreas (casillas) del aeropuerto simulado

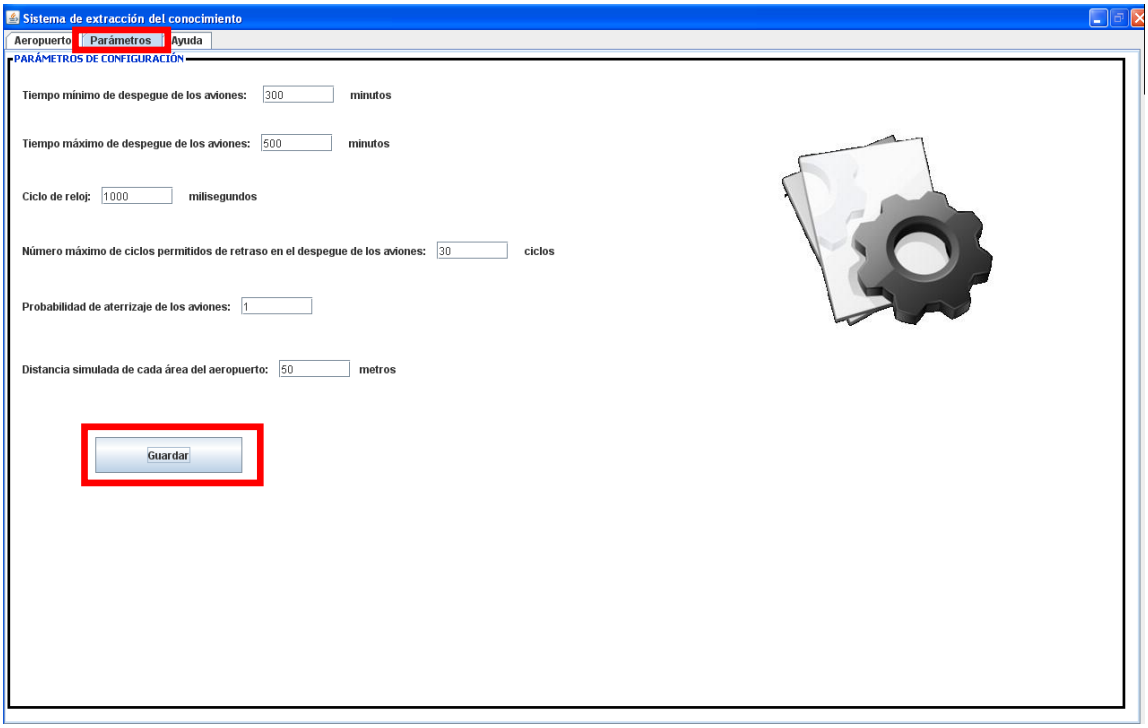


Ilustración 73: Parámetros de configuración del simulador

Pestaña aeropuerto

Finalmente en la pestaña "Aeropuerto" es donde se puede ver la gran mayoría de la funcionalidad del sistema: la sección de "Controles", donde se permite arrancar el sistema en diferentes modos; la sección de "Detalles", donde se visualizan aspectos de los elementos que hay en el aeropuerto (aviones, escalerillas y zonas de estacionamiento); la sección de "Asignaciones de escalerillas", donde se pueden realizar asignaciones y cancelaciones de escalerillas; la sección de "Log", donde se muestra información sobre sucesos importantes en la simulación; y la sección de "Aeropuerto", que es el simulador o consola aeroportuaria en cuestión. Antes de arrancar la simulación, las zonas de estacionamiento, tanto locales en color rosa como remotas en color naranja, están marcadas con un número identificador (este número identificador sirve para saber, cuando se generen los conjuntos de reglas, a qué zona de estacionamiento se está refiriendo en cada conjunto de reglas).

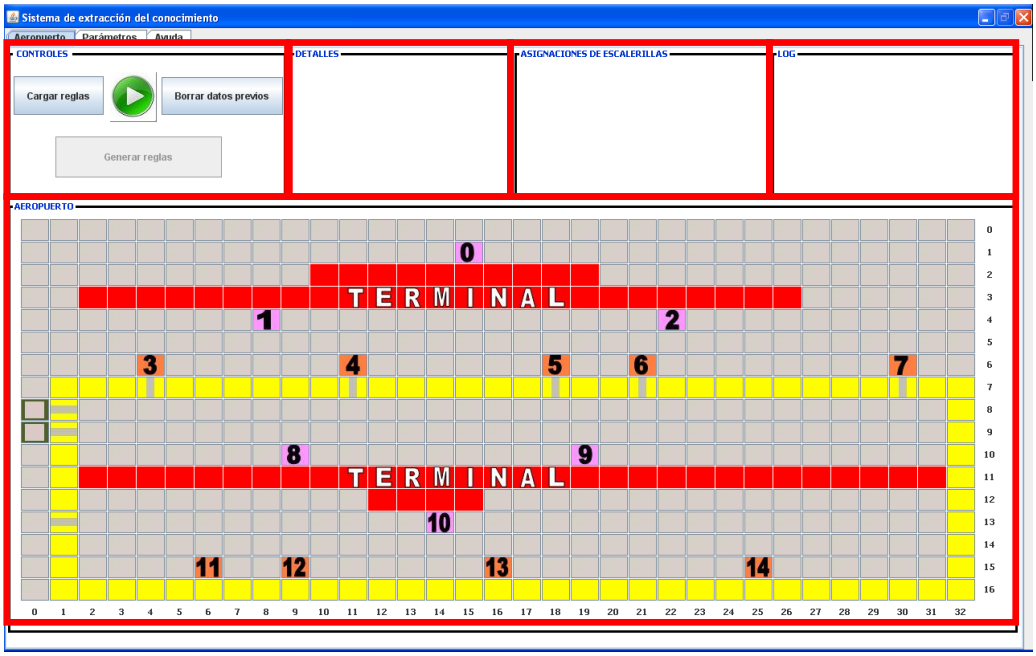


Ilustración 74: Pantalla de simulación del aeropuerto

Sección controles

En la sección de "Controles" hay cuatro botones que permiten realizar diversas acciones.

Ilustración 75: Controles del simulador aeroportuario

- Botón "Cargar reglas": al pulsar sobre este botón se arrancará la simulación en modo de asignación y cancelación de escalerillas automático, es decir, cogerá los conjuntos de reglas generados en una simulación anterior y asignará y cancelará las asignaciones de escalerillas en función de dicho conjunto de reglas. Si no se han generado los conjuntos de reglas se mostrará mensaje de error en la sección de "Log"

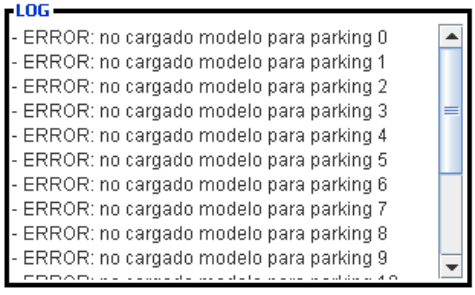


Ilustración 76: Mensaje de error en "Log"

- Botón  : al pulsar sobre este botón se arrancará la simulación en el modo que se permite al usuario asignar o cancelar manualmente las escalerillas. Si ha habido simulaciones previas se tendrán en cuenta los datos recogidos en ellas para generar los conjuntos de reglas, aparte de los datos que se recojan en esta nueva simulación. Tras arrancar la simulación este botón se convertirá en  para que cuando el usuario lo desee pueda pulsar sobre el mismo y pausar la simulación para asignar o cancelar la asignación de una escalerilla o simplemente para ver algo con más detalle. Cuando el usuario desee seguir con la simulación sólo debe pulsar de nuevo sobre el botón 

- Botón "Borrar datos previos": al pulsar sobre este botón se borrará toda la información que se tiene de simulaciones previas (datos, conjuntos de reglas y árboles de decisión J48) y no se tendrán en cuenta en posteriores simulaciones

- Botón "Generar reglas": este botón sólo se encuentra activo tras haber arrancado una simulación en modo de asignación y cancelación manual de escalerillas y encontrarse en pausa dicha simulación. Al pulsar sobre este botón, comenzará el proceso de construir los árboles de decisión J48 y, por tanto, obtener los conjuntos de reglas a partir de estos árboles de decisión J48. Este proceso, dependiendo del número de datos disponible, puede llevar un tiempo considerable hasta que finalice, y, para que el usuario sepa que ha finalizado, se muestra un mensaje en pantalla indicando esta información. Tras acabar este proceso se puede comprobar que habrá 4 carpetas (aparte de la carpeta "dist") en la ruta donde esté "SimuladorAeroportuario.jar": "datos" (que contiene varios ficheros del tipo "datosXX.arff" y en cada fichero está el conjunto de datos o instancias de la simulación de la zona de estacionamiento XX), "evaluacion" (que contiene un único fichero llamado "evaluacion.txt" donde se especifica el número y porcentaje de instancias bien y mal clasificados al testear usando una validación cruzada en 10 partes), "J48" (que contiene varias ficheros del tipo "j48(XX).model" que pueden ser cargados en Weka y en cada fichero está guardado el modelo o árbol de decisión J48 creado para la zona de estacionamiento XX) y la carpeta "reglas" (contiene varios ficheros del tipo "reglasXX.txt" y en cada fichero está el conjunto de reglas de la zona de estacionamiento XX)

Sección detalles

Para ver los detalles o información adicional de algún elemento del simulador aeroportuario no hay más que pinchar sobre él y en la sección de “Detalles” aparecerá dicha información.

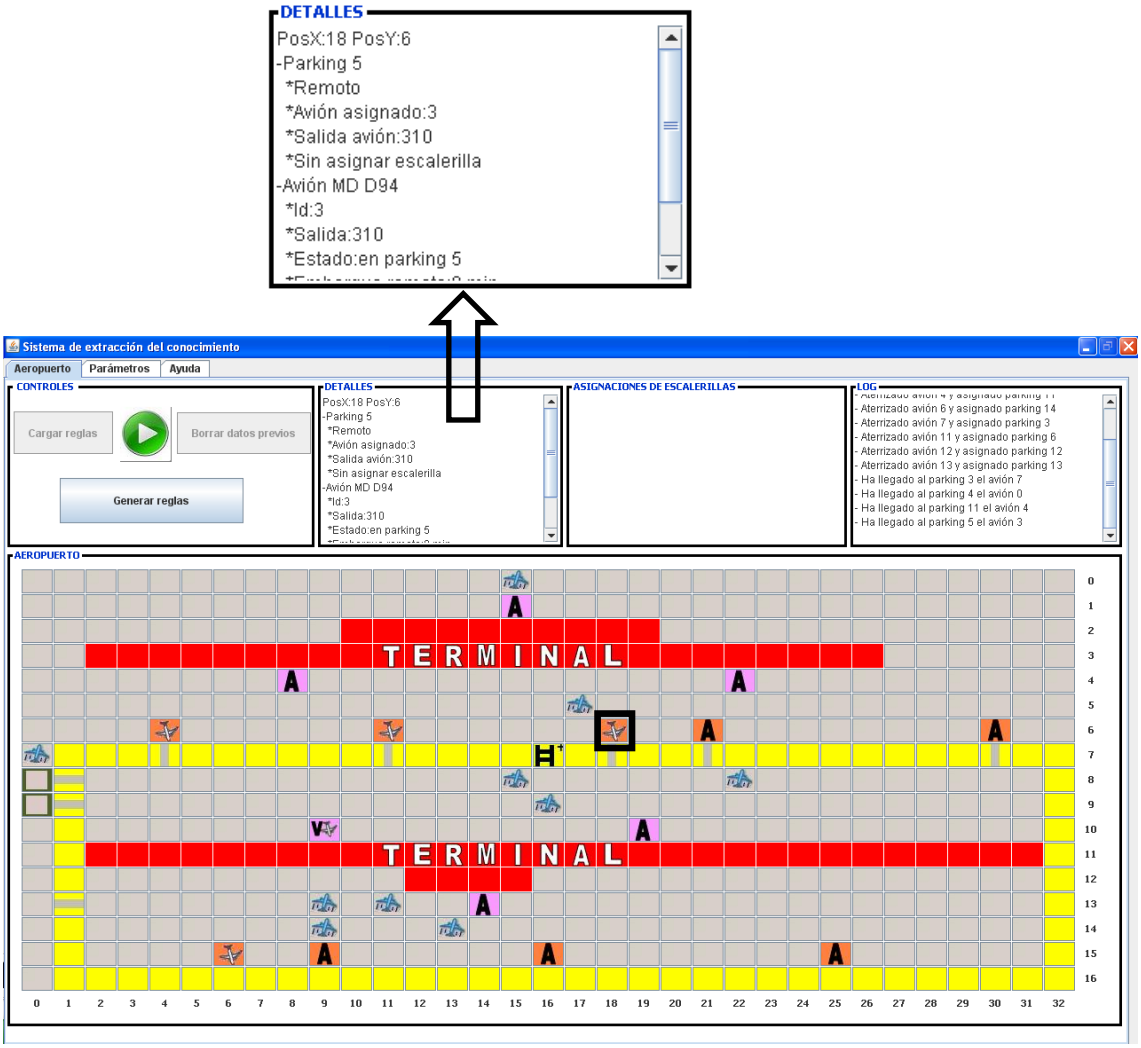


Ilustración 77: Detalles de la simulación

Sección de asignación y cancelación de escalerillas

Para poder asignar y cancelar asignaciones de escalerillas, primero se debe haber arrancado la simulación en modo que se permita al usuario asignar o cancelar manualmente las escalerillas y haber pausado la simulación. Tras esto se hace click sobre una escalerilla y en la sección de “Asignaciones de escalerillas” se selecciona si se quiere realizar la acción de asignar una escalerilla o por el contrario cancelar la asignación de una escalerilla. Después aparecen dos listas desplegables para seleccionar tanto el tipo de escalerilla sobre el que se quiere realizar la acción (ya que se puede haber hecho click sobre un área en el que haya varias escalerillas y, por tanto, es necesario especificar más cuál es la escalerilla a la que se refiere) y la zona de

estacionamiento o parking a la que se le asigna o cancela la escalerilla para realizar o no el servicio de desembarque a los pasajeros del avión que se dirige a esa zona de estacionamiento. En la lista desplegable de las zonas de estacionamiento aparece tanto la posición (con coordenadas “x” e “y”) como el identificador entre paréntesis, mientras que en la lista desplegable donde se selecciona la escalerilla aparece la descripción de la misma y entre paréntesis su identificador. Tras seleccionar la zona de estacionamiento y la escalerilla se pulsa sobre el botón “OK” para confirmar dicha asignación.

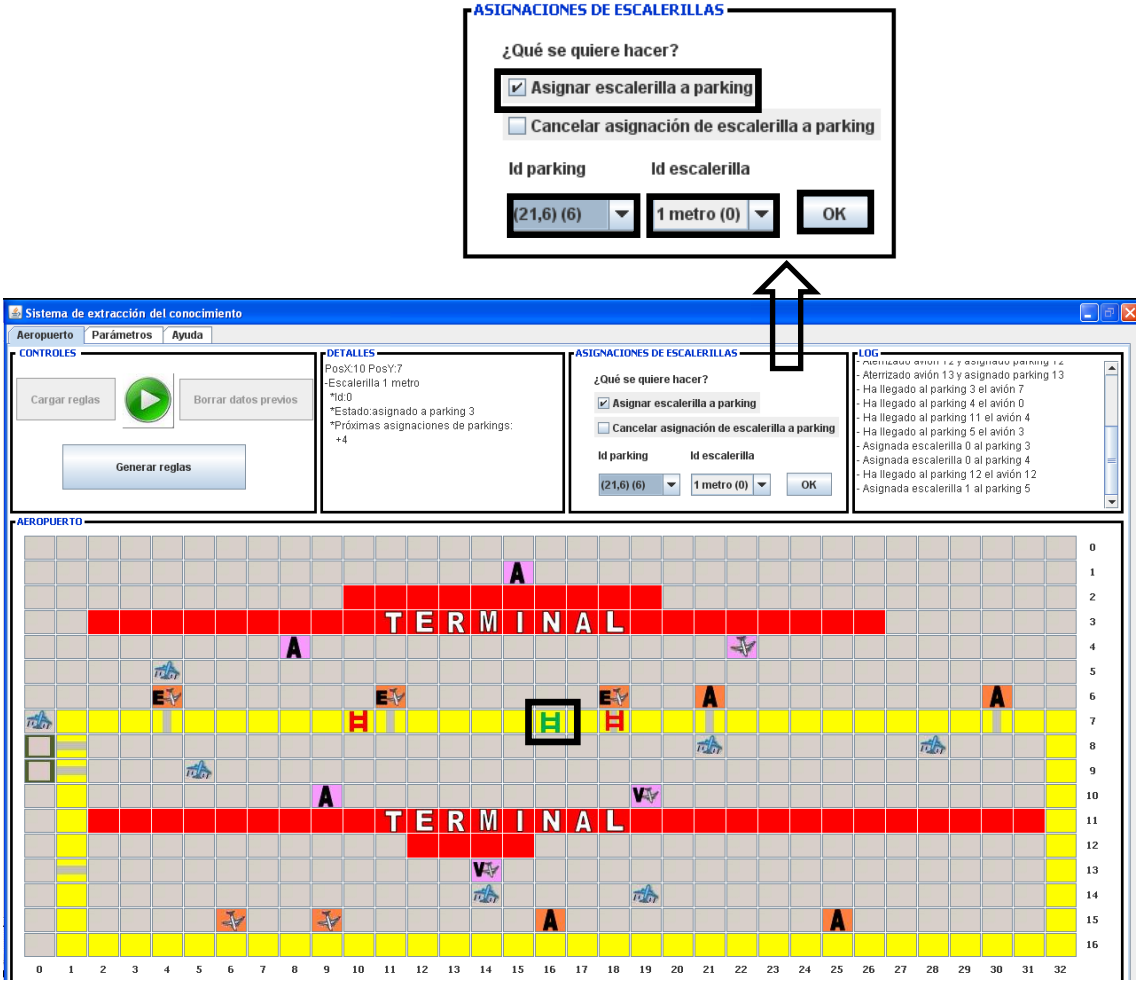


Ilustración 78: Asignación de escalerilla en simulación

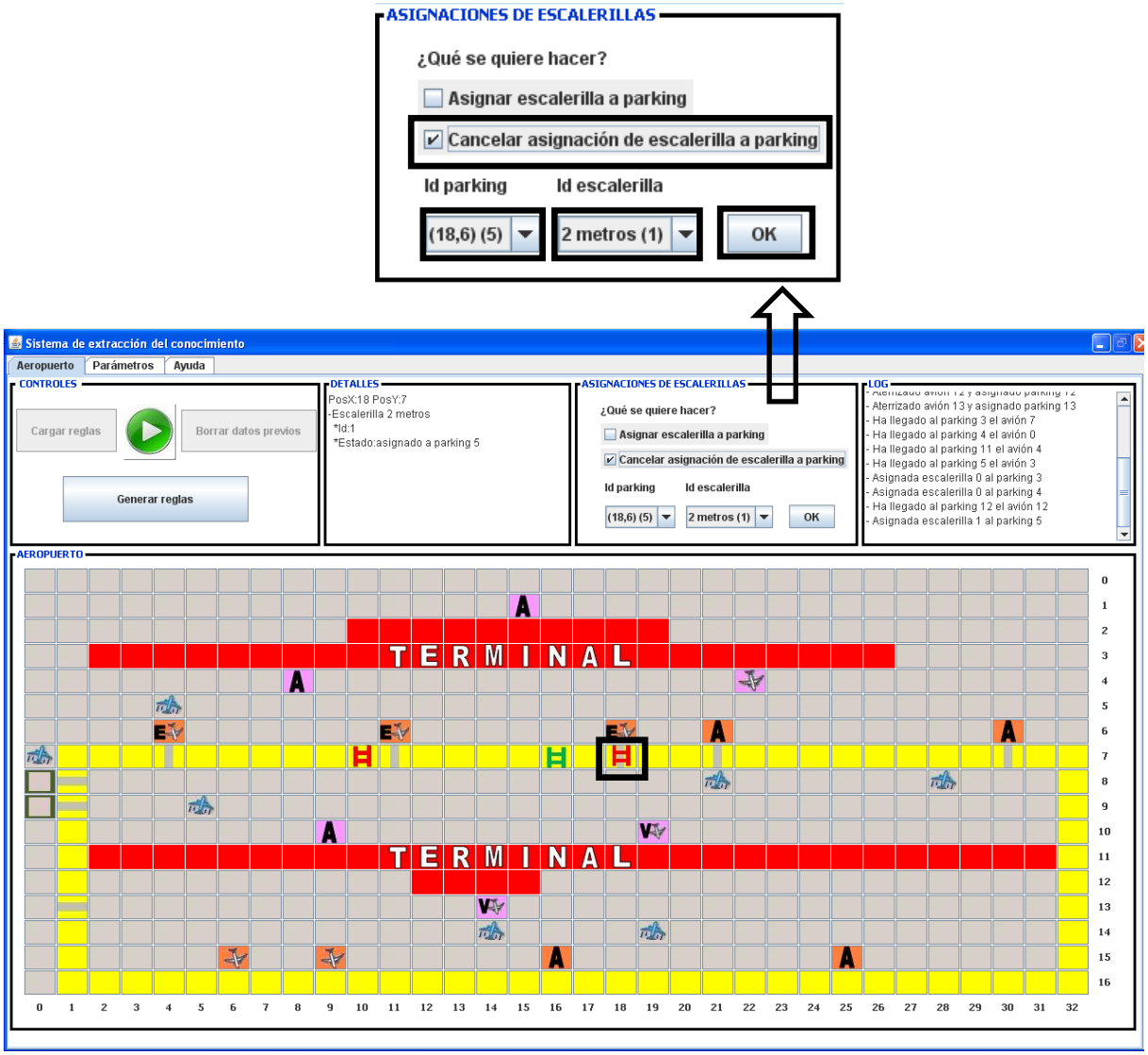


Ilustración 79: Cancelación de escalerilla en simulación

Sección log

En la sección de "Log" se muestra información sobre sucesos importantes en la simulación que afectan a las zonas de estacionamiento remotas (aterrizaje de aviones que van a esas zonas de estacionamiento, asignaciones de escalerillas a esas zonas de estacionamiento, cancelaciones de escalerillas a esas zonas de estacionamiento, etc.).

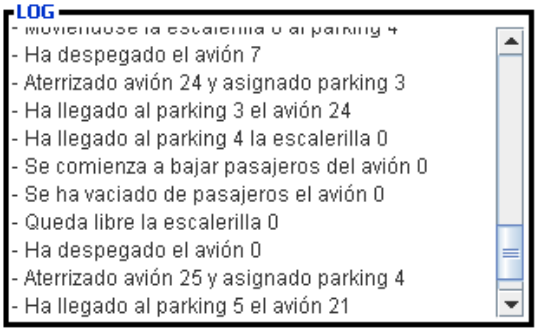


Ilustración 80: Log de la simulación

Sección aeropuerto

En esta sección es donde se ve visualmente el aeropuerto simulado, así como todos los elementos que lo componen y su estado (escalerillas, aviones, pista de material, zonas de estacionamiento locales y remotas y terminales).

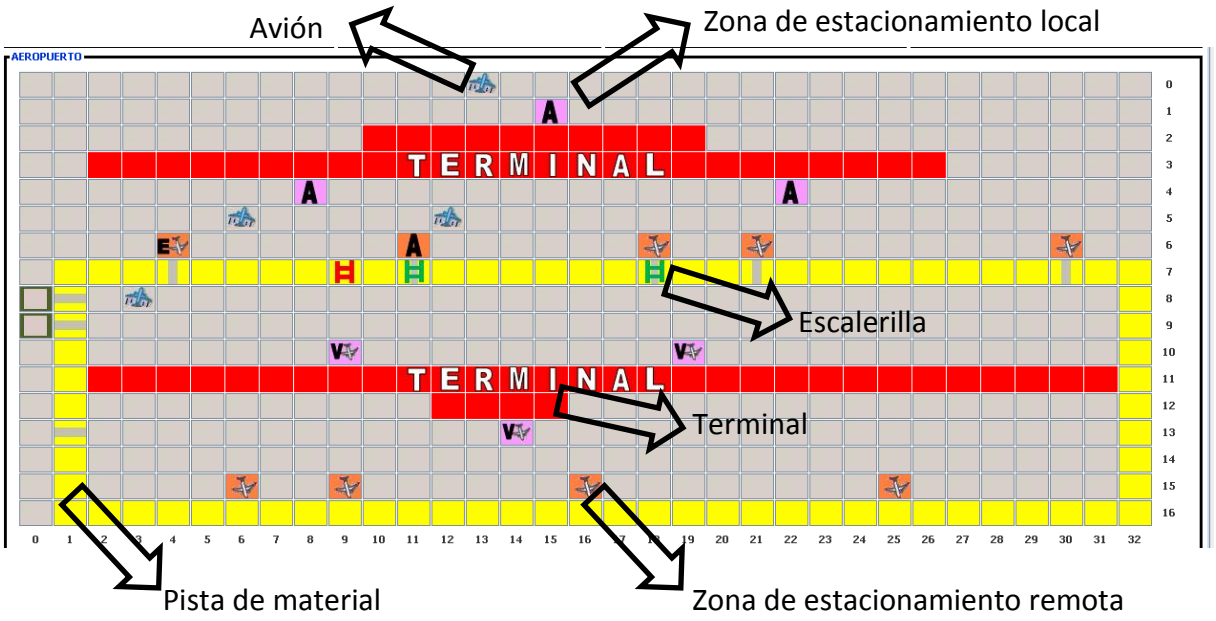


Ilustración 81: Elementos de la simulación

Cierre de la simulación

Para cerrar la simulación hay que dar a la "X" blanca que está dentro de un cuadrado rojo en la parte superior derecha de la aplicación. Es necesario recordar que aunque se cierre la aplicación los datos de la simulación serán guardados y estarán disponibles en la próxima ejecución de la misma.

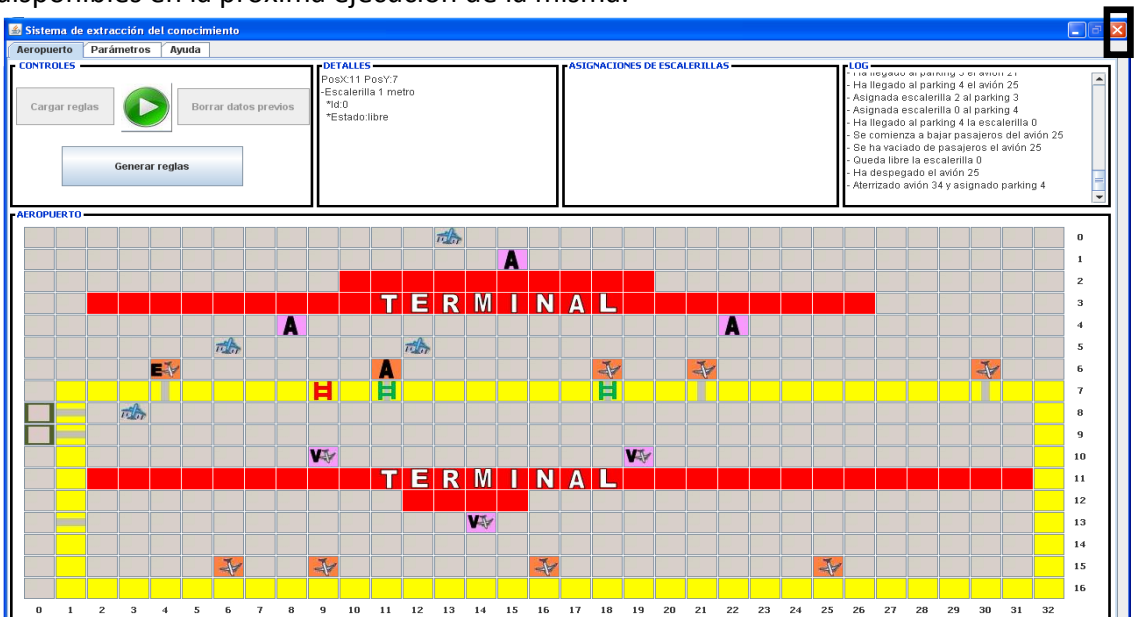


Ilustración 82: Cierre de la simulación

Bibliografía

[1] Dirección de Operaciones y Sistemas de Red. Departamento de Estadística Operativa. *Tráfico de pasajeros, operaciones y carga en los aeropuertos españoles* (2011).
http://www.aena-aeropuertos.es/csee/ccurl/134/389/anualProvisionales_2011.pdf
[Accedido el 25 de Marzo de 2012]

[2] Real Decreto 1161/1999, Ministerio de Fomento, 2 de Julio de 1999.
http://www.boe.es/boe/dias/1999/07/15/pdfs/A26786-26791.pdf
[Accedido el 25 de Marzo de 2012]

[3] Mariano Domingo Calvo. *Descubrir el handling aeroportuario*. Edición Centro de Documentación y Publicaciones de AENA, Madrid, 2005.

[4] Web oficial de Java, http://java.com/es/about/ [Accedido el 30 de Marzo de 2012]

[5] Web oficial de la herramienta NetBeans, http://netbeans.org/
[Accedido el 30 de Marzo de 2012]

[6] Web oficial de la herramienta Weka, http://www.cs.waikato.ac.nz/ml/weka/
[Accedido el 30 de Marzo de 2012]

[7] Iván Tejada Anguiano. *Descubrir los aeropuertos*. Edición Centro de Documentación y Publicaciones de AENA, Madrid, 2008.

[8] Fernando Muñoz. *Tecnología al servicio del pasajero. Aena dirige sus miras hacia un middleware de integración*. Septiembre de 2002.
http://www.idg.es/computerworld/Tecnologia-al-servicio-del-pasajero.Aena-dirige-su/seccion-/articulo-141956 [Accedido el 1 de Abril de 2012]

[9] Pilar Cejudo. *Curso de Especialista Superior en Gestión Aeronáutica y Aeroportuaria*. Curso 2009-2010.
http://193.146.228.22/ga/jar/aeropuertos/Tema%205%20D%20OPS%20Aeroportuarias.pdf [Accedido el 1 de Abril de 2012]

[10] Quintiq – *Airport Planning and Scheduling.*
http://www.airport-technology.com/contractors/consult/quintiq/
[Accedido el 2 de Abril de 2012]

[11] Sita WorkBridge – *Airport Resource Management Software / Systems.*
http://www.airport-suppliers.com/supplier/SITA_WorkBridge_AS/
[Accedido el 3 de Abril de 2012]

[12] Web oficial de Terminal Systems S.A, http://www.terminalsystems.eu/
[Accedido el 5 de Abril de 2012]

[13] Web oficial de Asseco Spain, http://asseco.com/es/
[Accedido el 10 de Abril de 2012]

[14] Jiang Hua. *Study on Knowledge Acquisition Techniques*. Segunda conferencia internacional de la aplicación de la tecnología de información inteligente, Diciembre, 2008, páginas 181-185.

[15] *Tema 3 – Adquisición del Conocimiento*. Asignatura Ingeniería del Conocimiento. Universidad Carlos III de Madrid, 2006.
http://www.giaa.inf.uc3m.es/docencia/II/IConocimiento/tema3.pdf
[Accedido el 12 de Abril de 2012]

[16] N. Nezafati, A. Khadivar, E. Afarideh y S. Mohammad Javad Jalali. *A Method for Human Driven Knowledge Acquisition (Case Study in a Petrochemical Company)*. Conferencia internacional de IEEE, 2007.

[17] *Knowledge Acquisition,* 2003, http://www.epistemics.co.uk/Notes/63-0-0.htm
[Accedido el 12 de Abril de 2012]

[18] Xiao-Bing Hu y Ezequiel Di Paolo. *A ripple-spreading genetic algorithm for the airport gate assignment problem*. Congreso IEEE de la computación evolutiva, 2009.

[19] M. Kamruzzaman y Md. Monirul Islam. *Extraction of Symbolic Rules from Artificial Neural Networks*. Academia del Mundo de la Ciencia, Ingeniería y Tecnología, Octubre de 2010.

[20] Robert Cattral, Franz Oppacher, Dwight Deugo. *Rule Adquisition with a Genetic Algorithm*. Escuela de la Ciencia de la Computación, Universidad de Carleton, 1999.

[21] Raul T. Santos, Júlio C. Nievola, Alex A. Freitas. *Extracting Comprehensible Rules from Neural Networks via Genetic Algorithms*. Conferencia del IEEE, 2000.

[22] J.H. Ang, K.C. Tan, A.A. Mamun. *An evolutionary memetic algorithm for rule extraction*. Departamento de Ingeniería Eléctrica y de Ordenadores, Universidad Nacional de Singapur, 2010.

[23] Eduardo R. Hruschka, Nelson F.F. Ebecken. *Extracting rules from multilayer perceptrons in classification problems: a clustering-based approach*. Séptima conferencia brasileña de redes de neuronas, Mayo de 2006.

[24] U. Markowska-Kaczmar, W. Trelak. *Fuzzy logic and evolutionary algorithm – two techniques in rule extraction from neural networks*. Onceava conferencia europea de redes de neuronas, Septiembre de 2004.

[25] Koichi Odajima, Yoichi Hayashi, Gong Tianxia, Rudy Setiono. *Greedy rule generation from discrete data and its use in neural network rule extraction*. Universidad Nacional de Singapur, 2008.

[26] I.H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes y S.J. Cunningham. *Weka: Practical Machine Learning Tools and Tecnhiques with Java Implementations*. Departamento de la Ciencia de la Computación, Universidad de Waikato, Nueva Zelanda,2009.
http://www.cs.waikato.ac.nz/~eibe/pubs/99IHW-EF-LT-MH-GH-SJC-Tools-Java.pdf
[Accedido el 14 de Abril de 2012]

[27] Antón Astudillo Zulueta. *Diseño de un sistema automático de inversión en bolsa basado en técnicas de predicción de series temporales*. Universidad Pontificia de Comillas, Escuela técnica superior de Ingeniería (ICAI), Junio de 2009. http://www.iit.upcomillas.es/pfc/resumenes/4a438ec8e82f0.pdf
[Accedido el 25 de Abril de 2012]

[28] Lior Rokach y Oded Z. Maimon. *Data mining with decision trees: theory and applications*. Serie de la Percepción de la Máquina e Inteligencia Artificial, World Scientific Publishing Co Pte Ltd volumen 69, capítulo 1, Diciembre de 2007. http://www.worldscibooks.com/etextbook/6604/6604_chap01.pdf
[Accedido el 14 de Abril de 2012]

[29] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Publicaciones Morgan Kaufmann, Diciembre de 1992.

[30] Julio Villena Román. *Resolución de problemas*. Inteligencia en Redes de Comunicaciones,2008.
http://www.it.uc3m.es/rcrespo/docencia/irc/apuntes/02%20Resolucion%20de%20problemas.pdf [Accedido el 14 de Abril de 2012]

[31] *Primero el mejor (best-first)*,
http://www.monografias.com/trabajos/iartificial/pagina3_2.htm
[Accedido el 15 de Abril de 2012]

[32] Wayne Iba y Pat Langley. *Induction of One-Level Decision Trees*. Procedimientos de la novena conferencia internacional de la máquina de aprendizaje, Aberdeen, Escocia, Julio de 1992, http://lyonesse.stanford.edu/~langley/papers/stump.ml92.pdf
[Accedido el 16 de Abril de 2012]

[33] API de la herramienta Weka, http://weka.sourceforge.net/doc/
[Accedido el 16 de Abril de 2012]

[34] Roman Timofeev. *Classification and Regression Trees (CART). Theory and Applications*. Universidad de Humboldt, Berlín, Diciembre de 2004.

http://edoc.hu-berlin.de/master/timofeev-roman-2004-12-20/PDF/timofeev.pdf
[Accedido el 16 de Abril de 2012]

[35] Bruno Escarcega. *Herramientas para la definición del Modelo*. Enero de 2008. http://www.gravitar.biz/index.php/bi/herramientas-para-la-definicion-del-modelo/
[Accedido el 16 de Abril de 2012]

[36] Dánel Sánchez Tarragó. *Pronóstico de supervivencia de infarto cerebral aterotrombótico usando aprendizaje automatizado*. Sexto congreso internacional de informática en salud. Noviembre de 2006.

[37] *¿Qué es Java?*, http://java.ciberaula.com/articulo/que_es_java [Accedido el 17 de Abril de 2012]

[38] Web oficial de descarga del JRE de Java,
http://www.oracle.com/technetwork/java/javase/downloads/index.html
[Accedido el 20 de Abril de 2012]

[39] Web oficial de la herramienta Microsoft Access,
http://office.microsoft.com/es-es/access/ [Accedido el 22 de Abril de 2012]

Acrónimos

AENA Aeropuertos Españoles y Navegación Aérea

ANN Artificial Neural Networks

AODB Airport and Obstacle DataBase

API Application Programming Interface

ARFF Attribute-Relation File Format

ASCII American Standard Code for Information Interchange

BF Best First

CART Classification And Regression Tree

CGA Clustering Genetic Algorithm

EMA Evolutionary Memetic Algorithm

GA Genetic Algorithm

GAP Gate Assignment Problem

GRG Greedy Rule Generation

HP Hewlett-Packard

IP Internet Protocol

IREP Incremental Reduced Error Pruning

JRE Java Runtime Environment

MP Multilayer Perceptron

NN Neural Networks

NNGE Nearest Neighbor with Generalization

ODBC Open DataBase Connectivity

PC Personal Computer

RAGA Rule Acquisition Genetic Algorithm

REANN Rule Extraction Artificial Neural Networks

REP Reduced Error Pruning

REX Rule EXtraction

RIPPER Repeated Incremental Pruning to Produce Error Reduction

SADAMA Sistema de Ayuda a la Asignación de Medios Aeroportuarios

TCP Transmission Control Protocol

UCA Uso Compartido de AENA

Glosario de términos

AENA: entidad pública empresarial que gestiona los aeropuertos civiles de interés general y las instalaciones y redes de ayuda a la navegación aérea en España.

Aeronave: aparato que puede sustentarse en la atmósfera por reacciones del aire que no sean las reacciones del mismo contra la superficie de la tierra.

Aeropuerto: área preparada para el aterrizaje, despegue y movimiento en tierra de aeronaves, que cuenta con las instalaciones necesarias para atender tráfico comercial.

Algoritmo evolutivo: métodos de optimización y búsqueda de soluciones basados en los postulados de la evolución biológica.

Algoritmo genético: algoritmo que hace evolucionar una población de individuos sometiéndola a acciones aleatorias (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Atributo: especificación que define una propiedad de un objeto.

Calzo: cuña de madera o metal que se coloca delante y detrás de las ruedas de un avión posado en tierra para impedir cualquier clase de movimiento.

Carga aérea: cualquier bien transportado o para ser transportado en una aeronave.

Caso de uso: descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso.

Clúster: agrupación de una serie de objetos con unas características similares.

Compañía aérea: empresa de aviación que realiza el transporte de pasajeros, equipajes y carga.

Conjunto difuso: conjunto que puede contener elementos de forma parcial.

Conocimiento tácito: forma de conocimiento que es completa o parcialmente inexplicable, que no se ha podido o sabido explicitar o comunicar verbal o visualmente

Conocimiento explícito: aquel conocimiento que ha sido o puede ser articulado, codificado y almacenado en algún tipo de medio.

Coordinador de servicios: personal del "handling" aeroportuario encargado de asignar los recursos existentes en el aeropuerto (personal y material) a los diferentes servicios que se deben prestar en el mismo.

Desembarque: acción por el cuál una persona, equipaje o carga es sacado de algún tipo de embarcación (avión, barco, etc.) y puesto en tierra.

Diagrama de clases: tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

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

Elitismo: técnica que consiste en guardar siempre el mejor elemento de la población de un algoritmo genético sin hacerle ningún cambio.

Embarque: acción por el cuál una persona, equipaje o carga se introduce en algún tipo de embarcación (avión, barco, etc.) para ser transportado de un lugar a otro.

Escala: tiempo que transcurre entre la llegada y la salida de una aeronave en un aeropuerto.

Escalerilla: equipo necesario para el acceso o salida de los pasajeros, al o desde el avión, cuando no se dispone de pasarelas de embarque/desembarque.

"Handling" aeroportuario: prestación de un conjunto determinado de servicios aeroportuarios a las aeronaves, pasajeros, equipajes y mercancías en un aeropuerto y que son necesarios para el intercambio del modo de transporte aéreo al terrestre y viceversa, así como del aéreo al aéreo.

Heurística: manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

Host: término usado para referirse a las computadoras conectadas a una red, que proveen y utilizan servicios de ella.

Instancia: ejemplo compuesto por los valores de sus atributos.

Jardinera: equipos que se utilizan para el traslado de pasajeros entre el avión y la terminal y viceversa.

Minería de datos: proceso de extracción no trivial de información que reside de manera implícita en los datos a través de la preparación, sondeo y exploración de los datos.

Modelo de datos: lenguaje orientado a describir la estructura, restricciones de integridad y operaciones de manipulación de los datos en una base de datos.

Modelo de minería de datos: solución que da la minería de datos a un problema.

Parking: área del aeropuerto reservado para aparcar el avión después del aterrizaje del mismo. También es conocido como zona de estacionamiento.

Pasarela: túnel de unión que conecta directamente el edificio terminal de pasajeros con las portezuelas de embarque de un avión.

Patrón singleton: patrón de programación diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

Poda: técnica que se suele interpretar como un árbol de soluciones, donde cada rama nos lleva a una posible solución posterior a la actual. El algoritmo de esta técnica se encarga de detectar en qué ramificación las soluciones dadas ya no están siendo óptimas, para "podar" esa rama del árbol y no continuar malgastando recursos y procesos en casos que se alejan de la solución óptima.

Prestador de servicios: personal aeroportuario encargado de llevar a cabo el "handling" aeroportuario.

Red de neuronas: sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida.

Requisito: necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio.

Simulador: aparato que permite la reproducción de un sistema, reproduciendo sensaciones que en realidad no están sucediendo.

Sistema experto: sistema que emula el razonamiento de un experto en un dominio concreto y en ocasiones son usados por éstos. Con los sistemas expertos se busca una mejor calidad y rapidez en las respuestas, dando así lugar a una mejora de la productividad del experto.

Terminal: edificio para uso de pasajeros o carga.

Zona de estacionamiento: área del aeropuerto reservado para aparcar el avión después del aterrizaje del mismo. También es conocido como parking. Si la zona de estacionamiento es local significa que está colindante con una terminal y si es remota significa que no lo está.